# Universidad de las Ciencias Informáticas
# Facultad 6


**Título:** Ambiente Integrado para el modelado y ejecución de procesos de gestión de información. Herramienta web para el diseño de modelos de recogida de datos.

## Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

**Autoras:** Yanet Hernández Rivero.
Glennis Tamayo Morales.

**Tutoras:** Ing. Rosayda Valiente Mesa.
Ing. Aida Portelles Valdes.

**Cotutor:** Lic. Yoandris Pacheco Aguila.

Ciudad de la Habana, Junio 2010
"Año 52 de la Revolución"

*"... Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes."*

*Fidel Castro Ruz*

## 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 _____.

**Glennis Tamayo Morales**

_____
Firma de la Autora

**Yanet Hernández Rivero**

_____
Firma de la Autora

**Rosayda Valiente Mesa**

_____
Firma de la Tutora

**Aida Portelles Valdes**

_____
Firma de la Tutora

## Datos de contacto

Tutora: Ing. Rosayda Valiente Mesa
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Categoría docente: Instructor Recién Graduado
Categoría Científica: no
Años de experiencia en el tema: 0
Años de graduado: 1

Tutora: Ing. Aida Portelles Valdes
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Categoría docente: Instructor Recién Graduado
Categoría Científica: no
Años de experiencia en el tema: 0
Años de graduado: 1

## Agradecimientos

*Quisiera agradecer a todas las personas que han hecho posible que mi sueño se hiciera realidad.*

*Comenzando por mis padres que me dieron todo su apoyo, y depositaron en mí toda su confianza, por su sacrificio, por su dedicación, por su amor y preocupación, por la educación y los principios que me inculcaron.*

*A mis abuelos que siempre han estado pendientes de mí, por su amor y su comprensión, por estar siempre a mi lado.*

*A mi hermano, que es el mejor hermano de este mundo, por confiar en mí y estar siempre conmigo apoyándome en todo.*

*A Luisi que ha sido extraordinario, por su amor, su paciencia, por su dedicación, porque mis sueños también son los suyos, mis preocupaciones y mis problemas también son sus problemas, por ser el hombre más especial de este mundo.*

*A mis suegros Luis y Belci que los quiero como a unos padres, por su amor y confianza. A Yuli que es la cuñada más linda del mundo, por su preocupación y cariño.*

*A toda mi familia los amo y les doy las gracias porque todos han puesto su granito de arena para que mi sueño se hiciera realidad.*

*A mis tutores Aida, Rosayda y Pacheco por su dedicación y apoyo incondicional, en especial al profe Pacheco que es para mí un ejemplo a seguir como profesional y como persona. A nuestro tribunal Jorge, Yanet y Julio, gracias por sus críticas, por su exigencia y por dedicarnos su tiempo.*

*A mis amigas Dayris y Daymelis que en estos cinco años han sido incondicionales, ha estado conmigo en los buenos y en los malos momentos, han aprendido a lidiar con esta pesadita pero que las quiere con el corazón.*

*A mi dúo de tesis Glennis, por ser la mejor compañera, gracias por tu comprensión y colaboración en la realización de este trabajo, en este día tan especial para las dos quiero decirte que aprendí a quererte y para mí eres una amiga más.*

*A mis amigas Yuli y Ene, por todos los momentos que compartimos juntas, por estar siempre a mi lado cuando las necesité, las quiero mucho. A Keiler, Oscar, Omar y Daniel por su colaboración, porque siempre trabajamos juntos e intercambiamos ideas.*

*A todos los profesores que han contribuido en mi educación, en mi formación como profesional y como persona. También quiero agradecer a la revolución, con la que me siento comprometida y a Fidel que fue el promotor de este proyecto, por darnos la oportunidad de estudiar en esta universidad.*

*A todo los que no mencioné, pero que de una forma u otra contribuyeron en la realización de este trabajo. A todos, agradecida infinitamente.*

*Yanet Hernández Rivero.*

## Agradecimientos

*A mis padres por darme el regalo más grande: la vida. Por enseñarme a luchar para hacer realidad mis sueños. Por demostrarme que si se puede. Por todo el amor y cariño infinito que siempre me han acompañado, por creer en mí. Por ayudar en mi formación como mujer, como persona y como profesional. A ustedes los amo.*

*A mis hermanos Adriana e Iván por estar siempre presentes por quererme y cuidar de mi. Los amo.*

*A mis tíos Ciriam y Rafael por ser como unos padres para mí, por apoyarme y ayudarme en todo cuanto he necesitado. Los quiero.*

*A mi familia por todo el amor y el apoyo que siempre he recibido de ellos, por todo cuanto han hecho para hacerme feliz. A todos los amo.*

*A mi novio por su cariño y amor, por todo el apoyo en este tiempo tan importante para mí, por estar siempre a mi lado, por todos los momentos de felicidad. Te quiero chuchú.*

*A los profesores por toda la educación y el conocimiento que pusieron en mis manos, por contribuir en mi formación profesional y personal. A mis tutores Aida y Rosi por todo el tiempo que dedicaron a apoyarnos en la realización de este trabajo. A Pacheco por tantas horas de desvelo, por su preocupación y por su apoyo incondicional.*

*A nuestro tribunal Jorge Luis, Julio y Yanet por el tiempo que incondicionalmente dedicaron en la realización de este trabajo. Gracias a todas las críticas constructivas que recibimos de ustedes logramos un mejor trabajo de diploma.*

*A mis amigas Yuli y Ariadna por soportarme todo este tiempo, por estar siempre presentes, por ayudarme en los momentos difíciles.*

*A mi dúo de tesis Yanet, porque no pude tener un dúo mejor. Por ser una excepcional persona, amiga y muy buena estudiante. Por todas las horas que puso en la realización de este trabajo, por los consejos. Yane te agradezco infinitamente que hayas compartido este momento especial conmigo te quiero mucho y te agradezco por todo. Eres una amiga muy especial.*

*A mi comandante Fidel, a Raúl y a la revolución por darme la oportunidad de estudiar y superarme en esta universidad de excelencia en la que he vivido los 5 años más importantes de mi vida.*

*A todos mis amigos por estar conmigo en los buenos y malos momentos. A Oscar, Keyler, Omar y Daniel por su apoyo y por su ayuda. Gracias.*

*Estoy y estaré agradecida infinitamente de todos los que de una forma u otra contribuyeron y ayudaron en la realización de este trabajo. A todos muchísimas gracias.*

*Glennis Tamayo Morales.*

## Dedicatoria

*Dedico este trabajo a las personas más importantes en mi vida:*

*A mis padres que han sido maravillosos, a mi mamá que es la mejor madre de este mundo, por su ternura, por su sacrificio, por su dedicación, por ser tan compresiva y tolerante con esta niñita malcriada que la ama.*

*A mi papá que es el hombre más bueno, sincero y modesto de este mundo, para mí ha sido, es y será un ejemplo a seguir, mi fuente inspiración, te amo.*

*A mis abuelos Cari y Lalito que también han sido mis padres, estos viejitos lindos el que con su amor y ternura me transformaron en la mujer que soy hoy, los amo con todo mi corazón y no me va a alcanzar la vida de retribuirles su amor.*

*A mis abuelos Migdalia y Pablo que aunque ya no estén los llevo en el corazón y sé que estarán muy orgullosos de mí.*

*A mis hermanos Rodin y Rosalí, que los adoro, por estar siempre conmigo, por su cariño y por todas las cosas lindas que hemos pasado juntos.*

*A mi niñito Ronall que la cosa más linda que la vida nos ha dado, con su alegría, sus ocurrencias, te amo mi pequeñín.*

