Universidad de las Ciencias Informáticas

Facultad 6

Título: Ambiente integrado para el modelado y ejecución de procesos de gestión de la información. Herramienta web para la modelación de los procesos.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores:

Omar Martínez Díaz.

Daniel Garófalo Hernández.

Tutores:

Ing. Aida Portelles Valdes.

Lic. Yoandry Pacheco Aguila.

Cotutor:

Ing. Aldis Joan Abreu Medina.

Junio, 2010.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Omar Martínez Díaz

_____

Firma del Autor

Lic. Yoandry Pacheco Águila.

_____

Firma del Tutor

Daniel Garófalo Hernández.

_____

Firma del Autor

Ing. Aida Portelles Valdes.

_____

Firma del Tutor

Ing. Aldis Joan Abreu Medina

_____

Firma del Cotutor

DATOS DE CONTACTO

Lic. Yoandry Pacheco Águila.

Universidad de Ciencias Informáticas, Ciudad de la Habana, Cuba.

Email: andypa@uci.cu.

Ing. Aida Portelles Valdes.

Universidad de Ciencias Informáticas, Ciudad de la Habana, Cuba.

Email: aportelles@uci.cu.

Ing. Aldis Joan Abreu Medina

Universidad de Ciencias Informáticas, Ciudad de la Habana, Cuba.

Email: ajabreu@uci.cu.

AGRADECIMIENTOS

Agradecemos a todos los que de una forma u otra tuvieron que ver con el resultado de esta tesis. A todos los que estuvieron con nosotros durante estos meses, dándonos apoyo y ayudándonos en todo.

A los tesistas de las otras dos tesis, que trabajamos como un equipo para lograr nuestros objetivos, a Keyler, Oscar y Glennis, gracias. A Yanet Hernández que no fueron pocas las veces que la consultamos, a cualquier hora, para aclarar dudas, gracias por tu disposición y atención. A nuestros tutores y cotutores, Aida y Aldis.

Queremos dar un agradecimiento especial a nuestro tutor Yoandry Pacheco, por toda la ayuda que nos brindó. Por permitirnos ocupar su tiempo, y siempre estar dispuesto a colaborar con nosotros, muchas gracias.

DEDICATORIA

De Omar:

Esta tesis esta dedicada a la memoria de mi abuela, que lamentablemente no pudo ver su más añorado sueño. Sé que estaría muy contenta y muy orgullosa de mí, al ver que todo lo que me enseñó surtió efecto.

A mi mamá por siempre estar a mi lado, y por esforzarse tanto para darme todo el apoyo del mundo. Sé que algún día haré lo mismo por ti.

A mi padre, mi mejor amigo, que siempre me ha comprendido y ha estado a mi lado para todas las cosas, aun en los momentos más difíciles.

De Daniel:

Este trabajo de diploma está dedicado a todos los que en el transcurso de vida me han ayudado a crecer emocional e intelectualmente; pero en especial a los que aún lo hacen.

A mi mamá que a pesar de la apariencia, sé que siempre confió en mí y me aconsejo y apoyó en todo momento, sin escatimar en esfuerzos. Por darme fuerzas y estar ahí, gracias.

A mi novia que está siempre a mi lado, dándome ánimos, siendo mi compañera. Me enseñó la lógica más absoluta, por difícil que sea el camino siempre habrá alguien que te acompañe a recorrerlos.

A mi hermano, por ser más que un hermano, un padre. Siempre ha sido mi ejemplo a seguir, espero algún día llegar a su nivel, todo intelectual, como humano.

A mi padre, por ser mi guía en todo momento, si he podido llegar a este día es gracias a ti. Todo lo que he aprendido te lo debo. Si soy una mejor persona, siempre será gracias a tus enseñanzas. Por todo gracias.

RESUMEN

Los modeladores de procesos no son más que herramientas creadas con el objetivo de representar la realidad de los procesos en las empresas. Siendo su función principal: el seguimiento y control de los procesos de una empresa, para mejorar la calidad de los mismos. En el presente trabajo de diploma se tiene como objetivo desarrollar una herramienta web que permita la modelación gráfica de los procesos de gestión de la información en los centros biotecnológicos.

Se emplea como metodología de desarrollo RUP, Proceso Unificado de Desarrollo de Software. Los artefactos se generaron usando como lenguaje de modelado UML, Lenguaje Unificado de Modelado y auxiliados por el Visual Paradigm como herramienta de Ingeniería de Software Asistida por Computadora. Como Entorno Integrado de Desarrollo se utilizó el NetBeans en su versión 6.8.

Este modelador de procesos cuenta con un parte encargada de la gestión de los mismos y otra más importante para su modelación. Haciendo uso del estándar BPMN, Notación de Modelado de Procesos del Negocio, se puede representar gráficamente la realidad del ciclo de vida de los modelos de recogida de datos dentro de los centros biotecnológicos. Además de llevar constancia de la última persona que gestionó y modeló un proceso.

Se seleccionaron los elementos de la notación BPMN fundamentales para el modelado. Considerándose estos elementos como los más representativos de la notación, lo que facilita para los usuarios finales de la aplicación, llevar a cabo el modelado.

PALABRAS CLAVES

BPMN, procesos de gestión de la información, Seguimiento y control de los procesos de una empresa.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El desarrollo científico cubano, y en particular el biotecnológico, es uno de los grandes logros de la Revolución. Cuba puede ser considerada en este campo, como un país del primer mundo, ya que en poco más de dos décadas puede exhibir importantes logros, que por su originalidad en la concepción, y buenos resultados pueden competir con las grandes compañías transnacionales.

Para Cuba el desarrollo científico siempre ha sido prioridad. Por tal razón se han creado un grupo de polos científicos a lo largo de la isla, que ven mejorado y facilitado su quehacer diario con los sistemas informáticos. En el mundo actual el desarrollo de los sistemas de gestión de la información unido a la posibilidad de automatización y robotización ha traído consigo un enorme incremento de la productividad. Estos sistemas resuelven una variedad de necesidades de los laboratorios, para cumplir con las normas y requisitos de calidad necesarios para ser competentes en el mercado mundial.

La trazabilidad, es uno de los requisitos que los sistemas de gestión de la información deben cumplir. Esto posibilita que todo proceso que se desarrolle sea monitoreado, lo que permite conocer qué persona o instrumento llevó a cabo una acción sobre el proceso, el momento en que ocurrió, y cuál fue el resultado que provocó.

Otra de las características que deben cumplir los sistemas de gestión de la información, es la flexibilidad, pues los laboratorios cambian sus características con regularidad. Es por eso que se necesita un gran poder de adaptabilidad para las situaciones más variadas. Este es uno de los requisitos que muchos de los sistemas existentes no cumplen, ya que son desarrollados para un negocio específico.

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir estos sistemas. Los mismos son capaces de facilitar el control del elevado volumen de información que generan los centros científicos.

Estos centros están formados por varios laboratorios, en los cuales se va recopilando información, que se obtiene a través de los modelos de recogida de datos. Actualmente no se cuenta con la posibilidad de seguir y controlar estos procesos, que son largos y muy complejos, con muchos puntos de contactos entre sí y donde interactúan muchas personas. Esto provoca que sea muy complicado de organizar y establecer el flujo al que son sometidos los modelos de recogida de datos, para que la obtención de la información requerida, contribuya a que los productos sean liberados con la calidad que establecen los estándares internacionales. Esta dificultad provoca que si alguno de estos procesos sufriera modificaciones, debido a

que para competir con éxito en el mercado, se necesita cumplir con mayores requisitos que avalen la calidad del producto, se debe emplear un tiempo extra en la realización de transformaciones en la organización, que provocan demoras en el descubrimiento y comercialización de los productos.

Por todo lo anteriormente expuesto se define como **problema científico**: ¿Cómo contribuir al seguimiento y control de los procesos de gestión de la información realizados en los centros de biotecnología para elevar la calidad de sus productos?

El problema planteado se enmarca en el **objeto de estudio**: Procesos de gestión de la información de los centros biotecnológicos.

Campo de acción: Modelado de los procesos de gestión de la información en los centros biotecnológicos.

Objetivo General: Desarrollar una herramienta web para la modelación de procesos de gestión de la información que contribuya al seguimiento y control para elevar la calidad de los productos en los centros de biotecnología.

Como **Objetivos Específicos:**

- Identificar las funcionalidades de la herramienta para la modelación de procesos de gestión de la información.
- Realizar el diseño de la herramienta para la modelación de procesos de gestión de la información.
- Realizar la implementación de la herramienta para la modelación de procesos de gestión de la información.

Para cumplir con los objetivos específicos, se trazaron las siguientes **tareas:**

- Revisión de las diferentes herramientas existentes en el mundo para la modelación de procesos utilizando BPMN como notación.
- Revisión de las tendencias y tecnologías para desarrollar una herramienta que facilite la modelación online de procesos de gestión de la información.
- Confección del modelo del dominio.
- Identificación de los requerimientos.
- Realización de los diagramas de clases del diseño de la herramienta de modelación de procesos de gestión de la información.
- Realización del diagrama de clases persistentes.

- Realización de los diagramas de despliegue y de componentes para la implementación de la herramienta para la modelación de los procesos de gestión de la información.
- Implementación de los componentes definidos.
- Realización de pruebas funcionales.

Este documento estará estructurado de la siguiente manera:

Capítulo 1: “Fundamentación Teórica”: En este capítulo se explicarán en detalles las funcionalidades de un modelador de procesos. Cuenta además con un análisis de otras aplicaciones para la modelación de procesos, así como algunas referencias a las tendencias y tecnologías actuales de notaciones para la modelación de procesos, haciendo énfasis en aquellas que se utilizarán en la solución.

Capítulo 2:**“**Características del Sistema**”:** En este capítulo se realizará la descripción del sistema a desarrollar para lograr una mayor claridad y comprensión por parte de los desarrolladores. Los puntos que se abordarán son los siguientes: descripción del modelo de Dominio, especificación de los requisitos funcionales y no funcionales, representación de los actores y casos de uso del sistema a través de un diagrama de casos de uso del sistema, así como la descripción textual de cada uno de ellos, y los patrones de casos de uso a utilizar.

Capítulo 3: “Análisis y Diseño”: En este capítulo se realiza el diagrama de clases del diseño con la pertinente descripción de los paquetes que lo conforman, incluyendo los estilos y los patrones de diseño seleccionados, además de explicar cómo se adaptan al diseño del sistema. También en dicho capítulo se encuentran presentes el diagrama de despliegue, el diagrama de clases persistentes y el modelo de datos relacional de la base de datos en el cual se almacena toda la información que se procesa y finalmente se describen las tablas que componen este último.

Capítulo 4: “Implementación y Prueba”: En este capítulo se describirá la implementación de la herramienta para la modelación gráfica de los procesos a partir del diagrama de componentes y se determinan las pruebas a realizar en el sistema para verificar su integridad y si se ajusta a los requerimientos planteados por el cliente. Además, se presentan varias pantallas de la aplicación donde se muestra el sistema dando cumpliendo a cada uno de los requisitos planteados en los requerimientos funcionales, así como los errores más comunes que son mostrados al cliente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se explicarán en detalles las funcionalidades de un modelador de procesos. Cuenta además con un análisis de otras aplicaciones para la modelación de procesos, así como algunas referencias a las tendencias y tecnologías actuales de notaciones para la modelación de procesos, haciendo énfasis en aquellas que se utilizarán en la solución.

1.1 Modelado de procesos.

Los procesos son cada una de las acciones que intervienen y se interrelacionan en el sistema y que permiten la evolución del ciclo de vida de la información, donde las entradas a un proceso del sistema pueden constituir la salida de otro o viceversa.

El modelo EFQM (*European Foundation for Quality Management*) define un proceso como la "sucesión de actividades en el tiempo con un fin definido; organización lógica de personas, materiales, energía, equipos y procedimientos en actividades de trabajo diseñadas para generar un resultado específico… Los procesos pueden ser tangibles, por ejemplo, la introducción de la información en bases de datos, e intangibles, como sucede con los análisis de contenidos. Estos personifican a la organización mediante cada una de las etapas en que se realizan… permiten el perfeccionamiento y mejoramiento de la eficacia general de toda organización en tanto se gestionen debidamente hacia la consecución de una meta". (1)

BPMN, notación para el modelado de procesos.

El modelado de procesos es la forma idónea para comunicarse entre todos los usuarios de todos los niveles. Constituye la base para todos los posteriores análisis que se pueden realizar para el descubrimiento de problemas y por lo tanto a la mejora de estos. Existen varios lenguajes o notaciones que ayudan a entender y modelar mejor los procesos, como IDEFO, UML, entre otras, pero dentro de todos estos el más integrador es BPMN[1]. Para su construcción sus creadores, los miembros del grupo de trabajo para la notación de BPMI[2], revisaron y analizaron diferentes notaciones existentes tomando de ellas las mejores ideas consolidándolas en una notación estándar. Tiene como objetivo principal servir

[1] Business Process Modeling Notation

[2] Business Process Management Initiative

como soporte para la gestión por procesos, como una notación que pueda ser entendida fácilmente desde los analistas que crean los bocetos iniciales del proceso, los desarrolladores técnicos responsables de implementar la tecnología que ejecutará estos procesos, hasta las personas que los ejecutan y aquellas que llevaran a cabo el monitoreo y supervisión de los procesos. Esta notación crea un enlace entre las etapas de diseño e implementación. (2)

BPMN cuenta con 3 categorías básicas:

- Objetos de flujo.
- Conectores.
- Veredas.

Objetos de flujo:

Los objetos de flujo son un pequeño conjunto de tres elementos, eventos, actividades, bifurcaciones. Los eventos están representados por círculos y se entienden como algo que sucede en el proceso, se dividen en tres tipos de eventos según en el momento en que afecten el flujo, de inicio, intermedios y finales. Las actividades están representadas por cuadrados de esquinas redondeadas. Pueden ser atómicas o no atómicas, es decir, que pueden o no contener, o representar otros procesos, también pueden ser llamadas sub-procesos. Solo se hará uso de las actividades no atómicas, debido a que para representar un sub-proceso, está pensado el uso de eventos intermedios que invoquen a otros procesos. Las bifurcaciones están representadas por un rombo, es el elemento que se encarga de controlar la divergencia o convergencia del flujo.

Conectores:

Los objetos de flujo están conectados entre sí en un diagrama para crear el esqueleto básico de un proceso de negocio. Hay tres objetos de conexión que ofrecen esta función, secuencia de flujo, mensaje de flujo y los de asociación. Solo se hará uso de los conectores de secuencia de flujo los cuales son representados por una línea con la punta de flecha sólida, y se utilizan para mostrar el orden en que las actividades se van realizando.

Veredas:

Diferentes metodologías utilizan el concepto de veredas (swimlanes) como un mecanismo para organizar actividades en distintas categorías visuales para ilustrar diferentes capacidades funcionales o

responsabilidades. BPMN cuenta con dos tipos de veredas principales que son los grupos (pool) y los carriles (lane). La herramienta solo permitirá el uso de carriles para representar un participante en un proceso. También actuará como un contenedor gráfico para los demás elementos BPMN.

La figura 1 muestra las 3 categorías BPMN.

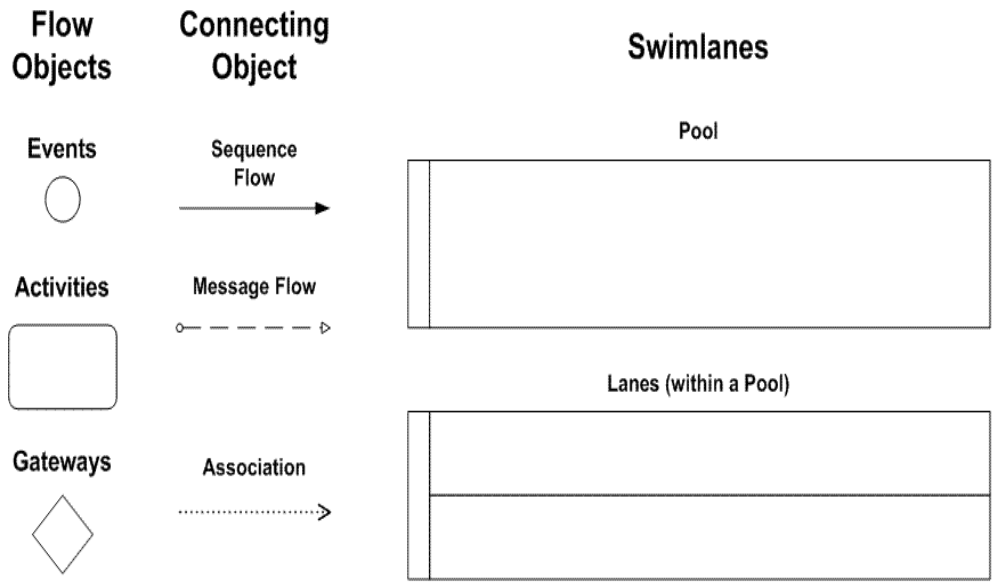

Figura 1: Categorías básicas de BPMN.

1.1.1 Modelos de Procesos.

El modelado de procesos debe ser entendido como dos cuestiones importantes: el modelado y los procesos. Frecuentemente los conjuntos de procesos y subprocesos integrados en una organización, son complejos de entender, amplios y confusos, con múltiples puntos de contacto entre sí y con un buen número de departamentos y puestos implicados, que de una forma u otra forman parte del ciclo de un proceso. Un modelo puede dar la oportunidad de organizar, documentar y mejorar la comprensión sobre un sistema.

Cuando un proceso es modelado, con ayuda de una representación gráfica, pueden apreciarse con facilidad las interrelaciones existentes entre distintas actividades, analizar cada actividad, definir los puntos de contacto con otros procesos, así como identificar los subprocesos comprendidos. Al mismo

tiempo, los problemas existentes pueden ponerse de manifiesto claramente, dando la oportunidad de iniciar acciones de mejora.

Componentes del modelo de procesos.

Entradas: se definen por las necesidades de las personas y las fuentes de información procedentes, tanto internas como externas.

Salidas: constituyen la conclusión del ciclo de vida de la información, posibilitan disponer de productos y servicios de información con valor añadido y deben garantizar la satisfacción de las necesidades de la comunidad de usuarios a la que se vincula el sistema con las exigencias de calidad que ellos demandan o necesitan.

Flujo de información: es el tránsito de la información, desde las entradas por cada uno de los procesos, hasta las salidas. En el paso de la información, desde las entradas a las salidas, intervienen una serie de procesos ordenados que se relacionan estrechamente por medio de diversos flujos, con vista a que el usuario obtenga una nueva información de valor añadido.

1.1.2 Procesos de gestión de la información.

Los procesos de gestión de la información son cada una de las acciones que intervienen en el análisis y utilización de la información que se ha recopilado y registrado para permitir a los usuarios de todos los niveles tomar decisiones. Lo cual implica:

- recoger y analizar la información.
- registrarla y recuperarla cuando sea necesaria.
- socializarla.

Obtener y analizar la información: la información puede conseguirse de informes de técnicos, libros de registro, formularios de los diferentes ejecutantes, reuniones con la comunidad, entrevistas, observación y mapas comunitarios.

Registro de la información: es importante guardar la información para futuras referencias. Puede guardarse en libros de registro locales, informes de progreso o formularios. El principio más importante del registro de informaciones es la facilidad con la que pueden recuperarse.

Divulgación o flujo de información: para que la información tenga un uso adecuado tiene que compartirse con los demás interesados o usuarios. Esta información puede ayudarles en sus decisiones de gestión y también puede ayudar al que la recoge a encontrar significados o usos relacionados con la gestión. (3)

1.2 Modeladores de Procesos desarrollados.

Cuecent BPMN:

Entorno de desarrollo para modelado de procesos amigable para el usuario, basado en estándares y disponible libremente. El modelador Cuecent BPMN proporciona una herramienta gráfica para modelar y documentar los procesos de negocio de la organización. Los modelos construidos con este modelador cumplen con los estándares BPMN y XPDL. Es un software de tipo freeware, lo que significa que está disponible para ser usado sin pagar, pero que no puedes acceder a su código, lo que imposibilitaría la realización de cambios, para adaptarlos a condiciones específicas.

BizAgi Process Modeler:

Esta herramienta permite la modelación de procesos utilizando BPMN. Incluye todos los elementos de esta notación, posibilitando la creación de modelos de procesos de forma fácil. Cuenta además con una herramienta que permite de forma online compartir los modelos realizados, y por consiguiente mejorarlos. El modelador de procesos permite exportar procesos a la Suite BPM de BizAgi, Edición Xpress. Esto le permite convertir sus procesos en aplicaciones ejecutables. Se puede hacer uso de forma gratuita del software por un plazo de tiempo.

Adonis:

Es una herramienta para una gestión integral de procesos de negocio. Posee un gran número de funcionalidades que ayudan a automatizar los procesos de negocio y reducir los costos. El código fuente de esta herramienta no está disponible por lo que se hace imposible modificarla, para que sea capaz de modelar procesos de gestión de la información. Cuenta con ADONIS Process Portal, una herramienta que puede integrase a la intranet de la empresa donde sea implantado, permitiendo la interacción entre todos los niveles administrativos de la misma, compartiendo los modelos de procesos, garantizando que surjan las mejoras.

Estas herramientas facilitan la modelación de los procesos y la ejecución de los mismos, las cuales cuentan con su propio motor de procesos, que no puede ser descargado libremente. Además, incluyen muchos elementos de modelación de la notación BPMN, que son prescindibles para la modelación de los procesos de gestión de la información. El no permitir tener acceso al código de estas aplicaciones imposibilita lograr integrarlas con otras herramientas.

1.3 Metodologías de desarrollo.

En la actualidad a los desarrolladores de grandes aplicaciones se les dificulta realizar una planificación organizada del trabajo, mantener las políticas del trabajo en equipo y a la vez asumir los adelantos en la informatización de los procesos productivos. Para contribuir con estas necesidades la comunidad desarrolladora de software ha estandarizado un conjunto de procedimientos ya definidos con el fin de complementar una metodología de trabajo a seguir para el desarrollo de productos software.

1.3.1 Open Unified Process (OpenUP).

OpenUP como metodología de desarrollo es conducida por el principio de colaboración para alinear intereses y para compartir su comprensión. Este proceso de desarrollo de software es extensible ya que en el proceso se puede agregar o adaptar según lo requiera el sistema. Es una metodología ágil que proporciona una comprensión detallada del proyecto, beneficiando a los clientes y desarrolladores. Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo.

Beneficios del uso de OpenUP:

Es apropiada para proyectos pequeños y de bajos recursos permitiendo disminuir las probabilidades de fracaso en los proyectos e incrementar las de éxito. Permite detectar errores tempranos a través de un ciclo iterativo. Evita la elaboración de documentación, diagramas e iteraciones innecesarias requeridos en la metodología RUP.

1.3.2 Extreme Programming (XP).

Metodología más destacada dentro de los procesos ágiles de desarrollo de software, utilizada para el desarrollo de proyectos cortos. Se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software. Extreme Programming se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones

implementadas. Se define como especialmente adecuada para proyectos con requisitos imprecisos, muy cambiantes y donde existe un alto riesgo técnico.

La metodología se basa en las pruebas realizadas a los principales procesos, haciendo pruebas de las fallas que pudieran ocurrir, además de la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

1.3.3 Rational Unified Process (RUP).

El Proceso Unificado de Desarrollo (RUP) es el proceso más apropiado para el desarrollo de grandes proyectos, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. RUP es un proceso bien definido, estructurado y adaptable específicamente a cada proyecto. Se define a partir de tres características esenciales: estar dirigido por los casos de uso, centrado en la arquitectura y ser iterativo e incremental.

En RUP los casos de uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba, constituyendo un elemento integrador y una guía del trabajo.

Existe una interacción entre los casos de uso y la arquitectura. Los casos de uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como casos de uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

Este proceso de desarrollo está definido por 4 fases fundamentales:

- **Inicio:** fase donde se determina la visión del proyecto.
- **Elaboración:** fase con el objetivo de determinar la arquitectura óptima.
- **Construcción:** fase donde se obtiene la capacidad operacional inicial.
- **Transición:** el objetivo es llegar a obtener la liberación del proyecto.

La selección de una metodología de desarrollo para lograr un producto con mayor calidad, es una de las dificultades que presentan los desarrolladores de software actualmente. El Proceso Unificado de Software (RUP) y Programación Extrema (XP) son los procesos más utilizados y mejor documentados a nivel mundial y OpenUP es la metodología más utilizada por el polo de Bioinformática de la facultad. En la tabla

1 se establece una comparación entre estas tres metodologías para determinar cuál de ellas es la más indicada para el desarrollo de la aplicación.

	Proceso Unificado de Software(RUP)	**Programación Extrema(XP)**	**OpenUP**
Tamaño de equipo	-Está pensado para grandes proyectos y equipos de trabajo en cuanto a tamaño y duración del software.	-Se implementan mejor para proyectos cortos y equipos de trabajo más pequeños.	-Apropiado para proyectos pequeños y de bajos recursos, con un equipo de desarrollo pequeño.
Carga de trabajo	-Es un proceso pesado, basado en mucha documentación en la que no son aceptables todos los cambios. -Existen diferentes elementos de planificación con los que se controla el desarrollo de software. -Define que artefactos son necesarios para	-Es un proceso ligero que no presenta demasiadas tareas organizativas sobre los desarrolladores. -En el desarrollo de este proceso es más importante la entrega al cliente del software que necesita que las funcionalidades que quedan por implementar. -Los cambios se acuerdan directamente con el representante del cliente, con los cuales se ajusta el plan de iteraciones y el de	-Colaborar para sincronizar intereses y compartir conocimiento. Este principio promueve prácticas que impulsan un ambiente de equipo saludable, facilitan la colaboración y desarrollan un conocimiento compartido del proyecto. -Equilibrar las prioridades para maximizar el beneficio obtenido por los interesados en el proyecto. Este principio promueve prácticas que permiten a los participantes de los proyectos desarrollar una solución que maximice los beneficios obtenidos por los participantes y que cumple con los requisitos y restricciones del proyecto. -Centrarse en la arquitectura de forma temprana para minimizar el riesgo y

	poder realizar una actividad y cuales se deberán crear.	liberación y se toman nuevas direcciones en el desarrollo.	organizar el desarrollo. -Desarrollo evolutivo para obtener retroalimentación y mejoramiento continuo. Este principio promueve prácticas que permiten a los equipos de desarrollo obtener retroalimentación temprana y continua de los participantes del proyecto, permitiendo demostrarles incrementos progresivos en la funcionalidad.
Relación con el cliente	-Se presentarán al cliente los artefactos al final de una fase y se valorarán las precondiciones para la siguiente. Solo después que el cliente acepte los artefactos generados se pasará a las siguientes fases.	-El cliente recibe después de cada iteración una parte funcional del programa, estando continuamente informado del estado del proyecto y así poder intervenir en caso de que no cumpla con sus necesidades.	-OpenUP procura un equilibrio entre las necesidades de los involucrados con los resultados del proyecto y los costos técnicos, con el fin de maximizar el valor de los involucrados y las guías del proceso de desarrollo. Desarrolla un ciclo de vida interactivo que mitiga el riesgo a tiempo y ofrece demostrar resultados en curso al cliente del proyecto.

Tabla 1: Comparación entre RUP, XP y OpenUP.

Teniendo en cuenta que RUP es un proceso de desarrollo de software que se caracteriza por su empleo en proyectos de larga duración y con un alto grado de complejidad, ha sido seleccionado para el desarrollo de este proyecto como la metodología de trabajo a utilizar. Para la elección de esta metodología no solo se han tenido en cuenta las ventajas que brinda, sino que también es la metodología de trabajo más utilizada en la institución a la cual pertenece el equipo de desarrolladores.

1.3.4 Roles y Artefactos.

Analista: es el responsable del conjunto de requisitos que están modelados en los casos de uso específicos. Es su responsabilidad también delimitar el sistema, encontrando los actores, los casos de uso, asegurando que el modelo de casos de uso es completo y consistente.

Los artefactos que serán producidos por el analista serán los siguientes:

- **Modelo de Caso de Uso:** se utiliza como acuerdo entre el cliente y desarrolladores, proporciona la entrada fundamental para el análisis, el diseño y las pruebas.
- **Glosario de Términos:** define los términos más importantes que se usan en el proyecto. Es muy útil para alcanzar un consenso entre los desarrolladores relativo a la definición de diversos conceptos y para reducir el riesgo de confusiones.
- **Actor del Sistema:** no es más que el entorno externo del sistema. En este artefacto quedan representados mediante uno o más actores cada uno de los tipos de usuarios y de los sistemas externos que interactúan con el sistema.
- **Caso de Uso del Sistema:** es el artefacto que representa cada forma en que los actores usan el sistema.
- **Vista de Caso de Uso:** incluye los casos de uso que describan alguna funcionalidad importante y crítica, es decir, describe los casos de uso significativos para la arquitectura.

Diseñador de Sistema: es el responsable del diseño de una parte del sistema, el cual contiene los requisitos, la arquitectura y el desarrollo de proceso para el proyecto. Además, identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño.

A continuación se describen los artefactos que serán producidos por el diseñador en el proyecto:

- **Clase del Diseño:** se define como la descripción de un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones, atributos y semántica.
- **Realización de caso de uso:** describe cómo un caso de uso particular es realizado dentro del modelo de diseño, en términos de colaboración de objetos.
- **Subsistema de Diseño:** encapsula el comportamiento, proporcionando interfaces explícitas y formales, sin exponer su contenido interno. Esto proporciona la capacidad de poder encapsular las interacciones de un número de clases y/o subsistemas. Representa una forma de organizar los artefactos del modelo de diseño en piezas más manejables.

Arquitecto de Software: el arquitecto de software es responsable de la arquitectura del software incluyendo las decisiones técnicas que restringen el diseño e implementación general del proyecto. Debe identificar y documentar los aspectos arquitectónicamente significativos de las vistas de requerimientos, diseño, implementación y despliegue.

A continuación se describe el artefacto que será producido por el arquitecto de software:

- **Diagrama de despliegue:** define cómo se distribuye físicamente un sistema. Está representado por nodos procesadores o nodos dispositivos unidos mediante conexiones de comunicación, generalmente mediante protocolos de comunicación TCP/IP, HTTP o HTTPS.

Diseñador de base de datos: es el responsable de diseñar el almacenamiento de la información persistente que se usará en el proyecto. Debe definir el diseño detallado de la base de datos incluyendo tablas, índices, vistas, restricciones, procedimientos almacenados y cualquier otro elemento que se necesite para almacenar, recuperar y eliminar objetos persistentes.

Esta información se recoge en el artefacto Modelo de datos:

- **Modelo de datos:** describe la representación lógica y física de los datos persistentes usados por la aplicación. Puede ser inicialmente creado a través de ingeniería inversa de un almacenamiento de datos persistentes que ya exista (base de datos) o puede ser inicialmente creado a partir de un conjunto de clases del diseño persistentes en el modelo de diseño.

Implementador: es el responsable de desarrollar y probar componentes, en acuerdo con las normas adoptadas en el proyecto para la integración de subsistemas más grandes.

El artefacto que realizará el implementador se describe a continuación:

- **Diagrama de componentes:** son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando: los subsistemas de implementación y sus dependencias a la hora de importar código y organizar los subsistemas de implementación en capas.

1.4 Herramientas y tecnologías de desarrollo.

1.4.1 Herramientas CASE.

Las herramientas CASE (Computer Aided Software Engineering) están destinadas a aumentar la productividad en el desarrollo de software reduciendo tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, documentación o detección de errores entre otras. (4)

Visual Paradigm.

Visual Paradigm es una herramienta CASE que soporta UML como lenguaje de modelado, ofreciendo una solución completa para el desarrollo del software. El software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, a un menor costo. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y documentación. La herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos.

Se caracteriza por lo siguiente:

1. Visual Paradigm es una herramienta case que soporta las últimas versiones del UML (Lenguaje de Modelado Unificado), la notación y modelado de procesos de negocios. Desde un grupo administrador de objetos.
2. En adición al soporte de modelado UML esta herramienta provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java, .NET y PHP.

- Permite la integración con varias Java.

Además de lo anteriormente expuesto cabe destacar también que el equipo de desarrollo cuenta con una mayor experiencia con esta herramienta, que con otras existentes. Lo antes expuesto facilitará la construcción de algunos artefactos en el ciclo de vida del desarrollo de la herramienta de modelación como lo plantea RUP, en un plazo mas corto de tiempo.

1.4.2 Sistemas Gestores de Bases de Datos (SGBD).

Los sistemas gestores de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de

definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. Estos permiten manejar con facilidad grandes volúmenes de información en muy poco tiempo y ofrecen independencia y seguridad en el tratamiento de información. (5)

Existen varios objetivos que deben cumplir los SGBD, dentro de los cuales se consideran más importantes los siguientes:

- Seguridad: los SGBD deben disponer de un sistema de permisos a usuarios y grupos de usuarios, que permitan restringir los accesos no autorizados.
- Integridad: se debe proteger la información almacenada ante fallos de hardware, datos introducidos por descuido o cualquier situación que pueda alterar la integridad de los datos almacenados.
- Tiempo de respuesta: se debe minimizar el tiempo que el SGBD tarda en proporcionar la información solicitada y en guardar los cambios efectuados.
- Abstracción de la información: los usuarios deben ser ajenos a detalles sobre la localización física donde han sido almacenados los datos.

PostgresSQL.

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento. Está disponible en casi cualquier sistema Unix (34 plataformas en la última versión estable), y Windows.

Usa una estrategia de almacenamiento de filas llamada MVCC[3] para conseguir una mejor respuesta en ambientes de grandes volúmenes, característica esta de mucho peso si se tiene en cuenta que los datos que se generan los centros biotecnológicos crecen exponencialmente.

PostgreSQL es Open Source, y este publicado bajo la licencia BSD. Esta licencia tiene menos restricciones en comparación a otras, como la licencia GPL, permitiendo el uso del código fuente en software no libre.

[3] **MVCC:** estrategia de almacenamiento encargada de mantener copias sobre los datos de forma paralela, para acelerar el sistema de escritura de datos a disco duro, haciendo un control de concurrencia entre las distintas versiones que se van escribiendo.

Posee las características de los más potentes sistemas comerciales como Oracle o SQL Server. Entre ellas se pueden destacar:

- Completo soporte para transacciones: una transacción está formada por un conjunto de acciones de forma que o se ejecutan todas ellas o bien ninguna. Utilizando transacciones se asegura la consistencia de los datos.
- Soporte completo ACID (Atomicity Consistency Isolation Durability).
- Procedimientos almacenados: código ejecutable que se almacena compilado en el servidor. Entre otras cosas, permite optimizar las aplicaciones evitando transferencias innecesarias a través de la red (aplicaciones con parte cliente y parte servidor).
- Triggers: procedimientos almacenados que se lanzan automáticamente bajo determinadas circunstancias como actualizaciones de campos. Permite establecer reglas de integridad y consistencia a nivel del servidor de base de datos.
- Vistas: conjunto de registros resultado de una consulta que se comportan como una tabla física para facilitar su manejo.
- Orientación a objetos con herencia de tablas.

Cabe destacar que la institución a la cual pertenece el equipo de desarrolladores propone el uso del mencionado Sistema Gestor de Bases de Datos (SGBD). Agregar además el incremento del uso de software libre en el país, como vía para lograr una independencia tecnológica, hacen de PostgresSQL la mejor opción.

1.4.3 Ajax.

Ajax, acrónimo de Asynchronous JavaScript And XML, es una técnica de desarrollo web para crear aplicaciones RIA (Rich Internet Applications). Las aplicaciones construidas con Ajax se ejecutan en el navegador mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. (6)

Como principales ventajas de Ajax es importante que se mencione:

- Tráfico mínimo**:** las aplicaciones Ajax deben enviar y recibir una pequeña cantidad de información desde y para el servidor, por lo que se minimiza el tráfico entre el cliente y el servidor, asegurándose que no reciba o envíe información innecesaria.

- Convenciones establecidas**:** no gasta tiempo inventando nuevos modelos de interacción de usuario con los que no estarán además familiarizada.
- Multi-Browser y Multi-Plataforma.

El uso de esta técnica en la creación de la herramienta de modelación de procesos de gestión de la información, garantizará la independencia del tipo de lenguaje de programación que sea usado en el servidor. Por lo que un cambio en el lenguaje de programación no conllevaría a grandes cambios en la vista de la aplicación.

1.4.4 Frameworks de desarrollo.

Symfony.

Un framework es una estructura de soporte definida, que organiza el desarrollo de software. Puede incluir programas, bibliotecas y lenguajes interpretados entre otros, para mejorar el desarrollo y lograr una mejor unificación entre los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones más generales de las entidades del dominio, brindando una estructura y una metodología de trabajo bien definida. (7)

En el caso particular del framework Symfony simplifica el desarrollo de aplicaciones web mediante la automatización de algunos patrones utilizados para resolver tareas. Obliga a los programadores a crear códigos más legibles, fáciles de mantener y facilita la programación de complejas aplicaciones ya que encapsula operaciones con alto grado de complejidad en instrucciones sencillas. (8)

Garantiza una compatibilidad con la mayoría de los gestores de bases de datos más utilizados como: MySQL, PostgresSQL, Oracle y SQL Server, pues provee una abstracción a la base de datos mediante Creole y Propel como ORM[4], permitiéndolo ser independiente del SGBD.

Symfony provee de una capa de internacionalización que permite crear aplicaciones en varios idiomas. La internacionalización está integrada en el framework, siguiendo los estándares XLIFF[5].

La selección de symfony como framework de desarrollo esta basada en que la entidad a la cual pertenece el equipo de desarrollo, propone su uso por las razones anteriormente expuestas. Además, el equipo de

[4] Object-Relational mapping(mapeo objeto-relacional)

[5] XML Localization Interchange File Format

desarrolladores se encuentra familiarizado con el mencionado framework, lo que hace su empleo para el desarrollo de la herramienta, más sencillo.

Dojo Toolkit.

Dojo es una librería open source escrita en Javascript y destinada a facilitar el rápido desarrollo de aplicaciones basadas en Ajax. Su idea es la de abstraer al desarrollador de las complejidades del DHTML (Dynamic HTML) y de las discrepancias existentes entre navegadores, que hacen que el código JavaScript a utilizar sea diferente. Provee gran cantidad de funcionalidades las cuales divide en tres grandes proyectos, Dojo Core, Dijit y DojoX. Dijit y DojoX están basadas en la sólida base que Dojo Core ofrece. (9)

Entre las principales características que brinda Dojo se encuentran:

- Eventos: Dojo generaliza el sistema de eventos existentes en Javascript. Se puede conectar eventos en código y la aplicación resultante funciona en Firefox, Internet Explorer y Safari sin modificación alguna.
- XHR (Ajax): Dojo adiciona una buena fachada para los servicios nativos de xmlHttpRequest, con el uso de una sola función se puede pasar parámetros a una petición vía Ajax y obtener la respuesta en texto plano, en XML o en un objeto Javascript vía JSON.

Está disponible ya sea en virtud de los términos de la licencia BSD modificada o la Licencia Libre Académica versión 2,1. Ambas licencias le otorgan amplios derechos de uso y construir en ambos casos aplicaciones de código abierto.

Aunque actualmente existen varias librerías Javascript que facilitan el trabajo con Ajax y la creación de aplicaciones web semejantes a aplicaciones de escritorio, la selección de Dojo estuvo basada la experiencia de los desarrolladores con la mencionada librería, para lograr disminuir el plazo de tiempo de una primera versión funcional de la herramienta. El análisis realizado arrojó también que el manejo de elementos gráficos, necesarios para la modelación de los procesos, resulta más cómodo con la utilización de Dojo.