*Al amor de mi vida Luis, que estos cinco años ha sido mi amigo, mi confidente, mi compañero, por su compresión, su paciencia, su amor, por estar siempre a mi lado en los momentos más difíciles, por hacerme reír cuando lo necesitaba y ayudarme a vencer los obstáculos que la vida ha presentado, te amo tati.*

*Son infinitas las palabras de amor y los años que cada uno de ustedes me han dedicado como para pretender que quepan en un papel. Espero que se sientan orgullosos de mí como lo estoy yo de ustedes. He recibido de la vida el regalo más bello, mi familia.*

*Yanet Hernández Rivero.*

## Dedicatoria

*Dedico este trabajo a los dos tesoros más grandes de mi vida:*

*A la mujer más linda e importante de mi vida, mi madre, que con tanto amor y cariño ha sabido ser mi faro guía, mi ejemplo a seguir, mi más grande orgullo. Por el apoyo incondicional, por su esfuerzo y por el sacrificio que sé que no ha sido poco. Porque no existe madre como la mía siempre presta a ayudar en todo lo que me haga falta.*

*Mamita a ti dedico mi tesis con todo mi amor para que te sientas orgullosa, para que veas que no te he defraudado y que toda la confianza que pusiste en mi no fue en vano. Todo lo que soy lo debo a ti. Te amo mamita linda.*

*Al hombre más importante de mi vida, mi padre, por ser todo para mí, porque ha sabido ser padre, amigo, compañero, guía y maestro. Por enseñarme a luchar por lo que quiero en la vida, por ayudarme ha levantarme siempre que he caído. Por enseñarme el camino correcto y ayudarme a transitar por él. Por la educación y los principios que siempre me ha inculcado. Porque ha estado siempre a mi lado en las buenas y malas.*

*A ti papito querido dedico mi tesis porque sé que me amas tanto como yo a ti, porque como hasta ahora, sé que siempre me apoyarás y me darás todo cuanto he necesitado. Porque todo lo que soy lo debo a ti. Porque eres una parte indispensable de mí y porque sería nada si no te tuviera. Te amo papi.*

*Glennis Tamayo Morales.*

# Resumen

Esta investigación pretende dar respuesta a una de las problemáticas existentes en los centros dedicados a la biotecnología. En estas instituciones se genera un gran cúmulo de información, obtenida de todos los procesos que se llevan a cabo, por lo que se han definido previamente modelos estándares para la recogida de datos, los cuales cambian y se actualizan con rapidez, si estos cambios no son controlados, el producto no llega a liberarse.

El presente trabajo tiene como objetivo desarrollar una herramienta web que permita el diseño y control de los modelos de recogida de datos. Para ello se realizó una investigación en busca de herramientas que facilitaran el desarrollo de la aplicación, encontrándose el editor web TinyMCE que fue empleado en la solución. Además, se identificaron las funcionalidades y se realizó el diseño e implementación del sistema. Esta aplicación permitirá el diseño y control de cambios de los modelos de recogida de datos, de forma sencilla, a personas con conocimientos mínimos en el uso de aplicaciones web. Para garantizar la portabilidad de la aplicación se desarrolló bajo ambiente multiplataforma, fundamentalmente con herramientas de software libre.

PALABRAS CLAVE

Biotecnología, Modelos de Recogida de Datos, TinyMCE.

**ÍNDICE**

**ÍNDICE DE FIGURAS**

# Introducción

La Ciencia y la Tecnología son actividades intelectuales a través de las cuales es posible llegar a un conocimiento aproximado de la realidad y sobre todo, lograr su transformación en la procuración del desarrollo humano. En la actualidad el uso de las tecnologías ha tenido un gran auge a nivel mundial, obteniéndose grandes logros en la rama de la biotecnología.

Cuba aún siendo un país del tercer mundo, en este aspecto no ha quedado atrás. Importantes logros científicos puede exhibir la biotecnología en Cuba en poco más de dos décadas. Algunos son productos exclusivos en el mundo, y otros, por su originalidad en la concepción y eficacia en los resultados, compiten con los de las grandes compañías transnacionales. Estos logros tuvieron lugar a partir de la creación en los años 90 de los Polos Científicos del país dedicados a la Biotecnología e Industria Farmacéutica.

Entre los centros más destacados que integran estos polos se encuentran el Centro de Ingeniería Genética y Biotecnología (CIGB), Centro Nacional de Biopreparados (BioCen), Instituto Finlay, Centro de Inmunología Molecular (CIM) e Instituto "Pedro Kourí", por mencionar alguno de ellos. Cada uno de estos guían su trabajo e investigación en la producción de fármacos y vacunas para el tratamiento y prevención de enfermedades tales como: el cáncer, la hepatitis, la meningococo, entre otras, con el fin de mejorar la calidad de vida de los seres humanos.

Actualmente se encuentran inmersos en la tarea de informatización, planificando la vinculación de sus actividades al empleo de las tecnologías de la informática y las comunicaciones (TICs). De esta forma se lograría una mejor gestión en cuanto al control, acceso y manejo de la información, de manera que aumente la producción y comercialización de productos biológicos.

En estas instituciones se genera un gran cúmulo de información, que se obtiene a través de todos los procesos que se llevan a cabo, para esto se han definido previamente modelos estándares para la recogida de datos los cuales cambian, se actualizan o se modifican a partir de investigaciones realizadas o nuevos diagnósticos; lo que trae consigo que en ocasiones se trabaje con modelos de recogidas de datos

desactualizados u obsoletos. Como consecuencia de esto, el producto no llega a liberarse debido a que la documentación no cumple con las normas o estándares requeridos.

Por todo lo antes expuesto se propone como **Problema Científico**: ¿Cómo contribuir a la mejora del proceso de control de cambios y escalabilidad de los modelos de recogida de datos para los centros dedicados a la biotecnología?
Se propone como **Objeto de Estudio:** Proceso de gestión de información de los centros biotecnológicos.
**Campo de Acción:** Proceso de diseño y control de modelos de recogida de datos en los centros biotecnológicos.

Se plantea como **Objetivo General:** Desarrollar una herramienta web para el diseño y control de modelos de recogida de datos en los centros biotecnológicos.

Se definen como **Objetivos Específicos**:
- Identificar las funcionalidades de la herramienta web para el diseño de modelos de recogida de datos.
- Realizar el diseño de la herramienta web para la construcción de modelos de recogida de datos.
- Realizar la implementación de la herramienta web para el diseño de modelos de recogida de datos.

Para dar cumplimiento a los objetivos específicos, se trazaron las siguientes **tareas**:
- Investigación de las diferentes herramientas existentes en el mundo para la construcción de modelos de recogida de datos.
- Investigación de las tendencias y tecnologías para desarrollar una herramienta web que facilite la construcción de modelos de recogida de datos.
- Confección del Modelo de Dominio.
- Identificación de requerimientos.
- Identificación de las clases del diseño.
- Elaboración del modelo de diseño.
- Diseño de la base de datos.
- Elaboración del diagrama de componentes.
- Implementación de los requisitos funcionales.
- Realización de pruebas funcionales.

Esta investigación posee la siguiente estructura:

**Capítulo I: "**Fundamentación teórica"**:** Se estudian las herramientas para el diseño de modelos de recogida de datos existentes a nivel mundial y en Cuba, del que se nutre la base teórico conceptual de esta investigación. Se exponen las características fundamentales de la metodología de desarrollo, tecnologías y herramientas, definidas por la arquitectura, que son utilizadas en la implementación de la herramienta. Además, se manifiestan las tendencias actuales de los roles que participan en el proceso de desarrollo de software**.**

**Capítulo II: "**Características del sistema": En este capítulo se realiza la descripción del sistema a desarrollar para lograr una mayor claridad y comprensión por parte de los desarrolladores. Los puntos que se abordan son los siguientes: descripción del modelo de dominio, especificación de los requisitos funcionales y no funcionales, representación de los actores y casos de uso a través de un diagrama de casos de uso del sistema, así como la descripción textual de ellos.