1.4.5 IDE de desarrollo. NetBeans.

Un entorno de desarrollo integrado o IDE[6], es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

NetBeans es un entorno de desarrollo que está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso, tanto comercial como no comercial. (10)

Haciendo uso de NetBeans como IDE de desarrollo se aprovechan algunas importantes características, que fomentan el rápido y organizado desarrollo de la aplicación. Cuenta con un número importante de módulos entre los que se encuentra el módulo de desarrollo para aplicaciones web con PHP, el de Symfony que facilita el desarrollo de aplicaciones haciendo uso de este framework y el módulo de subversión, permitiendo desde el mismo entorno manipular las versiones del código durante la implementación.

1.4.6 Servidor Web. Apache.

Un servidor web es un programa que está diseñado para transferir hipertextos, páginas web o páginas HTML[7] : textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. El programa implementa el *protocolo HTTP* [8] que pertenece a la capa de aplicación del modelo OSI.

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1, siendo además flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. (11)

Apache presenta entre otras características que es altamente configurable, ya puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyos, por esta razón es muy sencillo

---

[6] I*ntegrated development environment*

[7] HyperText Markup Language

[8] *HyperText Transfer Protocol*

ampliar sus capacidades. Trabaja con gran cantidad de lenguajes de scripts como Perl, PHP entre otros. Por lo antes expuesto los autores de este trabajo de diploma deciden hacer uso este servidor web, aprovechando sus múltiples ventajas, entre las que se encuentran: ser modular, altamente configurable, multiplataforma y que logra una eficiente integración con php.

1.5 Conclusiones parciales del capítulo 1.

En la culminación de este capítulo y como consecuencia de haber realizado un análisis de las herramientas y tecnologías existentes en el mundo, se decide utilizar la metodología de desarrollo RUP, apoyado en el uso de Visual Paradigm 3.4 como herramienta CASE y UML como lenguaje de modelado. Para garantizar la organización del proyecto, la seguridad de la aplicación y el aprovechar el ORM que trae incluido, se usará el framework Symfony 1.2.8. Por la integración que logra el NetBeans en su versión 6.8 con el framework Symfony, es el IDE de desarrollo seleccionado. Dojo garantizará la creación de una aplicación RIA, que sea más amigable para el usuario final y un menor tráfico entre el navegador y el servidor. Como gestor de base de datos se seleccionó PostgreSQL 8.3 y como servidor de aplicaciones web Apache 2.0.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En este capítulo se realizará la descripción del sistema a desarrollar para lograr una mayor claridad y comprensión por parte de los desarrolladores. Los puntos que se abordarán son los siguientes: descripción del modelo de dominio, especificación de los requisitos funcionales y no funcionales, representación de los actores y casos de uso del sistema a través de un diagrama de casos de uso del sistema, así como la descripción textual de cada uno de ellos, y los patrones de casos de uso a utilizar.

2.1 Objeto de estudio.

Situación problémica.

El desarrollo científico cubano, en particular el biotecnológico, es reconocido en el mundo entre los grandes logros de la Revolución. Cuba puede ser considerada en este campo, como un país del primer mundo, ya que en poco más de dos décadas puede exhibir importantes logros, que pueden competir con las grandes compañías transnacionales.

Para nuestro país el desarrollo científico siempre ha sido prioridad. Por tal razón se han creado varios polos científicos a lo largo de la isla, los cuales cuentan con una gran cantidad de procesos necesarios para la creación y certificación de calidad de los productos que en ellos se realizan.

Los centros dedicados a biotecnología están formados por varios laboratorios, en los cuales se va recopilando información, que se obtiene a través de los modelos de recogida de datos. Actualmente no se cuenta con la posibilidad de seguir y controlar estos procesos, lo que hace que el trabajo llegue a ser hasta cierto punto desorganizado, y repercuta en el tiempo de descubrimiento y comercialización de los fármacos.

Estos procesos pueden sufrir modificaciones, debido a que para competir con éxito en el mercado, se necesita cumplir con mayores requisitos que abalen el producto. Por lo que si un proceso necesita ser modificado, se pierde tiempo en realizar estas transformaciones en la organización. Por tal razón se hace necesario el desarrollo de una herramienta que permita al usuario la construcción de sus propios procesos, que le facilite el control y modificación de los mismos y disminuya el tiempo de descubrimiento y comercialización de los fármacos.

Objeto de automatización.

Diseño y control de los procesos de gestión de la información. El poder crear y modificar los procesos de gestión de la información que se desarrollan en los centros biotecnológicos, no es una tarea para nada sencilla. Pero esta tarea es de vital importancia en estos centros, ya que sin una buena organización en la recogida de datos, los productos finales no cumplirán con las normas o estándares establecidos.

Los usuarios autorizados serán los encargados de realizar la creación y la modelación de cada uno de los procesos. Estos procesos deberán ser activados para que cada departamento o laboratorio pueda iniciarlos.

Propuesta del sistema.

Se propone la implementación de una herramienta web que permitirá la creación, el seguimiento y control de los procesos que se desarrollan en los centros biotecnológicos y además poder realizar la modelación de los mismos de forma gráfica. Esta herramienta contará con una parte encargada de la administración y gestión de los procesos, y otra área para el diseño gráfico de los mismos.

Esto provocará que sea más sencillo gestionar los procesos que se desarrollan en este tipo de centros. Garantizará que en todo momento se cumpla con el orden correcto a la hora de realizar algún tipo de proceso y que toda la organización este trabajando sobre los mismos procesos.

2.2 Modelo de dominio.

Cuando no existen procesos definidos, RUP propone realizar un modelo del dominio, que es un subconjunto del modelo de negocio. Un modelo del dominio recoge los tipos de objetos más importantes que existen en el sistema, y tiene por objetivo ayudar a comprender mejor los conceptos que utilizan los usuarios, con los que se trabaja. La mayoría de los objetos del dominio se obtienen mediante la entrevista con los clientes o expertos del dominio. El modelo de dominio se realiza con un diagrama de clases UML, y debe reflejar las relaciones que se establecen entre ellas y en caso necesario la multiplicidad.

Descripción del modelo de dominio.

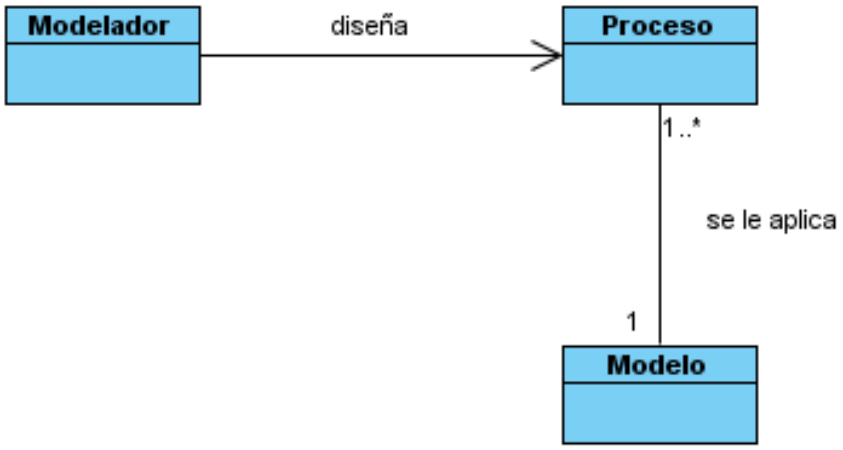

Diagrama 1: Modelo de Dominio

Clases conceptuales del dominio.

1. **Usuario**: se encarga de inicializar los eventos de creación y diseño de los procesos.
2. **Proceso:** describe el recorrido del modelo por la institución en la recogida de datos.
3. **Modelo:** es el documento que se confecciona para la recogida de datos en el centro.

2.3 Reglas del negocio.

Las Reglas del Negocio se basan en la extracción de información originada de las políticas, reglas y regulaciones del negocio y de la descripción del flujo. Es muy importante conocer que el uso de las reglas del negocio no influye en la estructura del modelo de casos de uso.

- Un proceso recién creado su estado será de Edición.
- No se podrá eliminar un proceso que este Activo, solo se podrá hacer si esta en Edición u Obsoleto, este último si el proceso no tiene instancias.
- Si a un proceso se le cambiara su modelado mientras su estado sea Activo, este proceso se duplicará, el nuevo proceso entra en estado de Edición y el proceso que estaba Activo pasa al estado Obsoleto.
- Los carriles, serán los encargados de organizar y categorizar las actividades, además de contener el resto de los objetos.
- Todos los modelos de los procesos contendrán un objeto que será el punto de inicio para la ejecución, así como otro para marcar el fin del proceso.

- Si la versión del modelo al cual se le está creando el proceso, es modificada, el proceso sufrirá cambios. Si su estado era activo, cambiará a obsoleto. En caso de que fuera edición, se eliminará todo lo relacionado con las variables de la versión en el proceso.

2.4 Requerimientos del sistema.

2.4.1 Requisitos funcionales del sistema.

Condición o capacidad que el usuario necesita para resolver un problema o resolver un objetivo. Es la condición o la capacidad que tiene que ser alcanzada o poseída por un sistema o componente de sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. En la herramienta a desarrollar se identificaron los siguientes requisitos funcionales:

RF1 Insertar un proceso.

RF2 Modificar un proceso.

RF3 Eliminar un proceso.

RF4 Buscar y visualizar procesos.

RF5 Insertar objeto.

RF6 Eliminar objeto.

RF7 Modificar propiedades de los objetos.

RF8 Mover objetos.

RF9 Guardar modelado.

2.4.2 Requisitos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente, están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. A continuación se exponen características no funcionales del sistema:

Requerimientos de software.

Sistema Operativo Windows XP o superior, Linux (cualquier distribución). Navegador web Mozilla Firefox 3.0 o Internet Explorer 5 o versiones superiores. Se requiere que el servidor de aplicaciones tenga instalado Apache 2.0 y PHP 5, mientras que el servidor de base de datos cuente con Postgres en su versión 8.3.

Requerimientos de hardware.

Se deberá contar con impresora en las PC clientes o compartidas que interactúen con la aplicación.

Se recomienda para un correcto funcionamiento del sistema y estar acorde con el tiempo de respuesta establecido los siguientes requisitos:

Servidor de aplicaciones:

- procesador Pentium IV a 3.0 GHz.
- 1 GB de RAM.
- 250 GB de espacio libre en el disco duro.

Servidor de Base de Datos:

- procesador Pentium IV a 3.0 GHz.
- 1 GB de RAM.
- 500 GB de espacio libre en el disco duro.

PC cliente:

- procesador Pentium IV.
- 256 MB de RAM.

Requerimientos de apariencia o interfaz externa.

La aplicación contará con una interfaz amigable con páginas no cargadas de mucha información, colores suaves, lo más cercano a una aplicación de escritorio.

Requerimientos de seguridad.

- **Confidencialidad:** existencia de distintos roles que establezcan que la información solo sea vista por aquellos usuarios que posean los privilegios suficientes; restringir la ejecución de acciones a usuarios sin credenciales que intenten acceder a las mismas.

- **Integridad:** validación de los datos en el servidor para evitar estados inconsistentes. La información manejada por el sistema estará protegida del acceso y divulgación no autorizada. Se debe realizar la confirmación sobre acciones irreversibles como eliminaciones.
- **Disponibilidad:** el sistema estará disponible las 24 horas del día a los usuarios autorizados, garantizando el acceso a la información en cualquier momento. Los mecanismos utilizados para lograr la seguridad no obstruyen el acceso a la información.

Estos tres principios de la seguridad informática serán garantizados por el módulo de gestión usuarios y el pluggin sfDojoPlugin, este pluggin permite la autenticación de los usuarios y la creación de sus credenciales.

Requerimientos de usabilidad.

La aplicación podrá ser utilizada por personal vinculado a la Biotecnología, que tengan conocimientos básicos de computación y de aplicaciones Web.

Restricciones del diseño y la implementación.

Para el diseño y documentación de la aplicación se utiliza la metodología RUP, usando el lenguaje de modelado UML 6.4. La Arquitectura está basada en el patrón Modelo Vista Controlador, utilizando para el desarrollo Visual Paradigm 3.4 y NetBeans 6.8. Los lenguajes a utilizar serán JavaScript y PHP, como estándares de codificación PHPCase y CamelCase. Utilizando las bibliotecas Symfony, DojoToolKit y patTools.

Requerimientos de soporte.

Garantía de instalación y prueba del sistema, además de un breve entrenamiento a los futuros usuarios. Se le dará asistencia técnica en un período de 6 meses. Así como pruebas de estrés durante el primer año de explotación.

Requerimientos de Persistencia.

La información del sistema debe almacenarse en bases de datos con carácter imborrable con el objetivo de poder realizar análisis de la misma en cualquier momento durante el paso de los años.

Rendimiento.

Reducción de los tiempos de respuestas y alta velocidad de procesamiento de la información a través de peticiones asincrónicas al servidor. Los tiempos de respuestas deben ser los más cortos posibles aproximadamente de 3 a 5 segundos, al igual que la velocidad de procesamiento de la información.

Portabilidad.

Será un sistema multiplataforma, se podrá disponer del mismo tanto en el sistema operativo Linux como Windows.

Legales.

El sistema será registrado en el Centro Nacional de Derecho de Autor a través de la Dirección de Servicios Legales de la UCI. Se estará usando para el desarrollo de la aplicación herramientas de software libre con licencia GNU/GPL.

2.5 Modelo de casos de uso del sistema.

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Un caso de uso representa la interacción entre los tipos de usuarios, representados por uno o varios actores, los que pueden representar un individuo o un sistema externo, y el sistema.

Se identificó como único actor del sistema al modelador.

Actor	Descripción
Modelador	Es la persona que se encarga del modelado de los procesos de gestión de la información que se llevan a cabo en la empresa.

2.5.1 Diagrama de casos de uso del sistema.

El patrón utilizado en el desarrollo de la herramienta de modelación, es el patrón CRUD (Create, Read, Update y Delete). Este patrón propone identificar un caso de uso que modela las operaciones de crear, leer, actualizar y eliminar, sobre una entidad. Quedando agrupados los 9 requisitos funcionales en los siguientes casos de uso.

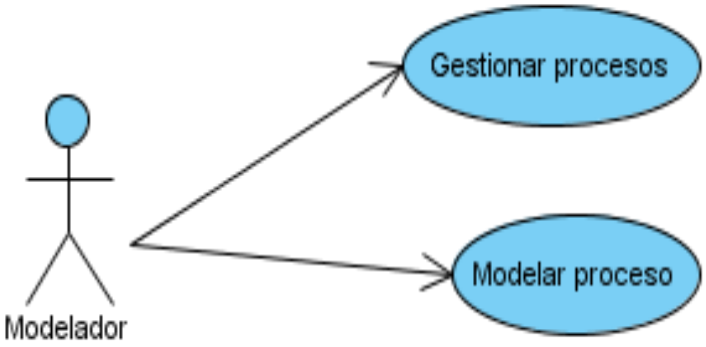

Diagrama 2: Diagrama de Casos de Uso del Sistema

2.5.2 Descripción textual de los casos de uso del sistema.

Caso de Uso:	**Gestionar Procesos**
Actores:	Modelador (inicia)
Resumen:	El modelador cuenta con la posibilidad de realizar diferentes acciones con los procesos, estas acciones pueden ser: insertar un nuevo proceso a la empresa, modificar uno existente, eliminar un proceso que este obsoleto, podrá buscar un proceso o varios según el criterio de búsqueda y cambiar el estado de un proceso existente.
Precondiciones:	El modelador debe de estar previamente autentificado en el sistema.
Referencias	RF1, RF2, RF3, RF4
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Inicia el caso de uso cuando el modelador decide ejecutar una de las siguientes acciones: • Insertar un proceso. • Eliminar un proceso • Modificar un proceso.	2. El sistema en dependencia de la opción que seleccione el modelador realiza las siguientes operaciones : • Si el modelador selecciona la opción de insertar un nuevo proceso ir a la sección “Insertar procesos”. • Si el modelador selecciona la opción de eliminar un proceso ir a la sección “Eliminar procesos”. • Si el modelador selecciona la opción de modificar un

• Buscar un proceso.	proceso ir a la sección “Modificar procesos”. • Si el modelador selecciona la opción de buscar un proceso ir a la sección “Buscar procesos”.
Sección 1”Insertar procesos”	
Acción del Actor	**Respuesta del Sistema**
	1- Se despliega un diálogo en donde se muestran los datos que debe tener un proceso, nombre, modelo y descripción.
2- El modelador ingresa los datos requeridos para la creación de un nuevo proceso, y presiona el botón crear.	3- El sistema verifica que el nombre del proceso no exista perteneciendo a otro proceso que referencie al mismo modelo, si no existe, inserta el nuevo proceso al sistema, y refresca el área donde se muestran los procesos, con el nuevo proceso creado. Finalizando así el caso uso.
Flujo Alterno 1 “Inserta procesos”	
2.1- El modelador ingresa los datos requeridos o no para la creación del nuevo proceso, y presiona el botón cancelar.	2.2- El sistema cierra el diálogo y no realiza ninguna acción, concluyendo la sección.
Flujo Alterno 2 “Inserta procesos”	
	3.2- Si existe, muestra un mensaje indicando al modelador que ya existe un proceso con el mismo nombre que referencia al mismo modelo, concluyendo así la sección.
Sección 2 “ Eliminar procesos”	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema verifica que exista algún proceso seleccionado.
	2- El sistema pide confirmación de la acción a realizar.
3- El modelador confirma que desea eliminar el proceso seleccionado.	4- El sistema verifica el estado del proceso, si está en edición lo elimina y actualiza el área donde se muestran los procesos, y el proceso es eliminado. Finalizando así el caso de uso.
Flujo Alterno 1 “Eliminar procesos”	
	1.1- Si no hay procesos seleccionados, el sistema muestra un mensaje indicando que debe seleccionar un proceso antes.

Flujo Alterno 2 "Eliminar procesos"	
3.1- El modelador confirma que no desea eliminar el proceso.	3.2- El sistema no realiza la acción de eliminar el proceso, concluyendo así la sección.
Flujo Alterno 3 "Eliminar procesos"	
	4.1- El sistema verifica el estado del proceso, si no está en edición, muestra un mensaje informando al usuario que no puede realizar esa acción debido a que el estado del proceso no es de "Edición", concluyendo así la sección.
Sección 3 "Modificar procesos"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema verifica que exista algún proceso seleccionado.
	2- El sistema verifica que el estado del proceso sea distinto de "activo" y despliega un diálogo en donde se muestran los datos que pueden ser modificados por el modelador, el nombre, la descripción del proceso y el estado.
3- El modelador introduce los nuevos datos y pulsa el botón modificar.	4- El sistema valida los datos introducidos.
	5- Si los datos introducidos son correctos se actualizará el área de trabajo con los nuevos datos del proceso modificado. Finalizando así el caso de uso.
Flujo alterno 1 "Modificar procesos"	
	1.1- Si no existe algún proceso seleccionado, el sistema muestra un mensaje.
Flujo alterno 2 "Modificar procesos"	
	2.1- Si el proceso seleccionado se encuentra en el estado de "activo", el sistema muestra un mensaje de alerta al modelador.
Flujo alterno 3 "Modificar procesos"	
3.1- El modelador introduce los datos y presiona el botón cancelar.	3.2- El sistema cancela la operación y deja el proceso con los datos anteriores, concluyendo así la sección.

Flujo alterno 4 “Modificar procesos”	
	5.1- Si los datos introducidos son incorrectos, el sistema mostrará un mensaje indicando que campo es el incorrecto.
Sección 4 “Buscar procesos”	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los siguientes campos. • Variable (código, nombre, descripción, fecha de creación) • Criterio. • Valor.
2. El modelador selecciona la variable por la que va a realizar la búsqueda y adiciona el criterio.	3. El sistema agrega el nuevo criterio de búsqueda y da la opción de introducir el valor a buscar.
4. El modelador introduce el o los datos para el criterio o criterios de búsqueda, y presiona el botón buscar.	5. El sistema valida el campo valor en dependencia de la variable seleccionada.
	6. El sistema refresca el área donde se muestran los procesos, con los procesos que coincidan con el criterio o criterios de búsqueda establecidos por el modelador. Finalizando así el caso de uso.
Flujo alterno 1 “Buscar procesos”	
4.1- El modelador introduce el o los datos para el criterio o criterios de búsqueda, y presiona el botón cancelar.	4.2- El sistema cierra el diálogo y no realiza ninguna búsqueda, concluyendo así la sección.
Poscondiciones	Según la acción del actor se insertó, eliminó, modificó o buscó un proceso.

Caso de Uso:	**Modelar proceso**

<table>
<tr><td>Actores:</td><td>Modelador (inicia)</td></tr>
<tr><td>Resumen:</td><td>El modelador cuenta con la posibilidad de realizar diferentes acciones con los elementos gráficos cuando este modelando un proceso, como pueden ser: adicionar nuevos objetos al área de diseño, eliminar objetos existentes en el área de diseño, modificar las propiedades o moverlos de posición.</td></tr>
<tr><td>Precondiciones:</td><td>Deben de existir procesos en el sistema los cuales se puedan modelar.
Debe existir un modelo de recogida de datos para que el proceso pueda ser modelado.</td></tr>
<tr><td>Referencias</td><td>RF5, RF6, RF7, RF8, RF9</td></tr>
<tr><td>Prioridad</td><td>Crítico</td></tr>
<tr><td colspan="2">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1. Inicia el caso de uso cuando el modelador decide ejecutar una de las siguientes acciones:
• Insertar un objeto.
• Eliminar un objeto.
• Modificar las propiedades de un objeto.
• Mover un objeto.
• Guardar el modelado.</td><td>2. El sistema en dependencia de la opción que seleccione el modelador realiza las siguientes operaciones :
• Si el modelador selecciona la opción de insertar un nuevo objeto ir a la sección “Insertar objetos”.
• Si el modelador selecciona la opción de eliminar un objeto ir a la sección “Eliminar objetos”.
• Si el modelador selecciona la opción de modificar las propiedades de un objeto ir a la sección “Modificar propiedades de los objetos”.
• Si el modelador selecciona la opción de mover un objeto ir a la sección “Mover Objetos”.
• Si el modelador selecciona la opción de guardar el modelado, ir a la sección “Guardar modelado”.</td></tr>
<tr><td colspan="2">Sección 1 ”Insertar objetos”</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1- El modelador selecciona el objeto a insertar.</td><td></td></tr>
<tr><td>2- El modelador da clic en el área</td><td>3- El sistema verifica que la posición en que la que se desea</td></tr>
</table>

de diseño en la que lo quiere colocar.	colocar el objeto seleccionado, sea correcta.
	4- Si la posición en la que se desea insertar el objeto es correcta, se muestra el objeto en la posición seleccionada por el modelador.
Flujos Alternos 1 "Insertar objetos"	
	4.1- Si la posición en la que se desea insertar el objeto no es correcta, el objeto no aparecerá.
Sección 2 " Eliminar objetos"	
Acción del Actor	**Respuesta del Sistema**
1- El modelador marca el objeto que desea eliminar.	2- El sistema bordea el objeto seleccionado con otro color.
3- El modelador presiona el botón eliminar.	4- El sistema eliminará el objeto seleccionado, en caso de que sea un objeto que contenga a otros, todos los objetos contenidos en él, se eliminarán igualmente.
Sección 3 "Modificar propiedades de los objetos"	
Acción del Actor	**Respuesta del Sistema**
1- El modelador selecciona el objeto.	2- El sistema bordea el objeto seleccionado con otro color.
	3- El sistema muestra todas las propiedades del objeto en una paleta de propiedades.
4- El modelador cambia los valores de las propiedades que tiene el objeto seleccionado.	5- El sistema valida las nuevas propiedades, y si son correctas actualiza el área de diseño mostrando al objeto con las nuevas propiedades.
Flujo alterno 1 "Modificar propiedades de los objetos"	
	5.1- Si las propiedades introducidas no son correctas, el sistema muestra un mensaje indicando que los datos no son válidos, y no realizará ningún cambio.
Sección 4 "Mover objetos"	
Acción del Actor	**Respuesta del Sistema**
1- El modelador selecciona el objeto que desea mover de	2- El sistema bordea el objeto seleccionado con otro color.

posición.	
3- El modelador arrastra el objeto seleccionado hasta la posición deseada.	4- El sistema dibuja nuevamente el objeto en la posición actual. Si es un objeto que contiene objetos, todos los objetos se moverán también.
Sección 5 "Guardar modelador"	
1- El modelador oprime el botón guardar, para salvar los cambios que ha hecho en el modelado del proceso.	2- El sistema verificara que el proceso al que se desea modelar exista.
	3- El sistema verifica que el diseño no sea nulo.
	4- El sistema guardará el diseño.
Flujo alterno 1 "Guardar modelador"	
	2.1- Si no existe el proceso el sistema se lo informará al modelador, y saldrá del área de diseño.
Flujo alterno 2 "Guardar modelador"	
	3.1- En caso de que el diseño este vació el sistema emitirá un mensaje de error, y no guardará el diseño, dando la oportunidad de volver a realizar el diseño.
Poscondiciones	Se insertó, eliminó, modificó las propiedades de los objetos, se movió el objeto y se guardó un diseño.

2.6 Conclusiones parciales capítulo 2.

En el desarrollo de este capítulo se le dio cumplimiento al primer objetivo específico planteado, identificándose 9 funcionalidades necesarias para el diseño de la herramienta de modelación de procesos de gestión de la información. Estas se encuentran agrupadas en 2 casos de uso arquitectónicamente significativos. Además de identificar los requerimientos no funcionales del sistema; modelar el diagrama de casos de uso del sistema y describir los casos de uso, para obtener así un mayor entendimiento del problema a resolver.

CAPÍTULO 3: DISEÑO DEL SISTEMA

En este capítulo se realiza el diagrama de clases del diseño con la pertinente descripción de los paquetes que lo conforman, incluyendo los estilos y los patrones de diseño seleccionados, además de explicar cómo se adaptan al diseño del sistema. También en dicho capítulo se encuentran presentes el diagrama de despliegue, el diagrama de clases persistentes y el modelo de datos relacional de la base de datos en el cual se almacena toda la información que se procesa y finalmente se describen las tablas que componen este último.

3.1 Arquitectura del sistema.

La arquitectura de software puede ser definida como una vista estructural de alto nivel, que ocurre en los comienzos del ciclo de vida del software y define el estilo o grupos de estilos adecuados para dar cumplimiento a los requisitos funcionales. Está compuesta por 4+1 vistas y entre ellas se encuentran la vista lógica y de despliegue.

Las clases del diseño más importantes agrupadas por paquetes, se encuentran en la vista lógica. Es representada por uno o varios diagramas de clases que son un subconjunto del modelo de diseño.

La vista de despliegue ayuda a comprender mejor la distribución física del sistema a través de nodos.

3.1.1 Patrón de arquitectura. Modelo Vista Controlador.

Buschmann define un patrón arquitectónico como: aspectos fundamentales de la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes.

Los patrones de arquitectura ofrecen el esquema fundamental de organización para sistemas de software, incluyendo reglas y guías para organizar las relaciones entre ellos.

Patrón Modelo Vista Controlador.

Para la creación de la herramienta de modelación de procesos de gestión de la información, se hará uso del patrón Modelo Vista Controlador (MVC Model View Controller) por sus siglas en inglés. Básicamente consiste en estructurar el programa en tres componentes que se relacionan entre ellos de una manera muy concreta:

- El modelo: es la representación de la información que maneja la aplicación, es decir, su lógica de negocio.
- La vista: es la representación gráfica del modelo para la interacción con el usuario, en el caso de una página web, transforma el modelo en una página HTML que permite al usuario interactuar con ella.
- El controlador: se encarga de manejar y responder las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista, de ser necesario.

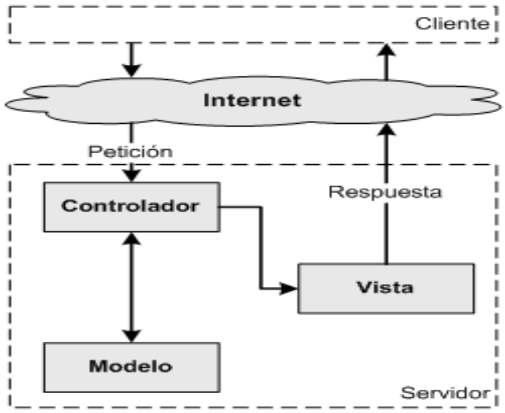

Figura 2: Patrón Modelo Vista Controlador.

El ciclo de vida de este patrón consta de tres etapas. La primera, es la que inicia el usuario cuando realiza una petición a la aplicación, el controlador recibe la solicitud hecha por el cliente y decide a quién debe delegar esa tarea.

El modelo es el que se encarga de realizar las operaciones sobre la información que maneja para cumplir con las solicitudes del controlador. Una vez terminada su labor le regresa al controlador la información resultante de sus operaciones, y este las muestra en las vistas. Entre las ventajas que ofrece la aplicación de este patrón arquitectónico se encuentran las siguientes:

- Mantenimiento más sencillo de las aplicaciones, pues si se requiere hacerse un cambio, por ejemplo en la vista, no hay necesidad de cambiar las acciones, o el modelo.
- Es mucho más sencillo agregar múltiples representaciones a los mismos datos, debido a que un mismo modelo puede contener más de una vista.

La implementación del MVC de symfony.

Symfony toma lo mejor de la arquitectura MVC y lo implementa de forma tal que la creación de aplicaciones sea lo más sencilla posible. El framework Symfony crea un controlador frontal y un layout comunes para cada aplicación. El controlador frontal es el único punto de entrada de la aplicación, es el encargado de cargar la configuración y determina la acción que debe ejecutarse.

Las clases necesarias para acceder al modelo de datos de cada aplicación son generadas por el framework también, por defecto la librería que se encarga de esta tarea es la Propel, aunque pueden utilizarse otras, como Doctrine. Symfony garantiza una abstracción a la base de datos, pues se realiza de forma nativa mediante PDO[9], esto posibilita que si se cambia el sistema gestor de bases de datos, no es necesario cambiar ni una sola línea de código. Para la parte de la vista, el framework Symfony cuenta con plantillas para cada una de las acciones, además de componentes y slots, los que facilitan el desarrollo de una vista agradable y por pequeños pedazos, mejorando el mantenimiento; además de la corrección de errores.

3.1.2 Patrones de diseño.

Un patrón es una unidad de información nombrada, instructiva e intuitiva que agrupa exitosas soluciones probadas a un problema recurrente dentro de un cierto contexto. El objetivo de los patrones es crear un lenguaje común para comunicar experiencia sobre los problemas y sus soluciones. Muchos autores han dado conceptos acerca de que es un patrón. (12)

Un buen patrón debe contar con las siguientes partes:

- Nombre del patrón.
- Clasificación del patrón.
- Intención.
- También conocido como.
- Implementación.
- Código de ejemplo.
- Usos conocidos.

[9] PHP Data Objects

- Patrones relacionados.

Según Grady Booch "Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles"

En la creación de la herramienta de modelación de procesos de gestión de la información se hizo uso de los siguientes patrones:

Creador.

El patrón Creador pertenece al conjunto de patrones GRASP. Lo que define este patrón es que una instancia de un objeto la tiene que crear el objeto que tiene la información para ello. Los programas orientados a objetos crean varias instancias de objetos, es por ello, que esta es una tarea que no se puede pasar por alto.

El empleo del mencionado patrón se evidencia cuando se crean objetos de las clases del modelo. El siguiente fragmento de código muestra la utilización de este patrón, donde es la clase **Userinfo** la encargada de crear la instancia de **GroupMembers**.

```
9   public function save( PropelPDO $con = null ){
10
11      parent::save($con);
12      if( !sfContext::getInstance()->getUser()->hasCredential(array('Administrators','Designers'),false) ) {
13          $gm = new GroupMembers();
14          $gm->setUserName( $this->getUsername() );
15
16          if( sfContext::getInstance()->getUser()->hasCredential('ProcessDesignersLeader') ) {
17              $gm->setGroupName('ProcessDesigners');
18              $gm->save();
19          }else if( sfContext::getInstance()->getUser()->hasCredential('ModelDesignersLeader') ) {
20              $gm->setGroupName('ModelDesigners');
21              $gm->save();
22          }else if( sfContext::getInstance()->getUser()->hasCredential('ReportDesignersLeader') ) {
23              $gm->setGroupName('ReportDesigners');
24              $gm->save();
```

Figura 3: Código ejemplo del uso del patrón Creador.

Alta cohesión.

La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada uno debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

La clase actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones e instanciar objetos, entre otras. Proporciona diferentes funcionalidades que se encuentran estrechamente relacionadas, garantizando que el software sea flexible frente a grandes cambios.

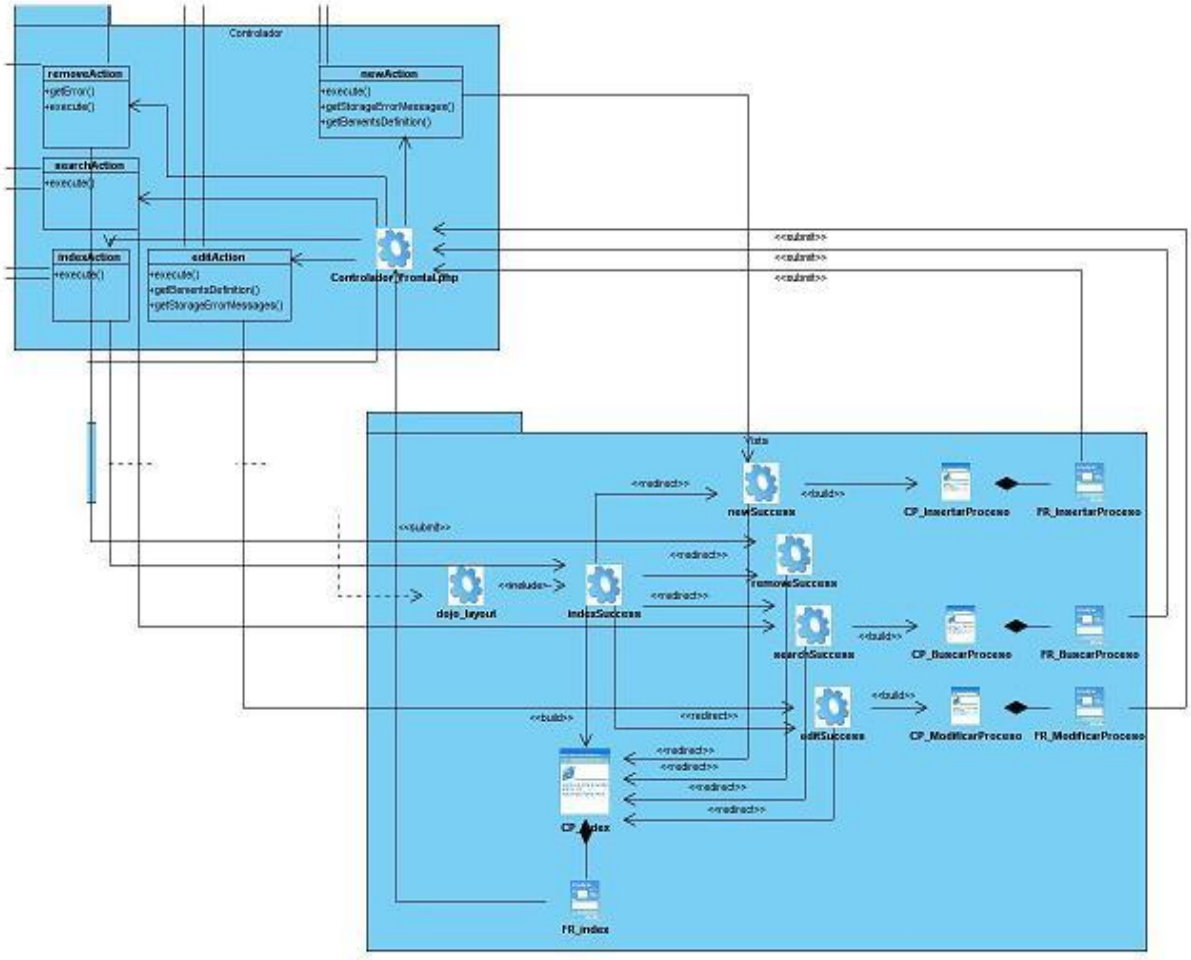

Figura 4: Ejemplo del uso del patrón Alta Cohesión.

Bajo acoplamiento.

Es la idea de tener las clases lo menos relacionadas entre sí que se pueda, logrando que no dependan de muchas otras. De esta forma, si se produce una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Un ejemplo de este patrón se puede apreciar en las clases controladoras: las acciones. Estas clases heredan solamente de sfAction para lograr un bajo acoplamiento de clases.

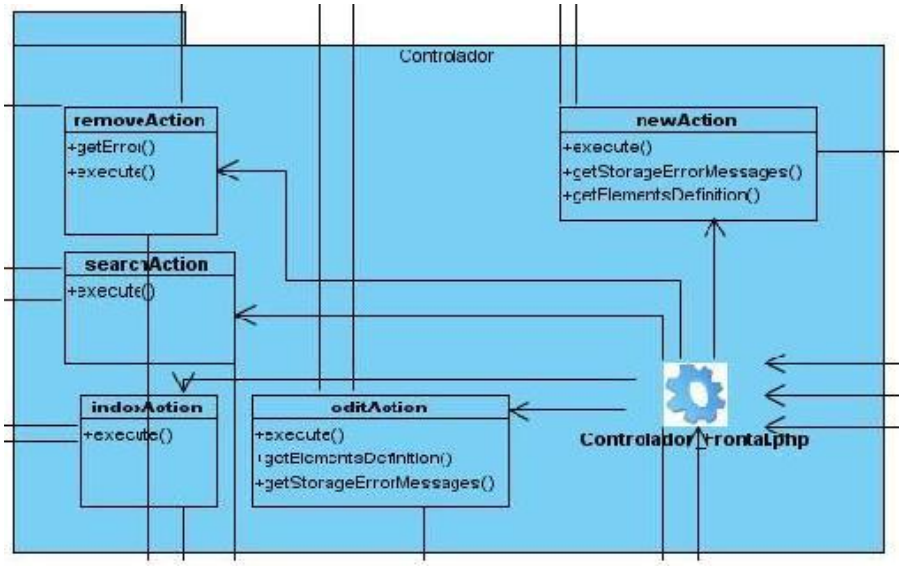

Figura 5: Ejemplo del uso de patrón Bajo Acoplamiento.

Experto.

El patrón GRASP experto en información es el principio básico de asignación de responsabilidades. Consiste en asignar la responsabilidad de creación de un objeto o la implementación de un método al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

El framework Symfony utiliza Propel como ORM (Object-Relational mapping) para lograr una capa de abstracción en el modelo. Encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades. Se generan 4 clases por cada tabla en la base de datos. Para la tabla bpmn_gateway_conditions, por ejemplo, se generan las clases **BpmnGatewayConditions, BpmnGatewayConditionsPeer, BaseBpmnGatewayConditions y BaseBpmnGatewayConditionsPeer,** donde las dos últimas son las encargadas del trabajo con la base

de datos, pues contienen los métodos y atributos necesarios para la realización dicha función. En la figura 6 se muestra la forma en que estas clases manipulan los datos, a través de un fragmento de código.