**Capítulo III: "**Diseño del sistema": El contenido que se aborda en este capítulo está relacionado con el diseño del sistema. El mismo incluye la descripción del estilo arquitectónico y patrones de diseño seleccionados, además de explicar cómo se adaptan y se representan cada uno de ellos en el diseño del sistema. Por otra parte, se realizan todos los diagramas de clases, de interacción y el diagrama de despliegue.

**Capítulo IV:** "Implementación y prueba": Se describe la implementación de los componentes a partir de los requerimientos identificados. Se presenta el diagrama de componentes correspondiente a cada uno de los casos de uso a implementar. Además, se muestran fragmentos relevantes del código y las vistas principales de la aplicación; así como ejemplos de las pruebas funcionales aplicadas, que permiten examinar las funcionalidades de la aplicación y garantizar la calidad del software.

# Capítulo 1: Fundamentación Teórica

Se estudian las herramientas para el diseño de modelos de recogida de datos existentes a nivel mundial y en Cuba, del que se nutre la base teórico conceptual de esta investigación. Se exponen las características fundamentales de la metodología de desarrollo, tecnologías y herramientas, definidas por la arquitectura, que son utilizadas en la implementación de la herramienta. Además, se manifiestan las tendencias actuales de los roles que participan en el proceso de desarrollo de software**.**

## 1.1 Desarrollo Biotecnológico en Cuba

El nivel de desarrollo de un país se mide, entre otras cosas, por los indicadores de salud de la población y dentro de estos, la producción de medicamentos es un parámetro importante para el progreso social.

Cuba ha emprendido un amplio programa para aumentar la producción de su industria farmacéutica y completar en breve plazo el suministro de los medicamentos más demandados, a la vez que busca incrementar las exportaciones. Para lograr esta meta, es de suma importancia, el poder contar con herramientas y tecnologías que permitan promover el aumento de su producción.

El proceso de desarrollo de estos fármacos genera un gran volumen de información que debe ser gestionada y la misma debe cumplir con las normas o estándares requeridos para que la producción y comercialización sea efectiva y se pueda llevar a cabo. Para ello, se definen previamente modelos estándares para la recogida de datos los cuales cambian, se actualizan o se modifican a partir de investigaciones realizadas o nuevos diagnósticos.

### 1.1.1 Modelo de Recogida de Datos

Un Modelo de Recogida de Datos no es más que una plantilla para almacenar datos, con un conjunto de variables definidas y una estructura estática que es estándar para el trabajo en la institución en la que se utilice; en ocasiones también llamados registros. Estos se pueden utilizar con diferentes propósitos, para llevar el control de las personas que radican en un centro laboral, para recoger datos de una investigación que necesitan ser archivados, depende del entorno en que se utilice.

En los centros dedicados a la biotecnología los modelos de recogida de datos tienen características peculiares, estos no tienen una estructura estática, la información puede cambiar con mucha rapidez y los datos que eran obligatorios registrar en un momento ya no lo son al paso del tiempo. Esto trae como consecuencia que se trabaje con modelos desactualizados y puede ser esta una razón para que el producto no llegue a liberarse.

### 1.1.2 Resultado de la investigación para el diseño de modelos de recogida de datos

Por todo lo explicado anteriormente se hace necesaria una herramienta que permita controlar los cambios y gestionar de forma correcta toda la información que se maneja en el proceso de producción de fármacos en los centros biotecnológicos. Para ello, se necesita que dicha herramienta proporcione la construcción de modelos de recogida de datos con estructura dinámica. A partir de aquí se desarrolla una profunda investigación sobre el tema, en la cual no se encontraron herramientas que cumplieran con las características antes expuestas, aunque resultaron diferentes editores web que pudieran ser muy útiles para el diseño de los modelos. El estudio se centró en sus principales características, ventajas, desventajas y facilidad de uso.

### 1.1.3 Editores web

Los editores web son aplicaciones diseñadas con el fin de facilitar la creación de documentos HTML o XHTML. Pueden ser de dos tipos, unos permiten codificar las páginas utilizando el propio lenguaje HTML, a base de etiquetas y otros permiten diseñar una página web como si se estuviera escribiendo con un editor de texto como Word. A estos últimos se les conoce como editores WYSIWYG, esta palabra proviene de las siglas What You See Is What You Get (traducido, "lo que ves es lo que obtienes"), referido a cualquier tecnología que permita ver las imágenes en pantalla como serán reproducidas en la web o cualquier otro medio.

Los editores WYSIWYG permiten de manera visual colocar distintos elementos sobre una vista previa de la página, encargándose el programa de generar el documento HTML. La manera de trabajar en este tipo de editores es muy similar a la que se usa cuando se emplea un procesador de texto. Pues el usuario no tiene necesidad de conocer las etiquetas del lenguaje de marcado. En lugar de esto, escribe el texto, lo formatea,

e inserta las imágenes en los lugares deseados. Posteriormente el editor transforma la vista por pantalla en código HTML perfectamente configurado.

## TinyMCE

Es uno de los editores WYSIWYG más conocidos, que proporciona muchas de las funcionalidades que se logran con un procesador de texto (cortar, pegar, alinear el texto, cambiar la fuente). Entre sus características se destaca su facilidad de integración en las páginas web, pues cuenta con pocas líneas de código y con una amplia documentación que facilita su uso; además se puede personalizar a través de plugins, se puede utilizar AJAX para guardar y cargar el contenido. TinyMCE es compatible con la mayoría de los navegadores y es soportado por plataformas tales como Windows, Linux, Solaris, Freebsd, Osx. Está desarrollado completamente en JavaScript, incluyendo el uso de CSS. Este editor incluye soporte multilenguaje, utilizando paquetes de idioma y fue liberado bajo licencia LGPL.

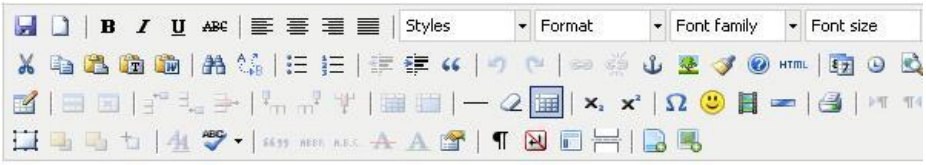


**Figura1. Editor TinyMCE**

## FCKeditor

Se basa en un núcleo escrito en JavaScript, pero también es implementado con tecnologías del lado del servidor. Dentro de sus características cuenta con generación de código HTML, soporte CSS e incorpora formularios; además de funcionalidades básicas de los editores de escritorio conocidos como Word, tales como: cortar, copiar y pegar, inserción de imágenes y creación de tablas. Por las características que este editor presenta no es necesario tener conocimientos de HTML pues es muy fácil su aprendizaje. En el 2009, se decidió cambiar el nombre del editor, con su próxima generación: CKEditor.


**Figura 2. Editor FCKeditor**

### CKEditor

CKEditor hereda la calidad y las características fuertes de FCKeditor, mejorando sus puntos débiles como la velocidad de carga, la complejidad y el rendimiento. Es accesible tanto a la interfaz de usuario como en el código HTML que genera. Este editor es compatible con los navegadores más utilizados, se puede adaptar, estilizar y extender mediante plugins. CKEditor está licenciado bajo código abierto y flexible de licencias comerciales, por lo que se puede integrar y utilizar en el interior de cualquier tipo de aplicación.