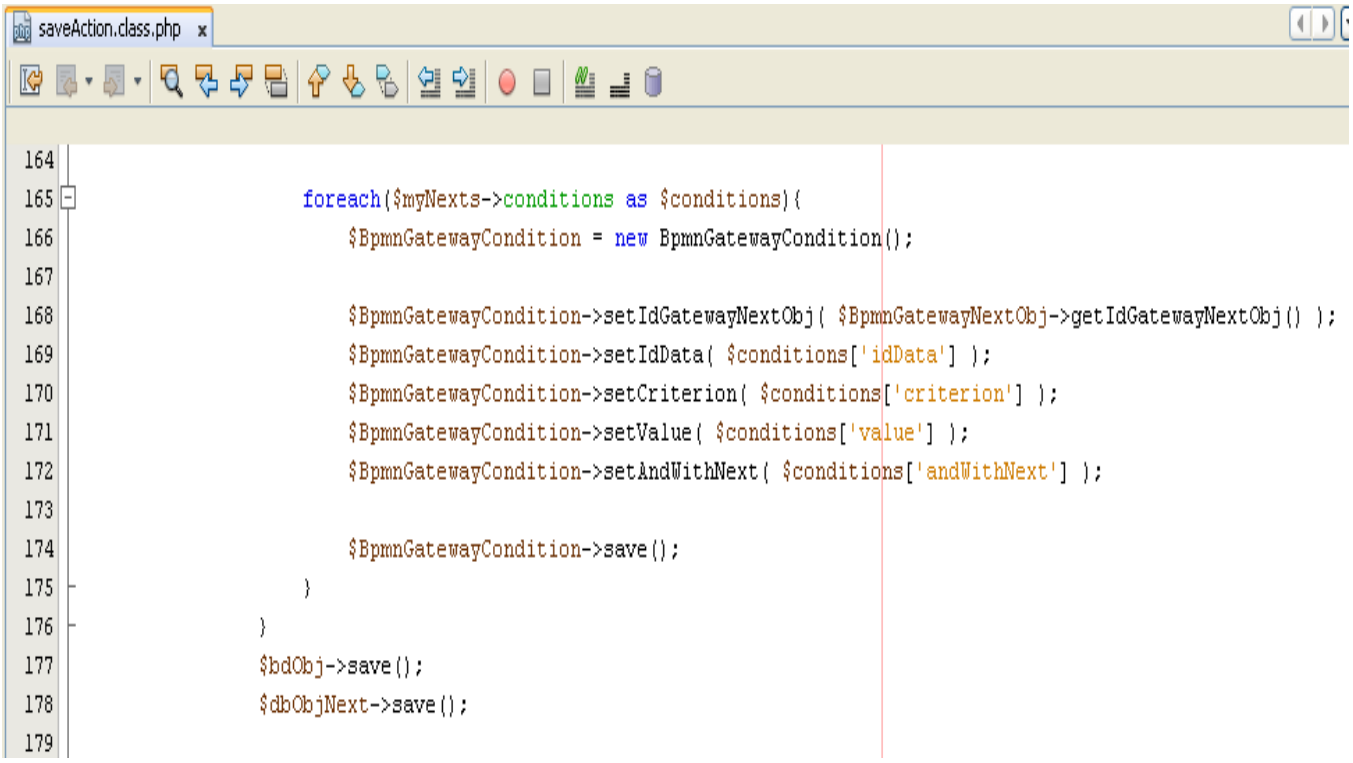

Figura 6: Ejemplo del uso del patrón Experto.

Controlador.

La función de este patrón es ser el intermediario entre la interfaz y el algoritmo que la implementa. Controla los eventos del sistema generado por actores externos. Se asocian con operaciones del sistema, como respuestas a eventos y es el encargado de delegar la responsabilidad de realizarlas a otras clases. Cuando se hace uso del framework Symfony, este patrón se hace común. Además de que el propio framework crea un controlador frontal encargado de manejar todas las peticiones web y es el único punto de entrada de la aplicación en un entorno determinado, queda evidenciado el uso de este patrón cuando se tiene una acción para cada template, encargándose esta de controlar todo un único proceso ya sea de eliminación, inserción o búsqueda de procesos. Como se muestra en la figura 7.

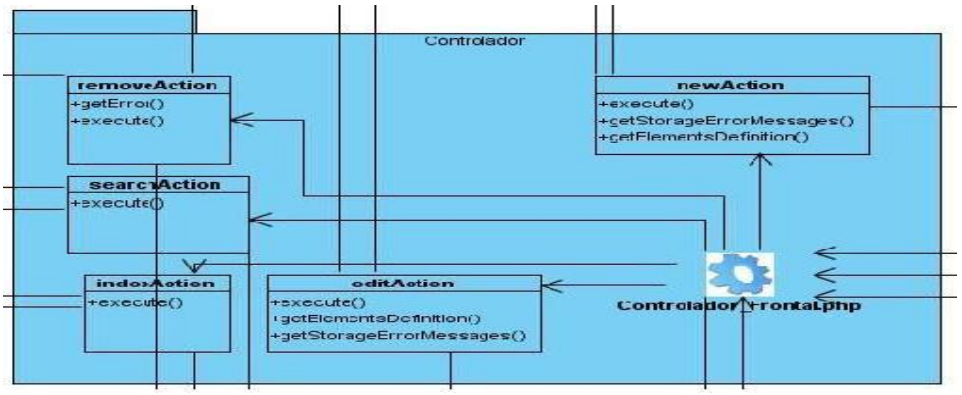

Figura 7: Ejemplo del uso del patrón controlador.

3.2 Modelo de diseño.

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación. (13)

3.2.1 Diagramas de clases del diseño.

Lo que se debe implementar es representado en los diagramas de clases del diseño. Estos diagramas representan la parte estática del sistema a través de las clases y sus relaciones.

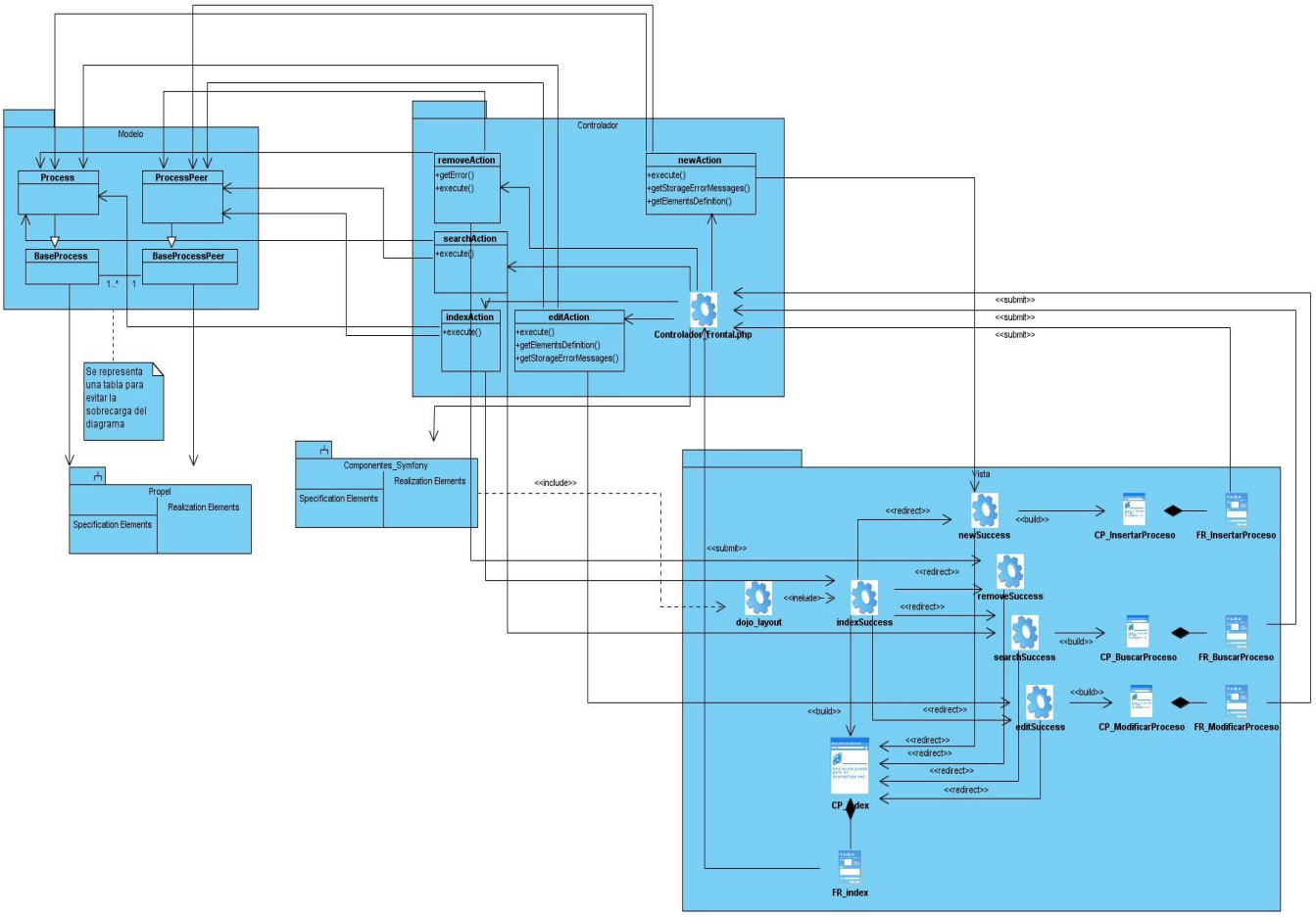

Diagrama 3: Diagrama de clases del diseño: CU Gestionar procesos.

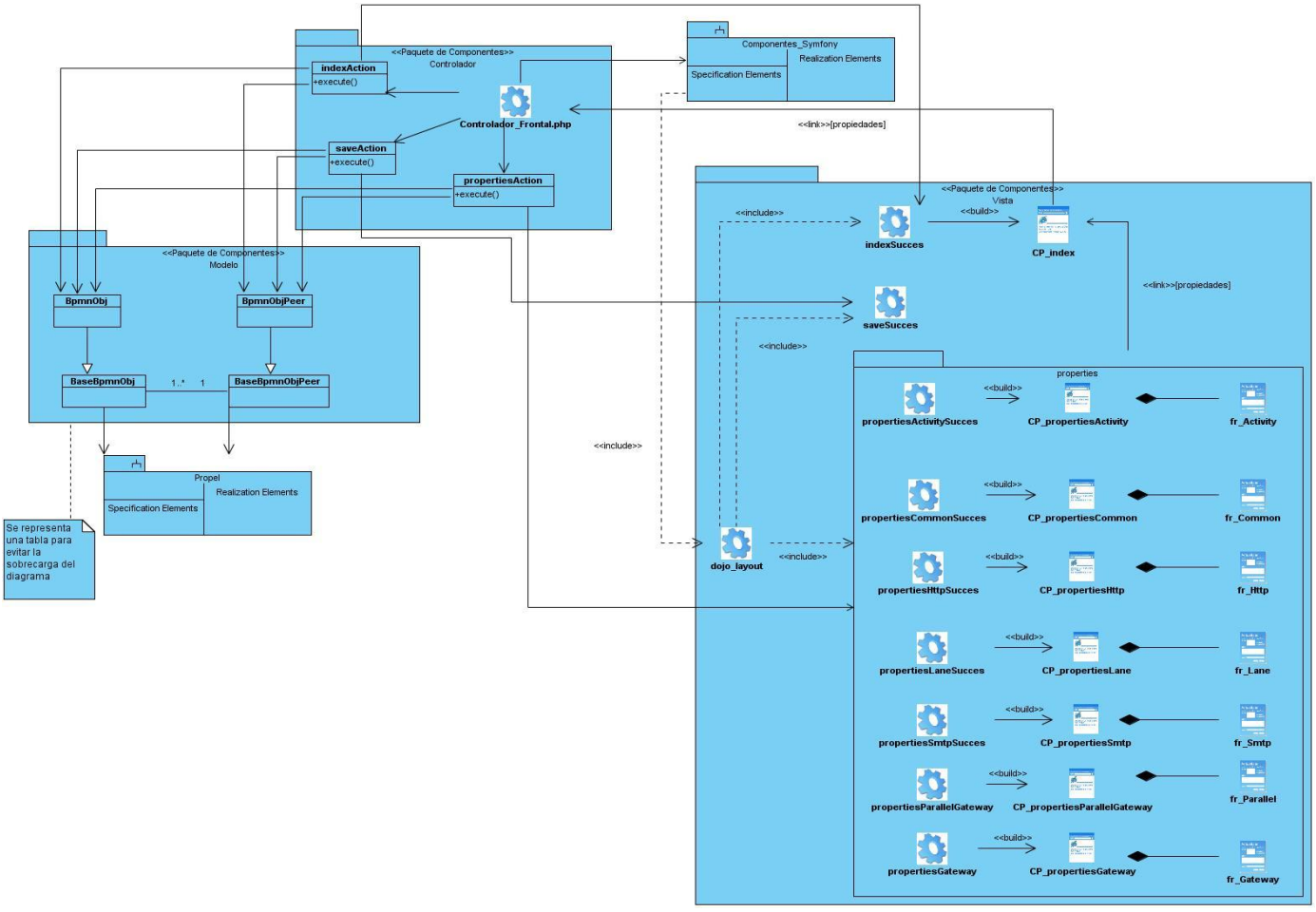

Diagrama 4: Diagrama de clases del diseño: CU Modelar proceso.

3.2.2 Diagramas de interacción.

Los diagramas de interacción muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas, por lo que son empleados para modelar aspectos dinámicos del sistema.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo, estos representan con más claridad el flujo de las acciones que debe realizar el sistema. Como resultado del diseño se obtuvo un diagrama de secuencia por cada escenario de los dos casos de uso. A continuación se muestran 4 diagramas de secuencia, siendo estos los más significativos. El resto de los diagramas se encuentran en los documentos complementarios.

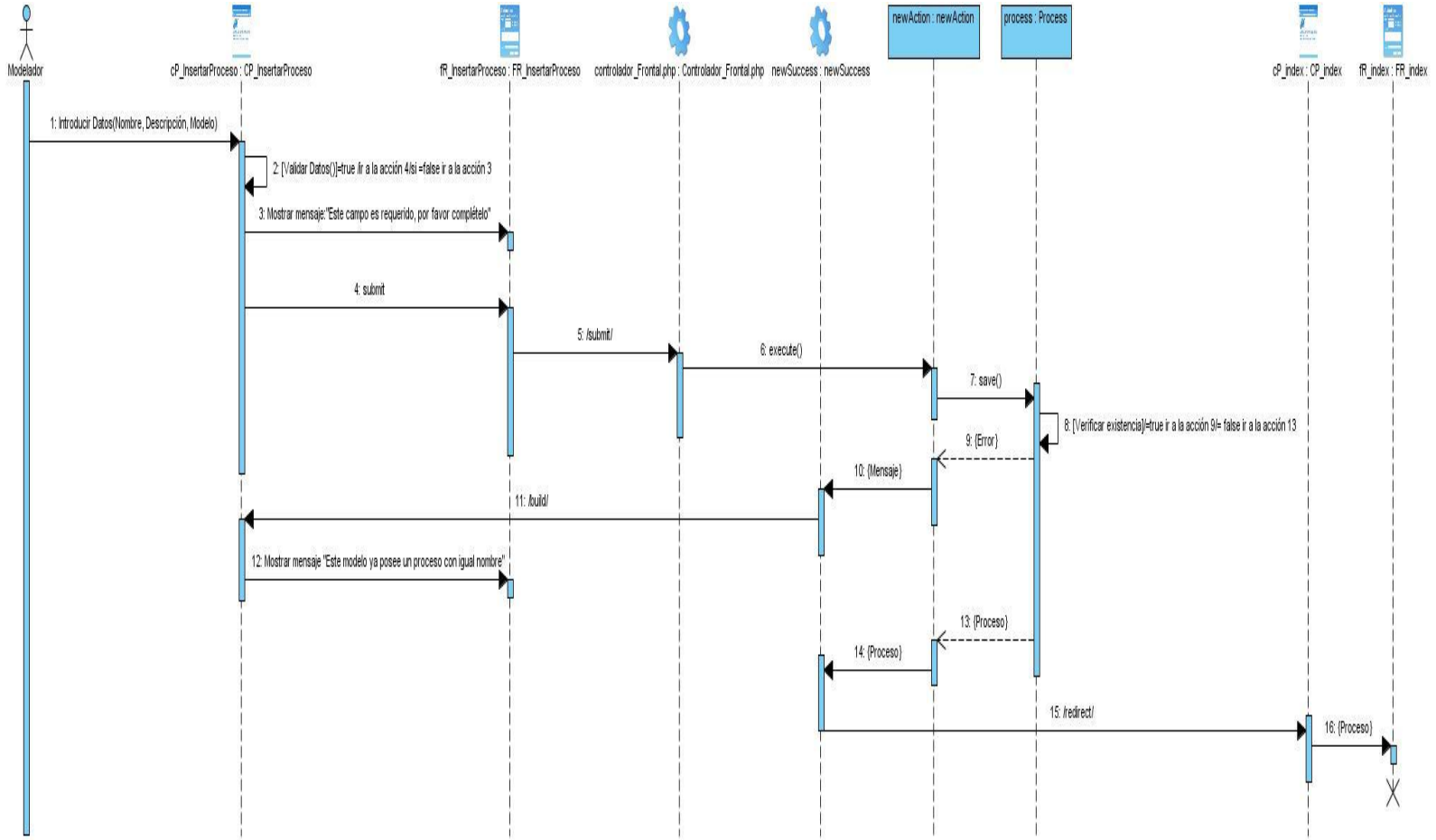

Diagrama 5: Diagrama de secuencia: CU Gestionar procesos (Escenario: Crear proceso).

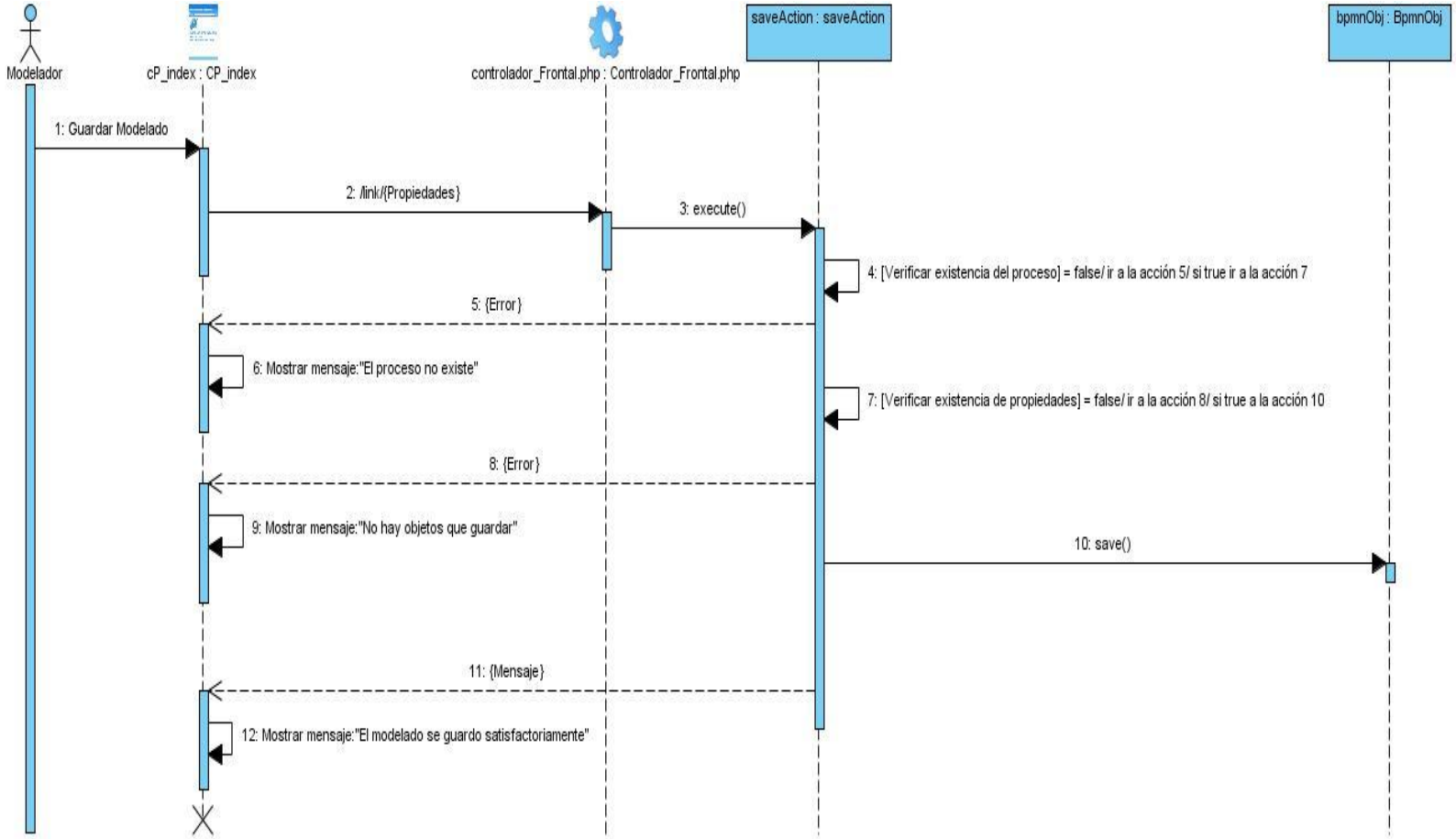

Diagrama 6: Diagrama de secuencia: CU Modelar proceso (Escenario: Guardar modelado).

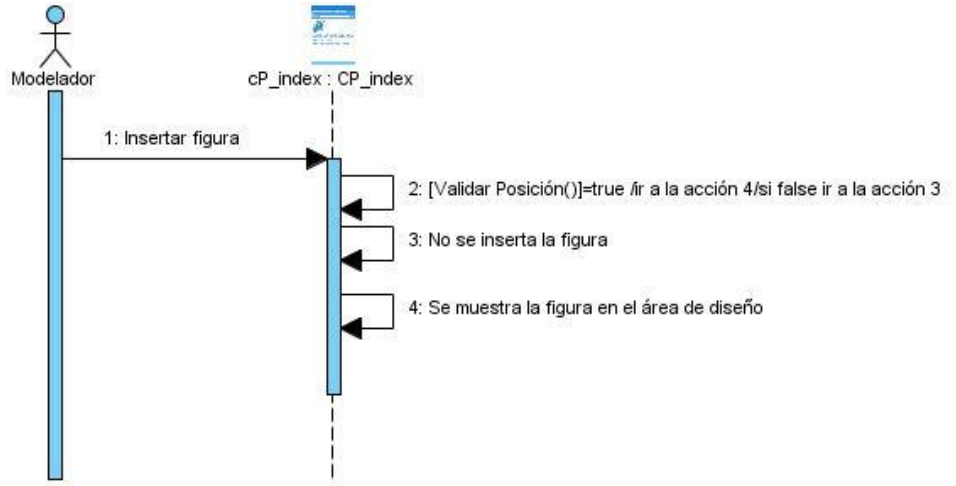

Diagrama 7: Diagrama de secuencia: CU Modelar proceso (Escenario: Insertar objeto).

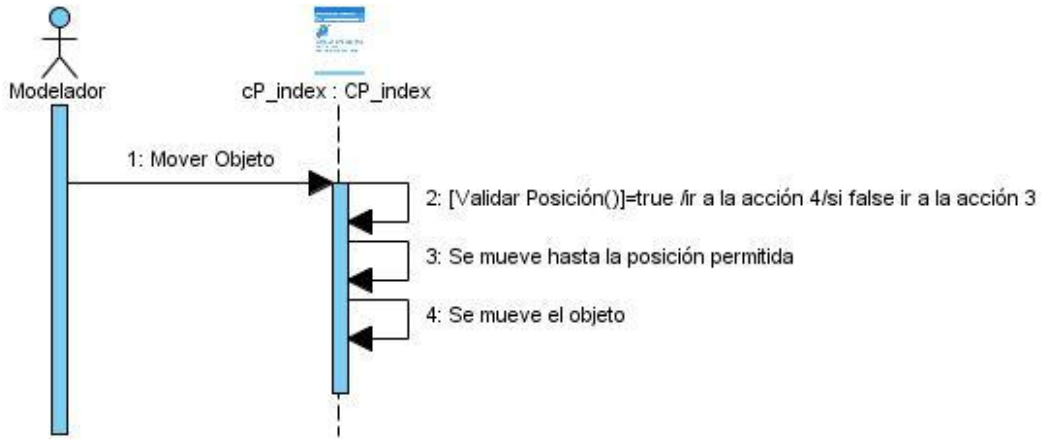

Diagrama 8: Diagrama de secuencia: CU Modelar proceso (Escenario: Mover objeto).

3.3 Diseño de la base de datos.

El diseño de la base de datos es uno de los diagramas de UML que no se debe pasar por alto debido a la importancia que se le atribuye dentro del desarrollo de cualquier sistema de gestión. El modelo de datos describe de una forma abstracta cómo se representan los datos en un sistema de gestión de base de datos. Es una herramienta para especificar los tipos de datos y la organización de los mismos que son permisibles. Este modelo es una guía para el diseño de la base de datos y es el elemento clave en el diseño de la arquitectura de un manejador de la misma.

3.3.1 Diagrama de clases persistentes.

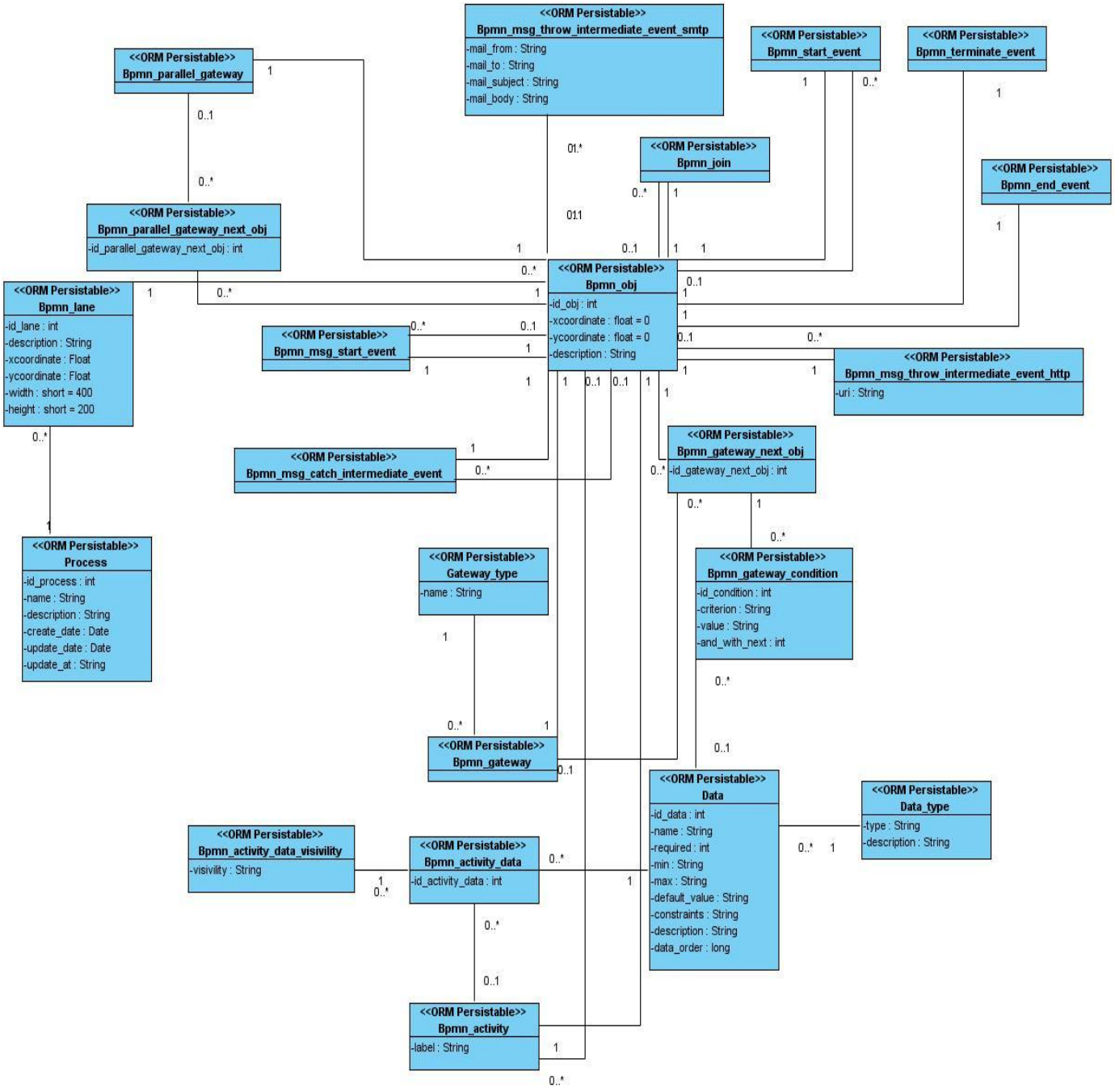

Diagrama 9: Diagrama de clases persistentes.

3.3.2 Modelo de datos.

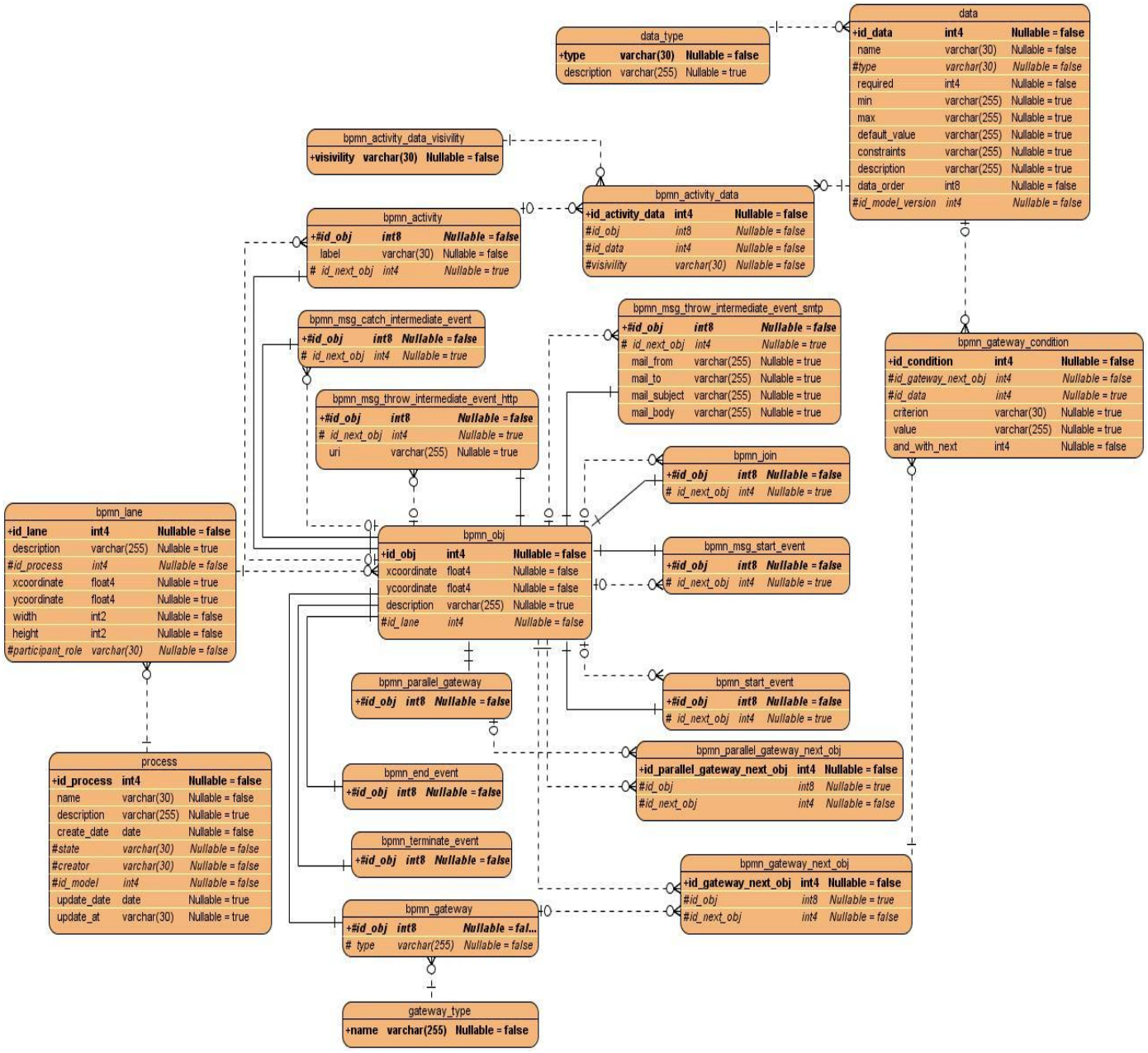

Diagrama 10: Diagrama de entidad relación.

3.4 Diagrama de despliegue.

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Forma parte de la vista de despliegue de UML, la cual representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Por sí mismo representa una correspondencia entre la arquitectura software y la arquitectura del sistema (el hardware).

El diagrama de despliegue planteado por los autores está compuesto por cuatro nodos: el servidor de aplicaciones, el microprocesador que servirá de servidor de base de datos, la terminal de trabajo de los clientes y por último la impresora. Este dispositivo será usado por el sistema una vez que el usuario seleccione la opción de imprimir un determinado contenido. Los protocolos de comunicación entre los nodos serán: HTTP entre la PC_Cliente y el servidor de aplicaciones, TCP/IP entre los dos servidores y USB entre la PC_Cliente y la Impresora.

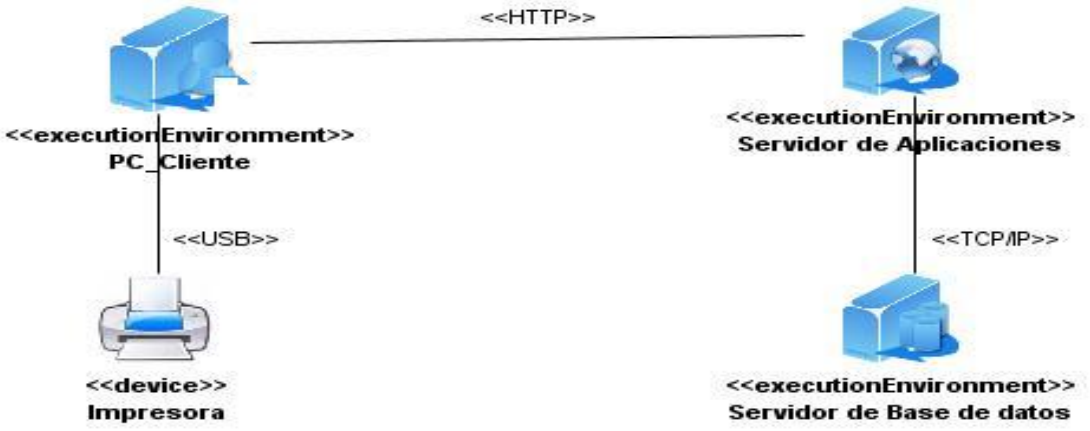

Diagrama 11: Diagrama de despliegue.

3.5 Conclusiones parciales capítulo 3.

En la culminación de este capítulo “Diseño del Sistema” se ha dado cumplimiento a uno de los objetivos específicos trazados, el diseño de la herramienta para la modelación de procesos de gestión de la información. El patrón MVC, así como la aplicación de los patrones de diseño descritos en secciones anteriores, han facilitado el diseño de las clases permitiendo que se obtenga una arquitectura estable y sólida, así como clases mejor diseñadas, modulares, flexibles y reutilizables. Se obtuvo un diagrama de clases del diseño por cada caso de uso, además de un diagrama de secuencia por cada escenario de los casos de uso, el modelo de datos y el diagrama de despliegue del sistema.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se describirá la implementación de la herramienta para la modelación gráfica de los procesos a partir del diagrama de componentes y se determinan las pruebas a realizar en el sistema para verificar su integridad y si se ajusta a los requerimientos planteados por el cliente. Además, se presentan varias pantallas de la aplicación donde se muestra el sistema dando cumpliendo a cada uno de los requisitos planteados en los requerimientos funcionales, así como los errores más comunes que son mostrados al cliente.

4.1 Modelo de implementación.

En la etapa de implementación se comienza con el resultado de la etapa de diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

El objetivo principal de la etapa de implementación es desarrollar la arquitectura y el sistema como un todo. De forma más específica, los propósitos de la implementación son:

- Definir la organización del código.
- Planificar las integraciones del sistema necesarias en cada iteración.
- Implementar las clases y subsistemas encontrados durante el diseño.

El artefacto fundamental que se desarrolla en esta etapa es:

- Modelo de Implementación, que incluye componentes, subsistemas y el producto.

4.1.1 Diagrama de componentes.

En los diagramas de componentes se describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros. Se realizó un diagrama de componentes por cada caso de uso. En los mismos queda evidenciado el uso del patrón arquitectónico MVC al agrupar los componentes en tres paquetes, el modelo, la vista y el controlador.

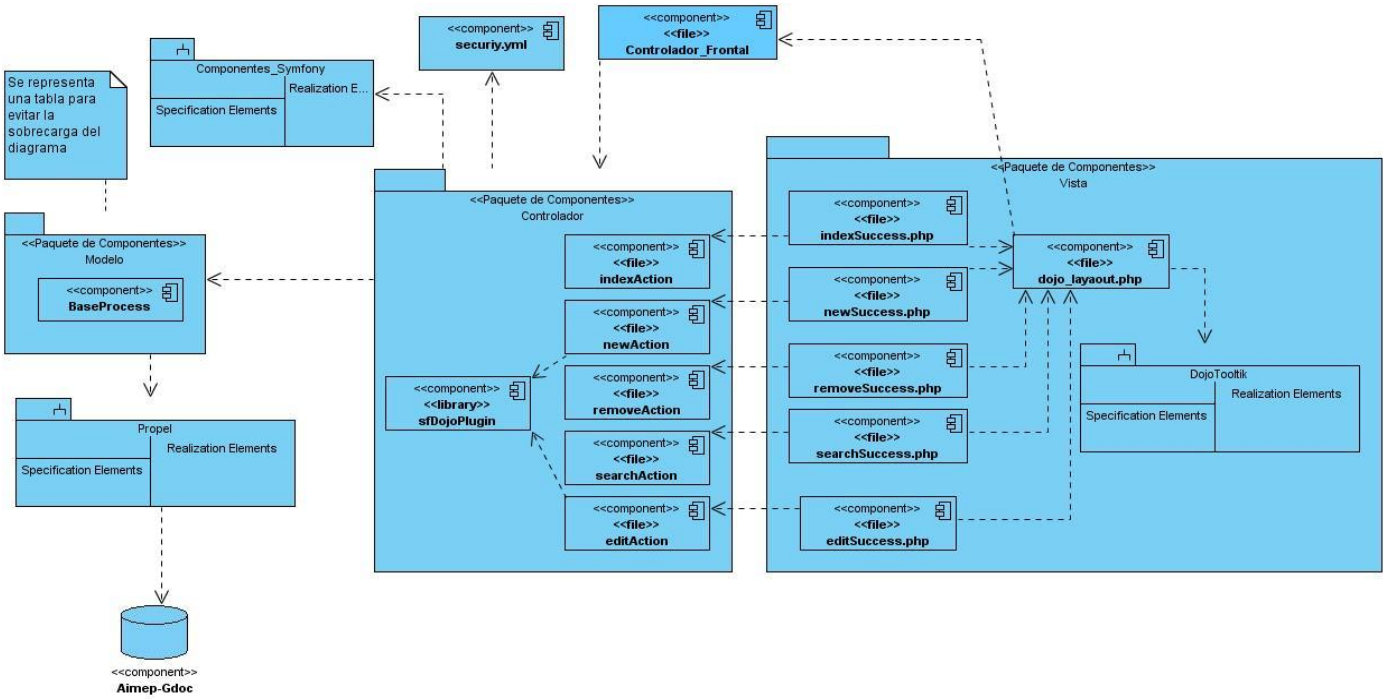

Diagrama 12: Diagrama de componentes CU: Gestionar procesos.