**Figura 3. Editor CKEditor**

El determinar un editor de texto para dar apoyo al desarrollo de la herramienta de diseño para modelos de recogida de datos se convierte en una tarea difícil, pues todos los estudiados tienen características similares que se ajustan a las expectativas de los desarrolladores; aunque en cualquiera de los casos se necesita realizar modificaciones, por lo que se decide utilizar el editor más sencillo en cuanto a la arquitectura de su implementación, facilidad de integración, simplicidad en el código y que el trabajo con las tablas sea más completo.

Después de realizar un estudio previo referente a los editores web existentes, y guiados para la elección en lo explicado anteriormente se decide utilizar el editor TinyMCE en su versión 3.3, el cual será sometido a cambios.

## 1.2 Metodología de desarrollo de software

El desarrollo de software sin dudas no es una tarea fácil. Como resultado a este problema ha surgido una alternativa desde hace mucho: la metodología. Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Estas imponen un proceso disciplinado con el fin de lograr un desarrollo más predecible y eficiente.

### 1.2.1 Proceso Unificado de Desarrollo (RUP por sus siglas en inglés, Rational Unified Process).

Esta es una de las metodologías más generales que existe. Está enfocada a cualquier tipo de proyecto así no sea de software. En ella se definen un conjunto de actividades que dan soporte al proceso de desarrollo de software, que han sido concentradas en grupos lógicos, definiéndose 9 flujos de trabajo principales. Esta metodología se basa además en la documentación generada por sus cuatro fases (inicio, elaboración, construcción y transición).

RUP se caracteriza por estar dirigido por casos de uso. Estos reflejan las necesidades de los usuarios que son captadas en el comienzo del primero de sus flujos de trabajo. A partir de aquí los mismos guían todo el proceso de desarrollo de software, ya que, cada uno de sus flujos de trabajo arroja modelos que representan la realización de los casos de uso.

Esta metodología es iterativa e incremental. Cada una de sus fases se desarrolla en iteraciones, una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Este proceso se centra en la arquitectura, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura, la misma muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

Elegir una metodología depende principalmente de dos factores, el tipo de proyecto y la cultura que exista en el entorno de desarrollo. No hay metodología que funcione de manera universal, de hecho, cada vez más las metodologías se conciben como 'marcos' metodológicos que son necesarios ajustar para cada organización y tipo de proyecto.

Dado que RUP es una metodología que provee una forma disciplinada de asignar tareas y responsabilidades, adaptable a las necesidades del proyecto y pretende implementar las mejores prácticas en la ingeniería de software; siendo capaz de arrojar como resultado un trabajo bien documentado, posibilitando que cualquier otro equipo de desarrolladores pueda continuar con próximas versiones de la

herramienta, además por ser la más conocida y utilizada en la institución a la cual pertenece el equipo de desarrollo, se define entonces esta como la metodología de desarrollo de software a utilizar.

## Roles y Artefactos

Un rol es una definición abstracta de un conjunto de responsabilidades para las actividades que deben ser realizadas y para los artefactos que se producirán. Es realizado por un individuo, o un conjunto de individuos, que trabajan juntos en el mismo equipo. Un miembro del equipo de proyecto satisface típicamente diversos roles.

En RUP se definen los artefactos como productos tangibles del proyecto que son producidos, modificados y usados. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

RUP propone un conjunto de roles, agrupados por participación en tareas relacionadas, estos son: Analistas, Desarrolladores, Gestores, Apoyos, Especialista en Prueba y otros Roles Adicionales. Cada uno de estos es responsable de la realización de un grupo de artefactos en correspondencia con los flujos de trabajo.
De acuerdo con las necesidades y características de la investigación los roles que se desarrollarán se muestran a continuación:

**Roles y artefactos:**

| Flujo de trabajo | Roles | Artefactos |
|---|---|---|
| Requerimientos | **Analista del sistema:** dirige y coordina la adquisición de requisitos esquematizando la funcionalidad del sistema y delimitándolo. | Definición de los Requisitos funcionales y Requisitos no funcionales.<br>Actor del sistema.<br>Caso de uso del sistema.<br>Diagrama de casos de uso del sistema. |
| | **Especificador de Requerimientos:** especifica y detalla los requisitos del sistema. | Descripciones textuales de los casos de uso. |

| | | |
|---|---|---|
| Diseño | **Diseñador:** dirige el diseño de una parte del sistema, dentro de las restricciones de los requisitos, arquitectura y proceso de desarrollo para el proyecto. | Realización de casos de uso del Diseño: diagramas de clases del diseño y diagramas de interacción.<br>Subsistemas del Diseño.<br>Paquetes del Diseño. |
| | **Diseñador de BD:** dirige el diseño de la estructura de almacenamiento de datos persistentes que se utilizará en el sistema. | Modelo de Datos. |
| | **Arquitecto:** 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. | Modelo de despliegue. |
| Implementación | **Implementador:** desarrolla los componentes de software y efectúa las pruebas de desarrollador para la integración en subsistemas más grandes, de acuerdo con los estándares adoptados de proyecto. | Implementación de componentes. |
| | **Arquitecto** | Diagrama de componentes. |

**Tabla 1. Descripción de los roles y artefactos de cada flujo de trabajo de RUP.**

### Lenguaje de Modelado

RUP utiliza como lenguaje de modelado: Lenguaje Unificado de Modelado (Unified Modeling Language).
*"UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema software".* (1)

UML permite la modelación de sistemas con tecnología orientada a objetos. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, debido a que es un lenguaje, cuenta con reglas para combinar tales elementos. La finalidad de estos diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo.

## 1.3 Herramientas y tecnologías actuales

### 1.3.1 Herramientas CASE

*''Se pueden definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de desarrollo de un software''.* (2)

### Visual Paradigm 3.4

Herramienta UML profesional que soporta el desarrollo de software desde el análisis y diseño orientado a objeto, construcción, pruebas y despliegue. Entre muchas de sus ventajas permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, permite la importación y exportación de archivos XML de diferentes versiones. Presenta licencia gratuita y comercial. Es multiplataforma, fácil de instalar y actualizar, compatible entre ediciones.

### 1.3.2 Lenguaje de Programación

*"Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo*". (3)

### PHP 5

Es un lenguaje de programación del lado del servidor, gratuito e independiente de la plataforma, está preparado para realizar diversos tipos de aplicaciones web gracias a las extensas bibliotecas de funciones por la que está dotado, se puede integrar con la mayoría de los servidores web; además está diseñado para soportar características de la programación orientada a objetos. Este lenguaje tiene una comunidad muy grande de desarrolladores, lo que permite encontrar con facilidad documentación, tutoriales y ejemplos de código fuente, facilitando así su aprendizaje.

PHP está disponible para la mayoría de sistemas operativos existentes desde Linux hasta Windows. Además, soporta una gran variedad de Sistemas Gestores de Base de Datos incluyendo PostgreSQL.

### JavaScript

Es un lenguaje de programación del lado del cliente, ya que es el navegador el que soporta la carga de procesamiento, permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Es interpretado, no requiere compilación y es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera y Mozilla Firefox, por lo que es el lenguaje de programación del lado del cliente más utilizado.

JavaScript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web, para que este pueda acceder a ellos y modificarlos dinámicamente.

## 1.3.3 Servidor web

*Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada)*

*Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:*

1. *Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).*
2. *Recibe una petición.*
3. *Busca el recurso.*
4. *Envía el recurso utilizando la misma conexión por la que recibió petición.*
5. *Vuelve al segundo punto.* (4)

### Apache 2.0

Apache es un servidor web multiplataforma, rápido, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación y publicación de documentos php con una estabilidad y eficacia ampliamente comprobada. Está estructurado en módulos que se pueden clasificar en tres categorías: Módulo Base, Módulo Multiproceso y Módulos Adicionales. En el Módulo Base se encuentran las funcionalidades más elementales, siendo necesario un Módulo Multiproceso para manejar las peticiones. El resto de las funcionalidades del servidor se consiguen por medio de Módulos Adicionales, por lo que puede ser adaptado a diferentes entornos y necesidades. Es una tecnología gratuita de código fuente. Trabaja con PHP y otros lenguajes script, además de Java y páginas jsp. Incluye todo el soporte que se necesita para tener páginas dinámicas.

## 1.3.4 Framework de desarrollo

En el contexto del desarrollo de software, un framework es una estructura de archivos y utilidades que aceleran la programación de una aplicación informática, proporcionando una metodología de trabajo que sistematiza y facilita la generación de formularios, funciones y módulos de uso común, permitiendo al desarrollador dedicar su atención hacia los aspectos específicos de cada aplicación.

### Symfony 1.2

Es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web, pues posee una librería de clases que permite reducir el tiempo de desarrollo. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web (incorpora el patrón Modelo Vista Controlador MVC). Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Está desarrollado en PHP5, se puede utilizar en plataformas *nix (Unix, Linux) y Windows. Soporta AJAX, es compatible con la mayoría de Gestores de Bases de Datos,

como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Es extensible, lo que permite su integración con bibliotecas desarrolladas por otros fabricantes como es el caso de Dojo Toolkit y Propel.

### Dojo Toolkit

Dojo Toolkit es un framework JavaScript que facilita el desarrollo de aplicaciones web enriquecidas en el cliente y que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, pues sólo con el nombre de la clase es capaz de localizar e incluir el archivo a utilizar.

Con el uso de este framework se pueden obtener aplicaciones web lo más semejante posible a una aplicación desktop. Es compatible con múltiples navegadores y código abierto. Cuenta con una variedad de complementos que permiten el enriquecimiento de la web, algunos de ellos son: menús, pestañas, tooltips, tablas ordenables, gráficos dinámicos, efectos de animación, soporte para arrastrar y soltar, formularios y rutinas de validación para los parámetros. Además, calendario, selector de tiempo y reloj, entre otros.

## 1.3.5 AJAX (Asynchronous JavaScript and XML)

"*AJAX no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes".* (5)

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas son ejecutadas en el navegador de los usuarios mejorando completamente la interacción de este con la aplicación. De esta forma, es posible realizar cambios sobre las páginas evitando las recargas constantes, ya que el intercambio de información con el servidor se produce en un segundo plano, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Para crear aplicaciones interactivas con AJAX, se combinan varias tecnologías como son: XHTML y CSS para crear una presentación basada en estándares, DOM para la interacción y manipulación dinámica de la presentación, XML y XSLT para el intercambio y la manipulación de información, XMLHttpRequest, para el intercambio asíncrono de información y JavaScript para unir dichas tecnologías.

### 1.3.6 Entorno de desarrollo

Un entorno de desarrollo integrado o IDE (Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o puede utilizarse para varios.

#### NetBeans IDE 6.8 Beta

NetBeans IDE 6.8 Beta es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Contiene las herramientas para que los desarrolladores de software puedan crear aplicaciones desktop, enterprise, web, y aplicaciones móviles, con el lenguaje Java, así como también C/C++, PHP, JavaScript entre otros.

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 subversion que desde el entorno permite manipular las versiones del código durante la implementación. Es un producto libre y gratuito, sin restricciones de uso, tanto comercial como no comercial.

### 1.3.7 Sistemas Gestores de Base de Datos

Un Sistema Gestor o Manejador de Bases de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una base de datos (BD), por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones.

#### PostgreSQL 8.3

Es un potente sistema de base de datos relacional libre. Funciona en sistemas operativos como: Linux, Unix, Mac OS, Windows entre otros. Presenta todas las características de una base de datos profesional (triggers, procedimientos almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas). Incluye la mayoría de los tipos de datos como son INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE e INTERVAL así como almacenaje especial para tipos de datos grandes. Soporta los lenguajes de programación más conocidos: PHP, C, C++, Java, Perl, Python, entre otros. Es altamente

adaptable a las necesidades del cliente. Dispone de una documentación bien organizada, pública, con comentarios de los propios usuarios y comunidades muy activas.

## Conclusiones parciales del capítulo

➢ En la investigación realizada sobre herramientas para la construcción de modelos de recogida de datos, no se encontró ninguna herramienta que cumpla con las características necesarias para dar respuesta al problema planteado, aunque resultaron diferentes editores web que podrían facilitar el desarrollo de la aplicación, seleccionando TinyMCE como el editor a utilizar para dar apoyo en la construcción de la herramienta, el cual será sometido a cambios.

➢ Del estudio de las herramientas y tecnologías a emplear en la solución se define RUP como metodología de desarrollo de software, Visual Paradigm 3.4 como herramienta CASE y UML como lenguaje de modelado. El lenguaje de programación a utilizar será PHP5 y JavaScript para la validación, con NetBeans como IDE, Symfony 1.2 como framework de desarrollo y Dojo Toolkit como framework para el diseño. Como Servidor web Apache 2.0 y el Sistema Gestor de Base de Datos PostgreSQL 8.3.

# Capítulo 2: Características del Sistema

En este capítulo se realiza 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 a través de un diagrama de casos de uso del sistema, así como la descripción textual de ellos.

## 2.1 Objeto de estudio

### 2.1.1 Situación Problémica

El desarrollo de la biotecnología e industria farmacéutica de un país es una tarea fundamental para lograr el avance de la sociedad. En Cuba existen centros que guían su trabajo e investigación en la producción de fármacos y vacunas para el tratamiento y prevención de enfermedades, procurando así, el bienestar de sus habitantes. En dichos centros se generan grandes volúmenes de información resultantes de todos los procesos que allí se desarrollan, toda esta información debe ser controlada y recogida en modelos o registros los cuales cambian, se actualizan o se modifican con el paso del tiempo, a partir de investigaciones o nuevos diagnósticos.

Estos centros están compuestos por varios laboratorios y cada uno de ellos trabaja con los mismos modelos, pero recogiendo diferente información, en caso de que un modelo sufra cambios será necesario ir a cada laboratorio a informar los mismos. Por lo que se hace necesario desarrollar una herramienta que permita al usuario la construcción de sus propios modelos de recogida de datos.

### 2.1.2 Objeto de Automatización

- **Proceso de diseño y control de cambios de los modelos de recogida de datos.**

Este es uno de los procesos más importantes de estas instituciones, pues para liberar un producto es necesario que los modelos cumplan con las normas o estándares establecidos. Los usuarios autorizados definen los datos que se deben registrar, lo informan y distribuyen a cada uno de los laboratorios y en caso de modificaciones debe comenzar nuevamente el proceso.

### 2.1.3 Propuesta del sistema

Se propone la implementación de una herramienta web que le permitirá a los usuarios crear de forma dinámica modelos de recogida de datos, para ello la herramienta contará con un área de diseño donde se podrá definir el formato y la estructura de los modelos, además una vez creado el modelo pueden diseñarse varias versiones sin necesidad de cambiar la versión anterior, con el objetivo de mantener los cambios controlados.

## 2.2 Descripción del Modelo de Dominio

El modelo de dominio es una representación visual estática del entorno real objeto del proyecto. Es un diagrama con los objetos que existen (reales) relacionados con el proyecto que se va a desarrollar y las relaciones que hay entre ellos. Este modelo ayuda a comprender los conceptos que utilizan los usuarios, es decir, los conceptos con los que trabajan y con los que deberá trabajar la aplicación.