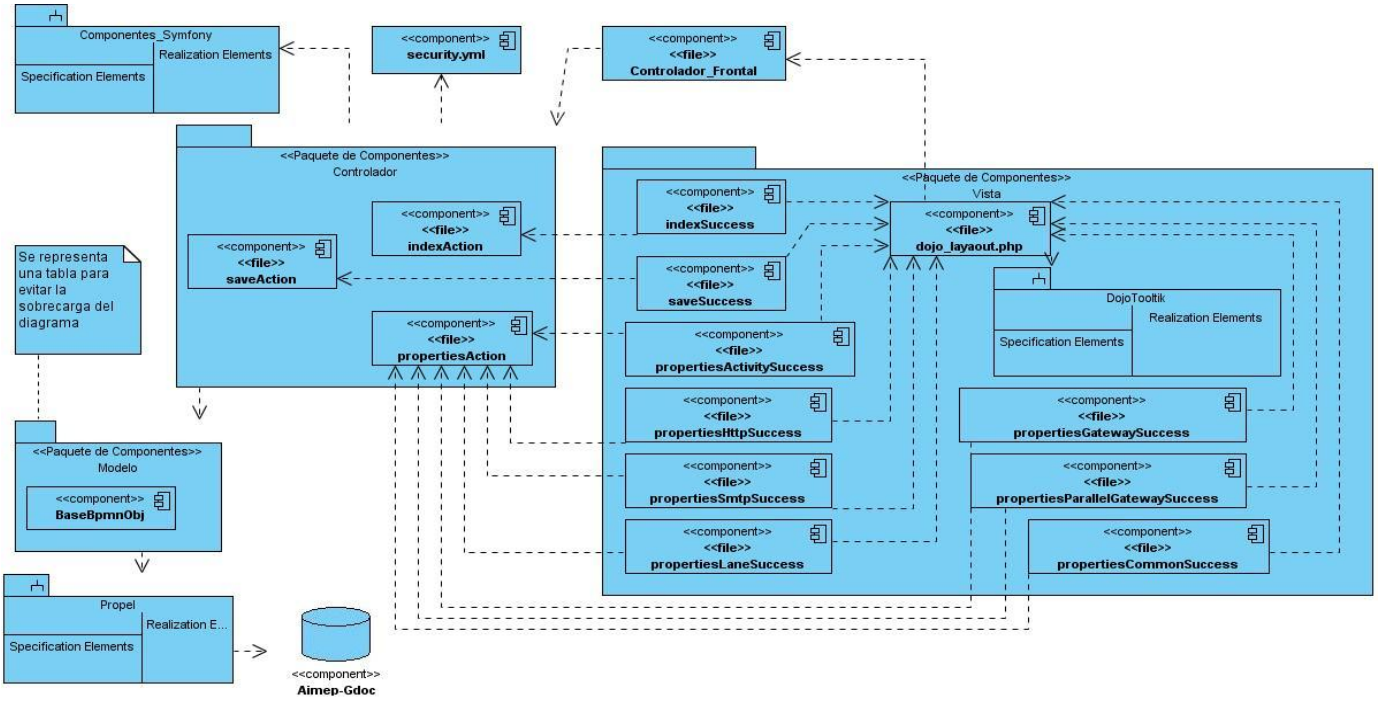

Diagrama 13: Diagrama de componentes CU: Modelar proceso.

4.2 Segmentos principales del código fuente.

Segmento de código de la clase Process.php perteneciente al modelo.

La clase Process.php es la creada por el ORM Propel para el trabajo con la tabla de la base de datos, process. La misma hereda de la clase **BaseProcess.php**, que es la que cuenta con los métodos y atributos que permiten la interacción con la base de datos sin necesidad de conocer ningún lenguaje SQL[10]. Para dar cumplimiento a una de las reglas del negocio establecidas, de no poder eliminar un proceso que su estado sea activo, es necesario redefinir el método **delete()**, como se muestra en el siguiente fragmento de código.

```
    public function delete(PropelPDO $con = null){

        if( $this->getState() != "active" ){
            try{
                parent::delete();
            }catch (Exception $exc){
                throw new Exception(sfContext::getInstance()->getI18N()-
>__('CAN_NOT_DELETE_PROCESS_WITH_INSTANCE'));
            }
        }else{
            throw new Exception(sfContext::getInstance()->getI18N()-
>__('CAN_NOT_DELETE_ACTIVE_PROCESS'));
        }

    }
```

Fragmento de código 1: Método delete de la clase Process.php.

Además, se captura un posible error que puede lanzar la base de datos a la hora de eliminar un proceso y es que el mismo cuente con instancias de ejecuciones.

Segmento de código JavaScript perteneciente al archivo workflow.js.

Este archivo hereda de **dojox.drawing.Drawing,** que es el que contiene los métodos necesarios para la creación de objetos gráficos. Se redefinieron y agregaron algunos métodos a este archivo con el objetivo de adaptarlo a las necesidades de la herramienta. Las funciones redefinidas más importantes fueron: **exporter()** e **importer().** Estas funciones son las encargadas de exportar y cargar un arreglo JavaScript el

10 Structured Query Language

cual contendrá toda la información de los objetos BPMN. Cada uno de los objetos creados para dar solución al problema planteado, contienen a su vez un método **exporter()** el que es capaz de devolver las propiedades que posee cada uno, en un arreglo.

```
exporter: function(){
    var result=[];
    var stencils = this.stencils.stencils;
    for(var i in stencils) {
        if (stencils[i] instanceof dojoy.drawing.bpmn.stencil.swimlane) {
            result.push(stencils[i].exporter())
        }
    }

    return result;
},
```

Fragmento de código 2: Función exporter del archivo workflow.js

```
importer: function( swimlanes ) {
    for( var i in swimlanes ){
        var swimlane = this.addStencil('swimlane', swimlanes[i]);

        for( var z in swimlanes[i].subObjs ){
            var o = dojo.clone( swimlanes[i].subObjs[z] );
            o.next = null;
            o.swimlane = swimlane.id;
            this.addStencil(o.type, o );
        }
    }

    for( var x in swimlanes ){
        for( var y in swimlanes[x].subObjs ){
              if( swimlanes[x].subObjs[y].next ){
                  var obj = this.stencils.stencils[ swimlanes[x].subObjs[y].id ];

                  obj.setNext( swimlanes[x].subObjs[y].next );

              }
        }
    }

}
```

Fragmento de código 3: Función importer del archivo workflow.js.

Segmento de código de la clase saveAction.class.php perteneciente al módulo designer.

La clase **saveAction.class.php** es la encargada de guardar la información referente a los objetos BPMN en la base de datos. La misma recibe por parámetros un Json[11]. Esta cadena de texto eran objetos JavaScripts que fueron codificados del lado del cliente utilizando la función **toJson()** que provee Dojo. Además se recibe el identificador del proceso, al cual pertenece el diseño a guardar.

```
$values =  json_decode( $request->getParameter('wf') );
$id_process = $request->getParameter('id_process');
$process = ProcessPeer::retrieveByPK( $id_process );
```

Al capturar los valores, se pasa a decodificar el Json para poder trabajarlo como un arreglo y obtener el proceso por su identificador.

Se borran todos los datos anteriores que tuviera el proceso. Las relaciones establecidas en la base de datos, posibilita que la eliminación de todos los objetos asociados a un proceso sea una tarea sencilla, con solo eliminar cada uno de los objetos contenedores, en este caso los lanes**.**

```
try {

      foreach ($process->getBpmnLanes() as $lane ){
          $lane->delete();
      }
      $this->nexts = array();
```

El siguiente foreach es para recorrer el arreglo de lanes, obtener sus valores y guardarlos en la BD.

```
      foreach ($values as $sl) {

          $lane = new BpmnLane();
          $lane->setIdProcess( $id_process );
          $lane->setParticipantRole( $sl->participant );
          $lane->setXcoordinate( $sl->data->x );
          $lane->setYcoordinate( $sl->data->y );
          $lane->setHeight( $sl->data->height );
          $lane->setWidth( $sl->data->width );
          $lane->setDescription( ($sl->description) ? $sl->description: null );

          $lane->save();
```

11 JavaScript Object Notation

Los lanes contienen un arreglo de sub-objetos, el cual debe ser recorrido para ir obteniéndolos uno a la vez e ir guardándolos en la BD. Para la creación de cada uno de los objetos se definieron métodos encargados de esas tareas.

```
                foreach ($sl->subObjs as $subObjs){
                    if( $subObjs->type === 'activity' ){
                        $this->setActivity( $lane->getIdLane(), $subObjs );
                    }else if($subObjs->type === 'startEvent'){
                        $this->setStarEvent( $lane->getIdLane(),$subObjs );
                    }else if($subObjs->type === 'msgStartEvent'){
                        $this->setMsgStartEvent( $lane->getIdLane(),$subObjs );
                    }else if($subObjs->type === 'msgCatchIntermediateEvent'){
                        $this->setMsgCatchIntermediateEvent( $lane-
>getIdLane(),$subObjs );
                    }else if($subObjs->type === 'msgThrowIntermediateEventHttp'){
                        $this->setMsgThrowIntermediateEventHttp( $lane-
>getIdLane(),$subObjs );
                    }else if($subObjs->type === 'msgThrowIntermediateEventSmtp'){
                        $this->setMsgThrowIntermediateEventSmtp( $lane-
>getIdLane(),$subObjs );
                    }else if($subObjs->type === 'endEvent'){
                        $this->setEndEvent( $lane->getIdLane(),$subObjs );
                    }else if($subObjs->type === 'terminateEvent'){
                        $this->setTerminateEvent( $lane->getIdLane(),$subObjs );
                    }else if($subObjs->type === 'joinGateway'){
                        $this->setJoinGateway( $lane->getIdLane(),$subObjs );
                    }else if( $subObjs->type == 'inclusiveGateway' || $subObjs->type
== 'exclusiveGateway' ){
                        $this->setGateway( $lane->getIdLane(),$subObjs );
                    }else if($subObjs->type === 'parallelGateway'){
                        $this->setparallelGateway( $lane->getIdLane(),$subObjs );
                    }

                }
            }

            /*
```

La mayoría de los objetos contienen el identificador del objeto que lo precede. Pero en el momento que se crean cada uno de ellos es imposible guardar esta información ya que se puede estar haciendo referencia a un objeto que no haya sido creado aún.

```
          * Aquí es donde se ponen los nexts
            */
            foreach($values as $sl){
                /*
                *   Recorer cada uno de los subObjs que contiene cada lane
                */
                foreach ($ls->subObjs as $a) {
```

```
                    if($a->type == "parallelGateway"){
                        $bdObj = $this->nexts[$a->id]; // Objeto bd actual

                        foreach ($a->nexts as $myNexts){
                            $BpmnParallelNextObj = new BpmnParallelGatewayNextObj();

                            $dbObjNext = $this->nexts[$myNexts]; // Objeto bd next

                            $BpmnParallelNextObj->setIdObj( $bdObj->getIdObj() );
                            $BpmnParallelNextObj->setIdNextObj( $dbObjNext->getIdObj()
);

                            $BpmnParallelNextObj->save();
                        }
                        $bdObj->save();
                        $dbObjNext->save();

                    }else if($a->type == "inclusiveGateway" || $a->type ==
"exclusiveGateway"){
                        $bdObj = $this->nexts[$a->id]; // Objeto bd actual

                        foreach($a->nexts as $myNexts){
                            $BpmnGatewayNextObj = new BpmnGatewayNextObj();

                            $dbObjNext = $this->nexts[$myNexts->idNext]; // Objeto bd
next

                            $BpmnGatewayNextObj->setIdObj( $bdObj->getIdObj() );
                            $BpmnGatewayNextObj->setIdNextObj( $dbObjNext->getIdObj()
);

                            $BpmnGatewayNextObj->save();

                            foreach($myNexts->conditions as $conditions){
                                $BpmnGatewayCondition = new BpmnGatewayCondition();

                                $BpmnGatewayCondition->setIdGatewayNextObj(
$BpmnGatewayNextObj->getIdGatewayNextObj() );
                                $BpmnGatewayCondition->setIdData(
$conditions['idData'] );
                                $BpmnGatewayCondition->setCriterion(
$conditions['criterion'] );
                                $BpmnGatewayCondition->setValue( $conditions['value']
);
                                $BpmnGatewayCondition->setAndWithNext(
$conditions['andWithNext'] );

                                $BpmnGatewayCondition->save();
                            }
                        }
                        $bdObj->save();
                        $dbObjNext->save();
```

```
                    }else {
                        /*
                        * En los demás casos todos son nexts simples
                        * en caso de que tengan siguientes
                        */
                        if($a->next){
                            $bdObj = $this->nexts[$a->id]; // Objeto bd actual
                            $dbObjNext = $this->nexts[$a->next]; // Objeto bd next
                            $dbObj->setIdNextObj( $dbObjNext->getIdObj() ); // Los
relaciono

                            $dbObj->save();
                            $dbObjNext->save();
                        }
                    }
                }
            }
        } catch (Exception $exc) {
             $this->result = array(
                 'status' => 'failed',
                 'errors' => $exc->getMessage()
             );
        }
```

Fragmento de código 4: Método execute de la clase saveAction.class.php.

4.3 Pantallas principales de la aplicación.

La herramienta de modelación de procesos de gestión de la información consta con una interfaz principal la cual gestiona los procesos. Los procesos son listados en una tabla y en la parte superior se ubican botones que representan las distintas acciones que se pueden realizar con los procesos, como son: adicionar un nuevo proceso, editar o eliminar un proceso existente y buscar procesos, véase figura 8. También cuenta con la opción de diseñar un proceso, o sea, la modelación gráfica, que permitirá establecer en qué orden y como es que se va a realizar la recopilación de datos, a través de los modelos de recogida de datos.

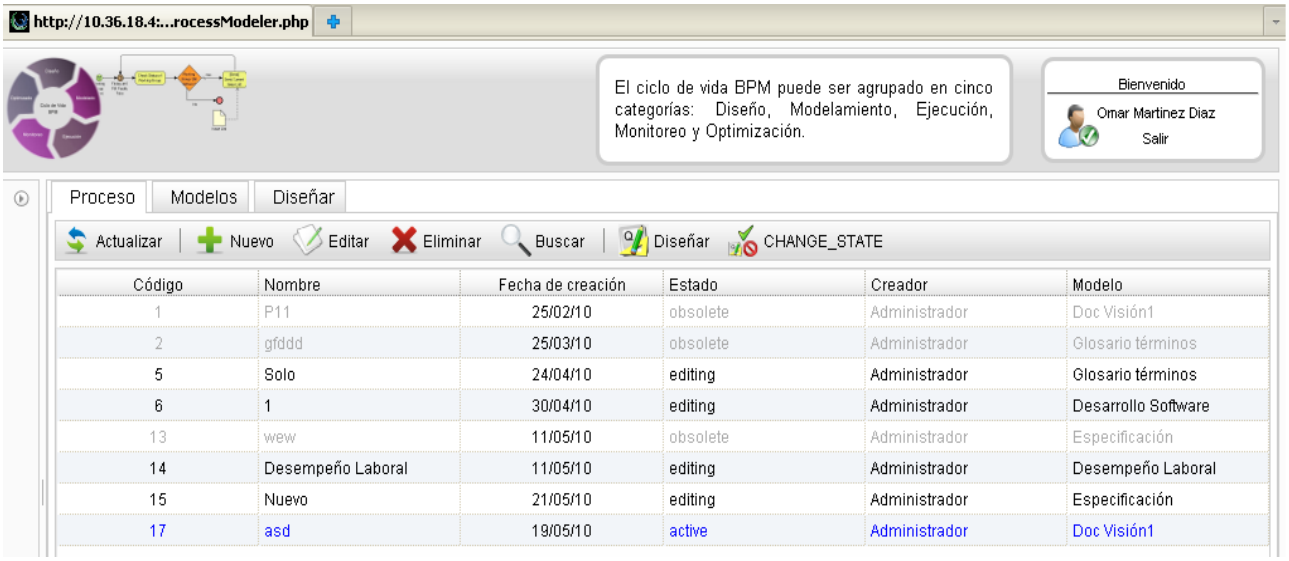

Figura 8: Interfaz Principal.

A continuación se describen cada una de las funcionalidades que presenta la herramienta desarrollada.

Insertar Proceso.

Dentro de las operaciones que puede realizar el usuario en la interacción con la aplicación está la inserción de un nuevo proceso. Para ello se necesita que se llenen los datos requeridos. En la figura 9 se representada dicha operación.

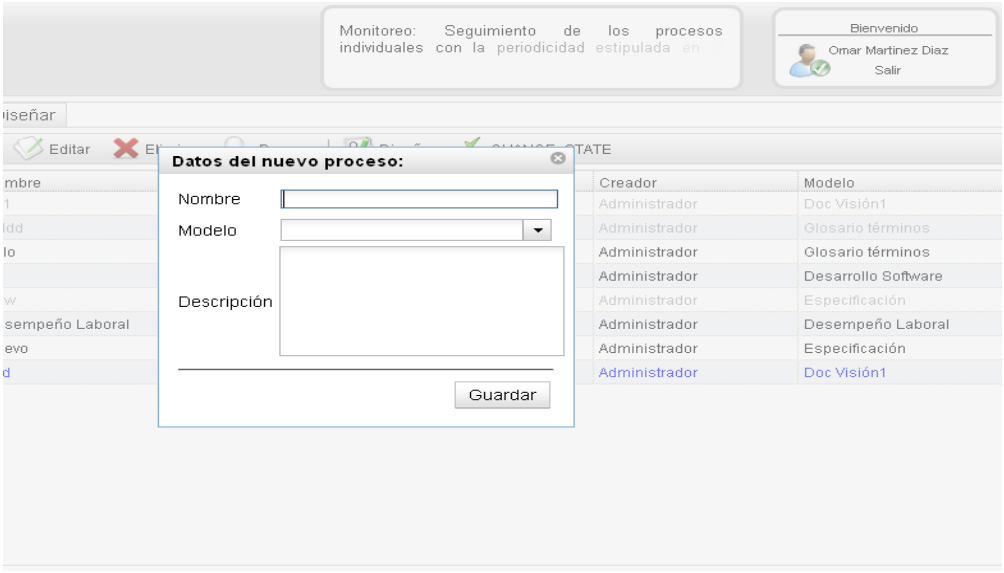

Figura 9: Interfaz de inserción de procesos.

A la hora de crear o editar un proceso hay que tener en cuenta que no se puede repetir el nombre del proceso si se le aplicara al mismo modelo. El mensaje de error que lanza la aplicación es el que se muestra a continuación en la figura 10.

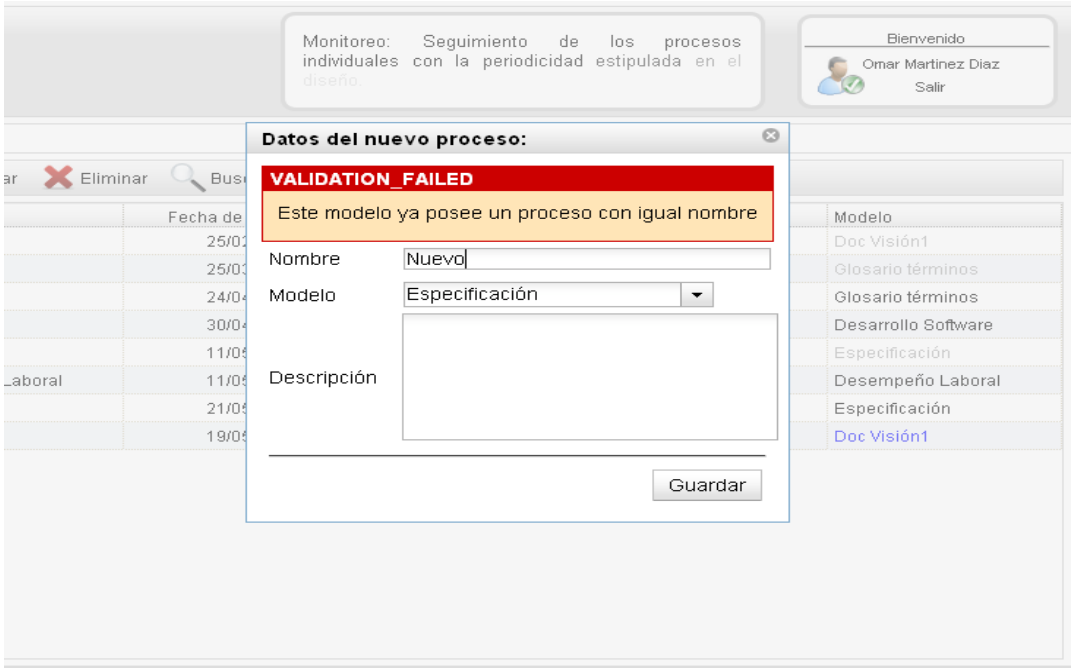

Figura 10: Error de inserción de un proceso.