El modelo de dominio se describe mediante diagramas de UML, especialmente mediante diagramas de clases. En la figura 4 se representa el modelo de dominio realizado.

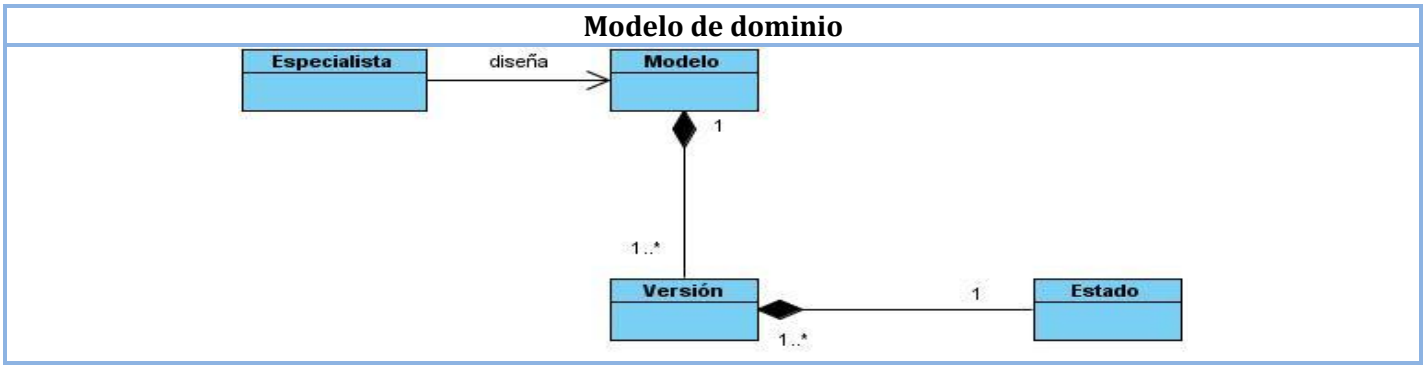


**Figura 4. Modelo de dominio.**

### 2.2.1 Definición de clases del Modelo de Dominio

1. **Especialista.**

Persona encargada de diseñar los modelos de recogida de datos, con una estructura y un formato previamente establecidos.

2. **Modelo.**

Planilla con un estándar o formato previamente definido, que contiene un conjunto de variables y campos que pueden cambiar o actualizarse a partir de nuevas investigaciones.

3. **Versión.**

Es la modificación de un modelo, realizada a partir de resultados obtenidos de nuevas investigaciones.

4. **Estado.**

Estado o fase en la que se puede encontrar una versión (activo, edición, obsoleto).

### Reglas del negocio

Las reglas de negocio no son más que las restricciones que existen, es decir, las acciones no válidas que la aplicación debe controlar para que el negocio no colapse.

1. Un Modelo de Recogida de Datos (MRD) sólo podrá ser eliminado si no tiene alguna versión asociada.
2. Una versión sólo tomará los estados de edición, activo y obsoleto.
3. Una versión puede ser eliminada sólo si se encuentra en estado de edición.

## 2.3 Definición de los Requisitos Funcionales

"…*los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir*". (6) Deben ser especificados por escrito, posibles de probar o verificar, descritos como una característica del sistema a entregar y lo más abstracto y conciso posible. Estos se mantienen invariables sin importar con que propiedades o cualidades se relacionen.
La herramienta a desarrollar debe cumplir con los siguientes requisitos funcionales:

RF1 Crear MRD.
RF2 Buscar y visualizar MRD.
RF3 Modificar MRD.
RF4 Actualizar listado de MRD.
RF5 Eliminar MRD.

RF6 Crear versión.

RF7 Eliminar versión.

RF8 Buscar y visualizar versión.

RF9 Actualizar listado de la versión.

RF10 Visualizar versión editada.

RF11 Insertar elementos a la versión.

RF12 Modificar propiedades de los elementos de la versión.

RF13 Eliminar elementos a la versión.

RF14 Modificar estado de la versión.

## 2.4 Definición de los Requisitos no Funcionales

*"Los requerimientos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencia de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad".* (7) Estos forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto.

Existen múltiples categorías para clasificar a los requerimientos no funcionales, aquí se presentan las identificadas para la herramienta a desarrollar, aunque no limitan la definición de otras.

**RNF1. 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 (host) tenga instalado PostgreSQL 8.3, Apache 2.0 y PHP 5.

**RNF2. Hardware**

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

Los requerimientos recomendados para un adecuado funcionamiento de la aplicación son:

**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.

**RNF3. Restricciones en el 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 y Dojo ToolKit.

**RNF4. Apariencia o interfaz externa**

La aplicación contará con una interfaz sencilla, con colores suaves, lo más semejante a una aplicación de escritorio.

**RNF5. Seguridad**

El sistema una vez integrado estará compuesto por 3 módulos que para acceder a cada uno de ellos se contará con una autenticación previa quedando limitado su acceso con la presencia de los diferentes roles.

**Confidencialidad:** Existencia de distintos roles que establezcan que la información sólo 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.

**RNF6. Usabilidad**

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

**RNF7. 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. Además, se aplicará pruebas de estrés durante el primer año de explotación.

**RNF8. 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**.**

**RNF9. 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.

**RNF10. Portabilidad**

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

**RNF11. 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.

## 2.5 Modelo de Caso de Uso del Sistema

El modelo de caso de uso del sistema contiene actores, casos de uso y sus relaciones. Describe lo que hace el sistema para cada tipo de usuario, representados por uno o varios actores, los cuales representan a individuos o sistemas externos que colaboran con este.

Para el desarrollo de esta herramienta se ha identificado el siguiente actor.

| Actor | Descripción |
|---|---|
| Diseñador | Es el encargado de gestionar el diseño de los modelos de recogida de datos. |

### 2.5.1 Patrones de Caso de Uso

Entre los diferentes patrones de caso de uso se encuentra el CRUD Completo, que propone formar un caso de uso a partir de los requisitos funcionales relacionados con las acciones de insertar, listar o mostrar, modificar, y eliminar una determinada información. Haciendo uso de este, las funcionalidades identificadas para el desarrollo de la aplicación quedaron agrupadas en 3 casos de uso: Gestionar MRD, Administrar Versión y Editar Versión.

### 2.5.2 Diagrama de Casos de Uso del Sistema

El diagrama de caso de uso del sistema quedó representado por un actor y tres casos de uso, como se muestra a continuación:

**Diagrama de Casos de Uso del Sistema**

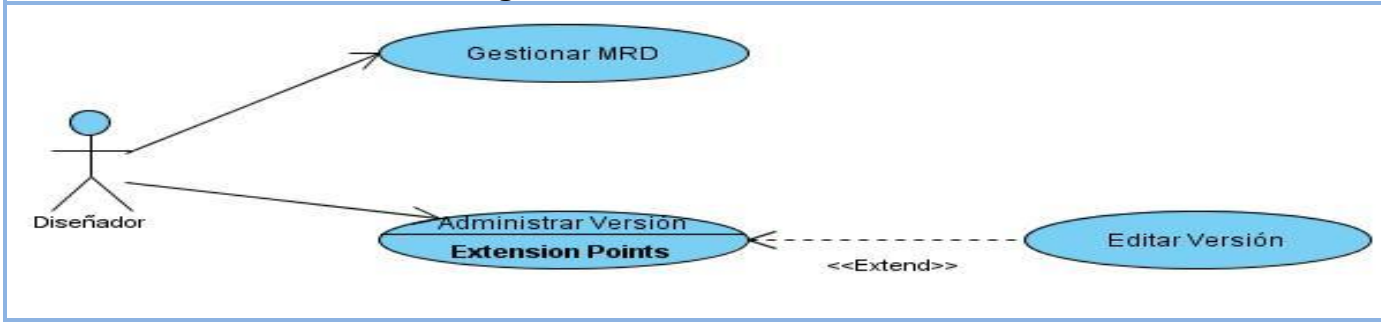


**Figura 5. Diagrama de Casos de Uso del Sistema.**

### 2.5.3 Descripciones textuales de los Casos de Uso del Sistema

**CUS-1 Gestionar MRD.**

| | |
|---|---|
| **Caso de Uso:** | **Gestionar MRD** |
| **Actores:** | **Diseñador(inicia)** |
| **Resumen:** | El caso de uso (CU) se inicia cuando el diseñador va a realizar algunas de las siguientes operaciones:<br>• **Crear MRD:** cuando el diseñador desea crear un MRD, este introduce los datos necesarios para su construcción y el sistema registra los mismos, finalizando así el CU.<br>• **Buscar y Visualizar MRD:** el diseñador selecciona los datos por los cuales desea realizar la búsqueda**,** el sistema muestra los resultados para dicha búsqueda, finalizando así el CU.<br>• **Modificar MRD:** el diseñador selecciona el modelo a modificar, el sistema muestra los datos correspondientes, el diseñador los modifica, finalizando así el caso de uso.<br>• **Actualizar listado de MRD:** el diseñador selecciona la opción actualizar, el sistema actualiza el listado de los modelos, finalizando el CU.<br>• **Eliminar MRD:** el diseñador selecciona el modelo e indica eliminar, el sistema lo elimina, finalizando así el CU. |
| **Precondiciones:** | El diseñador deberá estar autenticado en el sistema. |
| **Referencias** | **RF1, RF2, RF3, RF4 , RF5** |
| **Prioridad** | **Crítico** |

| **Curso Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Desea gestionar los MRD.<br><br>3.El diseñador selecciona la acción que desea realizar: | 2. El sistema muestra el listado de todos los modelos existentes.<br>4. El sistema en dependencia de la acción solicitada muestra la interfaz correspondiente: |

| | |
|---|---|
| a. Crear MRD.<br>b. Actualizar listado MRD.<br>c. Buscar y Visualizar MRD.<br>d. Modificar MRD.<br>e. Eliminar MRD. | a. Crear MRD: ir a la **sección: "Crear MRD".**<br>b. Actualizar listado MRD: ir a la **sección: "Actualizar listado MRD".**<br>c. Buscar y Visualizar MRD: ir a la sección**: "Buscar y Visualizar MRD".**<br>d. Modificar MRD: ir a la **sección: "Modificar MRD".**<br>e. Eliminar MRD: ir a la **sección: "Eliminar MRD".** |
| **Sección "Crear MRD". Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El diseñador selecciona la opción crear un modelo. | 2. El sistema muestra la interfaz para crear el MRD. |
| 3. El diseñador introduce los datos:<br>• Nombre del modelo (requerido).<br>• Título del modelo.<br>• Descripción. | 4. El sistema verifica que los campos requeridos hayan sido completados. |
| | 5. El sistema verifica que no exista un modelo con el mismo nombre. |
| | 6. El sistema genera un código único al modelo a crear. |
| | 7. El sistema registra y visualiza el nuevo modelo creado. |
| | 8. El sistema da la opción de:<br>Caso1: Crear otro modelo.<br>Caso2: Actualizar listado de modelo. |
| 9. El diseñador selecciona la opción crear otro modelo. | 10. El sistema va a la acción 2 de esta sección. |
| **Sección "Crear MRD". Flujo Alterno 1** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | 4.1 Si el campo nombre quedó vacío el sistema emite un mensaje: "Este campo es requerido, por favor complételo". |

| | |
|---|---|
| | 5.1 Si el modelo ya existe el sistema emite un mensaje: "Ya existe una entrada con este nombre ". |
| 9.1 El diseñador selecciona la opción de actualizar el listado de MRD. | 9.2 El sistema ejecuta la **sección "Actualizar listado de MRD".** |
| **Sección "Crear MRD". Flujo Alterno 2** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 9.1.1 Si el diseñador no desea realizar otra operación finaliza el CU. | |
| **Sección "Actualizar listado de MRD". Flujo Normal de Evento** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El diseñador selecciona la opción actualizar listado. | 2. El sistema busca los MRD existentes. |
| | 3. El sistema actualiza el listado y visualiza todos los modelos existentes finalizando así el CU. |
| **Sección "Buscar y Visualizar MRD". Flujo Normal de Evento** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El diseñador selecciona la opción buscar. | 2. El sistema muestra un formulario con los siguientes campos.<br>• Variable (código, nombre, título, descripción)<br>• Criterio.<br>• Valor. |
| 3. El diseñador selecciona la variable por la cual desea realizar la búsqueda, adiciona un criterio, introduce un valor y presiona el botón buscar. | 4. El sistema valida el campo valor en dependencia de la variable seleccionada. |
| | 5. El sistema muestra el(los) modelo(s) que cumpla(n) con estas condiciones y lo(s) visualiza. |
| | 6. El sistema da la opción de realizar otra búsqueda. |

| | |
|---|---|
| 7. Si el diseñador desea realizar otra búsqueda va a la acción 1 de esta sección. | |
| **Sección "Buscar y Visualizar MRD". Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | 4.1 El sistema muestra el mensaje "El valor especificado no es válido". |
| | 5.1 Si no existe un modelo que cumpla con estas condiciones el sistema emite un mensaje: "No se encontraron elementos". |
| 7.1 Si el diseñador no desea realizar otra búsqueda finaliza el caso de uso. | |
| **Sección "Modificar MRD". Flujo Normal de Eventos.** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El diseñador selecciona la opción modificar | 2. El sistema muestra el listado de los MRD existentes. |
| | 3. Da las siguientes opciones:<br>• Caso1: seleccionar un MRD del listado.<br>• Caso2: buscar y visualizar el MRD a modificar. |
| 4. El diseñador selecciona un MRD del listado para ser modificado. | 5. El sistema muestra un formulario con los datos del modelo seleccionado. |
| 6. El diseñador modifica los campos :<br>• Nombre del modelo.<br>• Título del modelo.<br>• Descripción. | 7. El sistema verifica que los campos requeridos hayan sido completados. |
| | 8. El sistema verifica que no exista un modelo con el mismo nombre. |
| | 9. El sistema actualiza el modelo con los nuevos datos y los visualiza. |
| | 10. El sistema brinda la opción de modificar otro modelo. |

| | |
|---|---|
| 11. El diseñador escoge la opción de realizar otra modificación y va a la acción 1 de esta sección. | 12. El sistema brinda la opción de actualizar el listado de modelos. |
| 13. El diseñador selecciona la opción de actualizar. | 14. El sistema ejecuta la **sección "Actualizar listado de MRD"** y finaliza el CU. |
| **Sección "Modificar MRD". Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 4.1 El diseñador selecciona buscar y visualizar el MRD. | 4.2 El sistema ejecuta la sección: **"Buscar y visualizar MRD".** |
| | 4.3 El sistema va a la acción 3 del flujo normal de eventos. |
| | 7.1 El sistema muestra un mensaje" Este campo es requerido, por favor complételo". |
| | 8.1 Si el modelo ya existe el sistema emite un mensaje: "Ya existe una entrada con este nombre ". |
| 11.1 El diseñador no desea realizar otra modificación. | 11.2 El sistema va a la acción 12 del flujo normal de eventos. |
| 13.1 Si el diseñador no desea actualizar el listado, finaliza el caso de uso. | |
| **Sección "Eliminar MRD". Flujo Normal de Eventos.** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El diseñador selecciona la opción eliminar. | 2. El sistema muestra el listado de los MRD existentes. |
| | 3. Da las siguientes opciones:<br>• Caso1: seleccionar un MRD del listado.<br>• Caso2: buscar y visualizar el MRD a eliminar. |
| 4. Selecciona un MRD del listado para ser eliminado. | 5. El sistema solicita la confirmación de la operación indicada. |
| 6. El diseñador confirma que desea | 7. El sistema verifica que el modelo seleccionado no tenga |