Buscar Procesos

Esta es una de las funcionalidades más importantes a la hora de gestionar los procesos, pues sería muy complicado el trabajo cuando se tenga un gran número de estos. A la hora de realizar una búsqueda de procesos, se mostrará un diálogo en el cual se podrán ir adicionando criterios para realizar una búsqueda más exacta. Como se muestra en la figura 11.

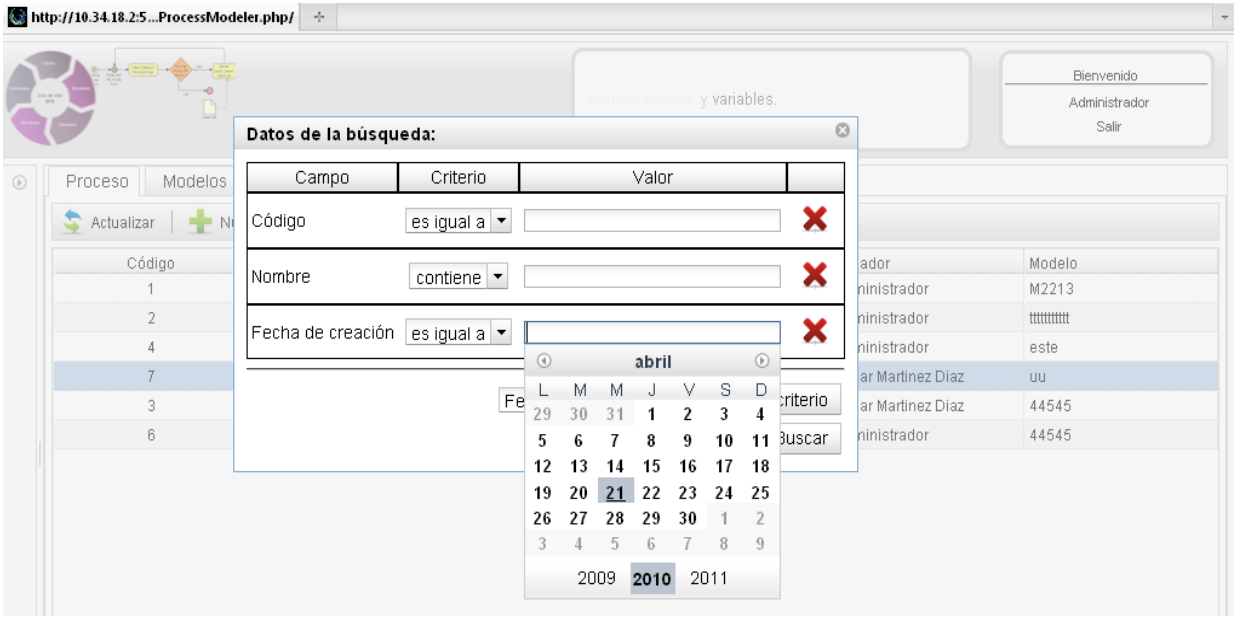

Figura 11: Interfaz para la búsqueda de procesos.

Pantalla principal del área de diseño

Esta pantalla muestra el área donde se modelan los procesos utilizando la notación BPMN. En la parte superior se encuentran los elementos, los cuales pueden ser agregados al área de diseño, como se muestra en la figura 12. Estos elementos cuentan con propiedades las cuales pueden ser modificadas. Para cada uno de los objetos cuenta con características distintas dependiendo de su tipo. Se válida que los valores que se introduzcan son correctos para evitar inconsistencias en la base de datos. Un ejemplo de esta validación se muestra en la figura 13.

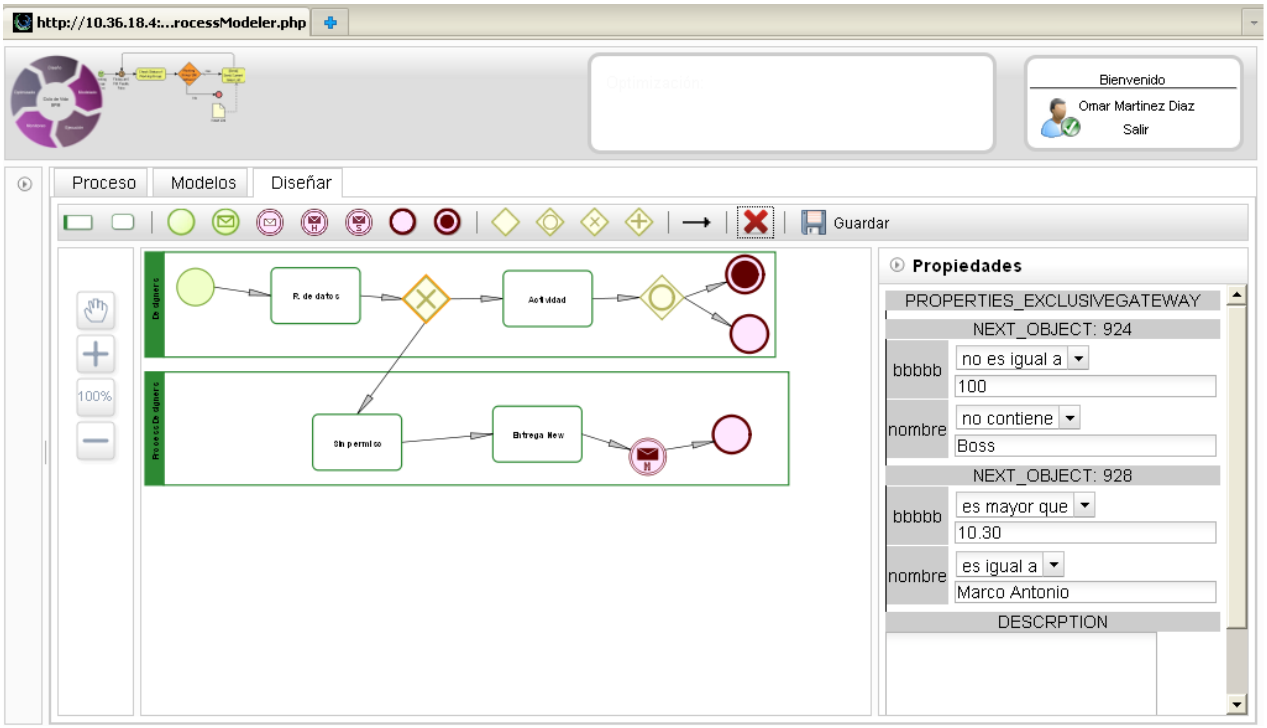

Figura 12: Interfaz principal del área de diseño.

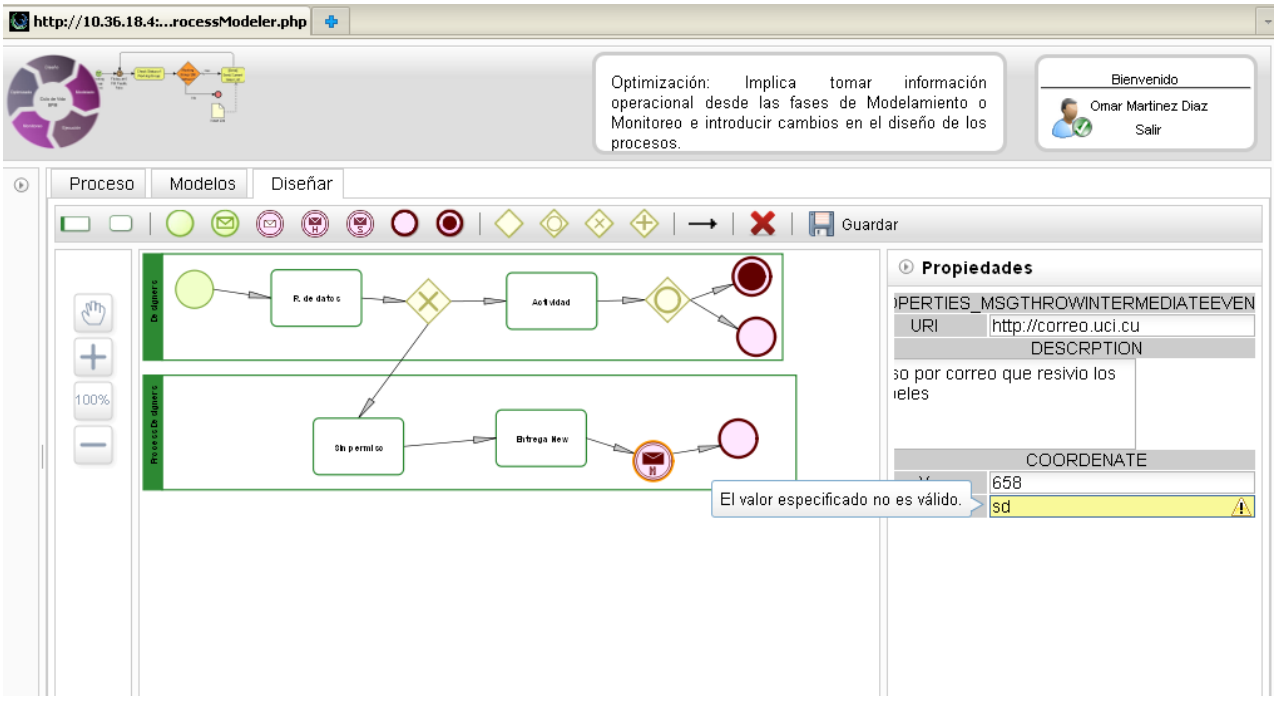

Figura 13: Validación de las propiedades.

4.4 Modelo de prueba.

Las pruebas no son más que los procesos de ejecución de un programa con la intención de descubrir errores. Constituyen un elemento crítico para la garantía de la calidad del software.

Las pruebas que van dirigidas a demostrar que cada función es completamente operativa, son las de Caja Negra. Cuando las pruebas aseguran que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada se hace referencia a las pruebas de Caja Blanca.

A la herramienta de modelación de procesos le fueron realizadas pruebas funcionales. Este tipo de prueba esta basada en la ejecución, revisión y retroalimentación de las funcionalidades previstas. Los resultados de estas pruebas así como los casos de pruebas, pueden ser encontrados en los documentos complementarios.

4.5 Conclusiones parciales capítulo 4.

Al culminar el capítulo “Implementación y Prueba”, se le ha dado cumplimiento al último objetivo específico trazado para el desarrollo de la aplicación, quedando totalmente implementada la herramienta para la modelación gráfica de procesos de gestión de la información. Se generaron los diagramas de componentes, uno por cada caso de uso. Las pruebas funcionales realizadas a la aplicación arrojaron resultados satisfactorios.

CONCLUSIONES

- El estudio realizado sobre las herramientas de modelación gráfica y los procesos de gestión de la información en los centros biotecnológicos permitió una mayor comprensión de las funcionalidades del sistema.
- Se realizó el diseño de la herramienta de modelación gráfica, haciendo un correcto uso de los patrones de diseño.
- Se implementó una herramienta web orientada al usuario, que permite el diseño personalizado de procesos de gestión de la información.

RECOMENDACIONES

- Aumentar el número de objetos BPMN en el modelado, para hacer un modelado más detallado y con mayores opciones.
- Dar la posibilidad al usuario de poder exportar a formato duro los modelos gráficos de los procesos.

REFERENCIAS BIBLIOGRÁFICAS

1. Ministerio de Política Territorial. [En línea] [Citado el: 22 de octubre de 2009.] http://www.map.es/iniciativas/mejora_de_la_administracion_general_del_estado/servicios_publicos/evaluacion_calidad/calidad/programas_basicos/pg_evalua/documento_efqm/efqm_criterios/document_es/Modelo_EFQM__202003_CRITERIOS.pdf.

2. ***White, Stephen A.; IBM Corporation.*** Introduction to BPMN.

3. ***Bartle, Phil.*** *[En línea] [Citado el: 11 de Noviembre de 2009.] http://www.scn.org/mpfc/modules/mon-miss.htm.*

4. Universidad Politécnica de Puebla. [En línea] [Citado el: 3 de Noviembre de 2009.] http://www.uppuebla.edu.mx/Profesores/INFORMATICA/PRODUCCION_CIENTIFICA/REBECA/Ingsoftware_II/Apuntes/6.herramicase.pdf.

5. ibercom. [En línea] [Citado el: 20 de noviembre de 2009.] https://www.ibercom.com/soporte/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=85.

6. Ajax.org. [En línea] [Citado el: 15 de noviembre de 2009.] http://www.ajax.org/.

7. Biblioteca Central Universidad de San Carlos de Guatemala. [En línea] Junio de 2008. [Citado el: 17 de Noviembre de 2009.] http://biblioteca.usac.edu.gt/tesis/08/08_8506.pdf.

8. ***Potencier, Fabien y Zaninotto, François.*** Symfony, La guía definitiva 1.2.

9. The Dojo Toolkit. [En línea] [Citado el: 15 de noviembre de 2009.] http://dojotoolkit.org/.

10. Sun Microsystems. [En línea] [Citado el: 15 de noviembre de 2009.] http://es.sun.com/.

11. ***The Apache Software Foundation.*** *Apache. [En línea] http://httpd.apache.org.*

12. ***Fowler, Martin.*** Patterns of enterprise application architecture.

13. ***Ivar Jacobson, Grady Booch and James Rumbaugh.*** El Proceso unificado de desarrollo de software. *Madrid : Addison Wesley, 2000. 84-7829-036-2.*

BIBLIOGRAFÍA

1. *ADONIS: Community Edition. [En línea] [Citado el: 14 de Octubre de 2009.] Disponible en:http://www.es.adonis-community.com/.*
2. *Ajax.org. [En línea] [Citado el: 15 de noviembre de 2009.] Disponible en:http://www.ajax.org/.*
3. ***Bartle, Phil.** [En línea] [Citado el: 11 de Noviembre de 2009.] Disponible en:http://www.scn.org/mpfc/modules/mon-miss.htm.*
4. *Biblioteca Central Universidad de San Carlos de Guatemala. [En línea] Junio de 2008. [Citado el: 17 de Noviembre de 2009.] Disponible en:http://biblioteca.usac.edu.gt/tesis/08/08_8506.pdf.*
5. *Bizagi. [En línea] Disponible en:http://www.bizagi.com/.*
6. ***Bradenbaugh, Jerry.** Aplicaciones JavaScript.*
7. ***CECMED.** Regulación No. 37-2004. Buenas Prácticas de Laboratorios para el Control de Medicamentos. 2004.*
8. *Cuecent | Business Process Management (BPM). [En línea] [Citado el: 14 de Octubre de 2009.] Disponible en:http://www.cuecent.com.*
9. ***Fowler, Martin.** Patterns of enterprise application architecture.*
10. *ibercom. [En línea] [Citado el: 20 de noviembre de 2009.] Disponible en:https://www.ibercom.com/soporte/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=85.*
11. ***IBM.** IBM Workflow Implementation Guide.*
12. ***Ivar Jacobson, Grady Booch and James Rumbaugh.** El Proceso unificado de desarrollo de software. Madrid : Addison Wesley, 2000. 84-7829-036-2.*
13. ***JACOBSON, I.** El proceso Unificado de Desarrollo de Software.*
14. ***López Urrutia, Antonio.** El Laboratorio General: Mecanización y Gestión.*
15. *Ministerio de Política Territorial. [En línea] [Citado el: 22 de octubre de 2009.] Disponible en:http://www.map.es/iniciativas/mejora_de_la_administracion_general_del_estado/servicios_publicos/evaluacion_calidad/calidad/programas_basicos/pg_evalua/documento_efqm/efqm_criterios/document_es/Modelo_EFQM__202003_CRITERIOS.pdf.*

16. ***Odelín Prieto, Yeniseis.*** *Buenas Prácticas de Laboratorio y las normas ISO 9001:2000.*

17. ***PCC.*** *Resolución del V Congreso del Partido Comunista de Cuba. 1997.*

18. ***Pedraza González, Eslavy y Puig Diaz, Pedro Manuel.*** *Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis y Diseño del módulo Biología Molecular. La Habana : s. n., 2008.*

19. ***Potencier, Fabien y Zaninotto, François.*** *Symfony, La guía definitiva 1.2.*

20. ***Pressman, Roger S.*** *Ingeniería de Software, Un enfoque práctico.*

21. ***Rodríguez Cruz, Francisco.*** *El gran descubrimiento de la Biotecnología cubana. Trabajadores.*

22. *Sun Microsystems. [En línea] [Citado el: 15 de noviembre de 2009.] Disponible en:http://es.sun.com/.*

23. *The Apache Software Foundation. [En línea] [Citado el: 16 de noviembre de 2009.] Disponible en:http://www.apache.org/.*

24. ***The Apache Software Foundation.*** *Apache. [En línea] Disponible en:http://httpd.apache.org.*

25. *The Dojo Toolkit. [En línea] [Citado el: 15 de noviembre de 2009.] Disponible en:http://dojotoolkit.org/.*

26. *Universidad Argentina de la Empresa Lima. [En línea] [Citado el: 26 de Octubre de 2009.] Disponible en:http://www.uade.edu.ar/.*

27. *Universidad Politécnica de Puebla. [En línea] [Citado el: 3 de Noviembre de 2009.] Disponible en:http://www.uppuebla.edu.mx/Profesores/INFORMATICA/PRODUCCION_CIENTIFICA/REBECA/Ingsoftware_II/Apuntes/6.herramicase.pdf.*

28. ***Viera Achang, José Rafael y Peña González, Loismarx.*** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Implementación del Módulo de Análisis Químico. La Haban : s. n., 2008.*

29. *Visual Paradigm. [En línea] [Citado el: 21 de 5 de 2010.] Disponible en:http://www.visual-paradigm.com/.*

30. ***White, Stephen A.; IBM Corporation.*** *Introduction to BPMN.*

ANEXOS

GLOSARIO

European Foundation for Quality Management: Fundación Europea para la Gestión de la Calidad, fundada en 1988 por los presidentes de las 14 mayores compañías europeas con el apoyo de la comisión europea.

XPDL: Es un formato de archivo basado en XML que puede ser usado para intercambiar modelos de procesos de negocio entre distintas herramientas. Es un formato de archivo que representa el “dibujo” de la definición del proceso. Tiene el tamaño y las coordenadas X y Y del nodo. Tiene un concepto de líneas que señalan el camino a seguir. Los nodos y las líneas tienen atributos que pueden especificar información ejecutable tales como roles, descripción de actividades, temporizadores, llamadas a web services. XPDL 2.0 contiene extensiones para ser capaz de representar todos los aspectos de BPMN

TCP/IP: Es un conjunto de protocolos. La sigla TCP/IP significa "**Protocolo de control de transmisión/Protocolo de Internet**" y se pronuncia "T-C-P-I-P". Proviene de los nombres de dos protocolos importantes del conjunto de protocolos, es decir, del protocolo TCP y del protocolo IP.

HTTP: Cada transacción de información realizada en la Web es realizada utilizando el protocolo HTTP, "HyperText Transfer Protocol" por sus siglas en inglés, o Protocolo de Transferencia de HyperTexto.

HTTPS: El protocolo HTTPS es la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP.

XLIFF: (XML Localisation Interchange File Format) es un lenguaje derivado del XML que se desarrolló para resolver una serie de problemas existentes en la gestión de la traducción multilingüe.

BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

GPL: Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés) es una licencia libre y gratuita con derecho de copia para software y otros tipos de obras.

HTML: Siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

Widgets: Se puede definir un widget como una pequeña aplicación, que los usuarios pueden instalar en la web, blog, red social favorita o incluso descargarse en un ordenador o teléfono móvil.

Propel: Es un ORM para PHP que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la BD mediante objetos, con la que se puede recuperar, insertar y modificar datos.