| | |
|---|---|
| eliminar. | versiones. |
| | 8. El sistema elimina el MRD, actualiza y visualiza el listado de MRD. |
| | 9. El sistema brinda la posibilidad de eliminar otro MRD. |
| 10. El diseñador decide eliminar otro MRD y va a la acción 1 de esta sección. | |
| **Sección "Eliminar MRD". Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 4.1 El diseñador selecciona buscar y visualizar el MRD. | 4.2 El sistema ejecuta la sección: **"Buscar y visualizar MRD".** |
| | 4.3 El sistema va a la acción 3 del flujo normal de eventos. |
| 6.1 El diseñador indica que no desea eliminar el MRD y finaliza el CU. | 7.1 Si el modelo seleccionado tiene versiones el sistema muestra un mensaje: "Este modelo no puede ser eliminado" y va a la acción 9. |
| 10.1 Si el diseñador decide no realizar otra eliminación, finaliza el CU. | |
| **Poscondiciones** | En dependencia de las acciones del diseñador:<br>• Se registra un nuevo modelo.<br>• Se modifica un modelo.<br>• Se elimina un modelo. |

***El resto de las descripciones textuales se pueden encontrar en: Documentos Complementarios.***

## Conclusiones parciales de capítulo

- Se identificaron 11 requisitos no funcionales y 14 funcionales, con los que se logró definir lo que el sistema debe hacer, permitiendo establecer un convenio entre desarrolladores y clientes de las características funcionales del software.

- Se identificó 1 actor y 3 casos de uso (Gestionar MRD, Administrar Versión y Editar Versión), quedando representado en diagrama de casos de uso del sistema la relación entre ellos.

# Capítulo 3: Diseño del Sistema

El contenido que se aborda en este capítulo está relacionado con el diseño del sistema. El mismo incluye la descripción del estilo arquitectónico y patrones de diseño seleccionados, además de explicar cómo se adaptan y se representan cada uno de ellos en el diseño del sistema. Por otra parte, se realizan todos los diagramas de clases, de interacción y el diagrama de despliegue.

## 3.1 Arquitectura del sistema

La arquitectura del software constituye una especificación de las principales ideas del diseño proporcionando una descripción más detallada de cómo realizar dicho sistema. Siguiendo la metodología utilizada, la arquitectura queda representada por las 4+1 vista entre las que se encuentra la Vista Lógica y la Vista de Despliegue.

La Vista Lógica contiene las clases del diseño más importantes organizadas por paquetes y subsistemas en capas de trabajo. También es representada por uno o varios diagramas de clases que son un subconjunto del modelo de diseño.

La Vista de Despliegue suministra una base para la comprensión de la distribución física del sistema a través de nodos. Describe la topología del sistema, es decir, la estructura de los elementos de hardware y de software que ejecuta cada uno de sus nodos.

### 3.1.1 Estilo arquitectónico utilizado

*"Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales".* (8)

Existen numerosos estilos arquitectónicos entre los que se encuentra el "Estilo de llamada y retorno" y dentro de él, el patrón de arquitectura Modelo-Vista Controlador (MVC). Symfony como framework de desarrollo que se emplea, toma lo mejor de la arquitectura MVC e implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo representándolo a través de tres elementos fundamentales: el modelo, la

vista y el controlador. Este patrón de arquitectura de software separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

### 3.1.2 Descripción del Patrón (MVC)

El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio. A continuación, se describe como se aprecia esto con el framework Symfony.

**Modelo:** localiza el código de acceso a datos, dichas clases se generan de forma automática en dependencia de la estructura de la base de datos, la librería Propel se encarga de esta generación automática y crea la estructura básica de las clases, generando el código necesario.

**Vista:** se encuentran los elementos cuyas responsabilidades están asociadas a la presentación de los datos, aquí está ubicado el layout que contiene la información visual común a todas las páginas. Además, se encuentra el código asociado a las plantillas, que se encarga de visualizar la información que el controlador devuelve.

**Controlador:** se encuentran las acciones que constituyen el núcleo principal de la aplicación, puesto que contiene toda la lógica de la aplicación. Las acciones utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, la URL define acción y parámetros de la petición.

Con el uso de este patrón se persigue mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos.

### 3.1.3 Descripción de los patrones de diseño utilizados

*"El patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas."* (9)

Los patrones solucionan problemas que existen en varios niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. Con el uso de patrones de diseño se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Permite formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

En el diseño de la aplicación se emplearon algunos de los patrones GRASP, relacionados a continuación:

**Patrón Experto:**

*"Asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad".* (10) Es decir, la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

Symfony dentro de su arquitectura tiene definido generar 4 clases por cada tabla de la BD, por ejemplo en la aplicación se tiene una tabla denominada **Model**, a partir de esta se crean las siguientes clases: **Model**, **BaseModel**, **ModelPeer** y **BaseModelPeer**. La clase **BaseModelPeer** es la encargada de hacer las consultas a la BD utilizando Propel, pues tiene los atributos necesarios para realizar dicha función, por tanto, debe implementar la responsabilidad de realizar las acciones directamente con la BD, evidenciándose así el uso del patrón Experto. Esto queda representado en la siguiente figura.

**Fragmento del diagrama de clases del diseño (CU Gestionar MRD)**

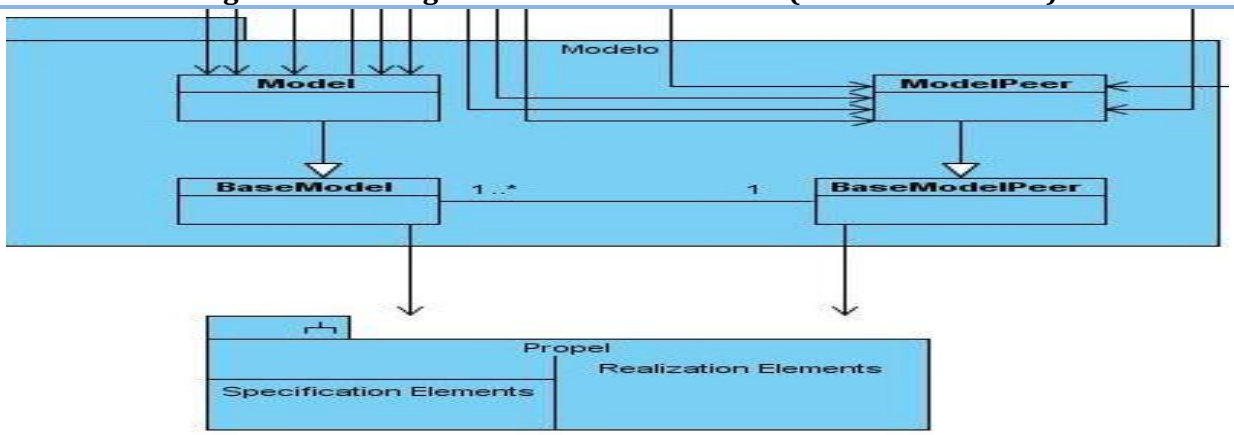


**Figura 6. Fragmento del diagrama de clases del diseño (CU Gestionar MRD) donde se evidencia la utilización del patrón GRASP: Experto.**

**Patrón Creador:**

*"El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento''.* (10)

En el diseño de la aplicación se asigna la responsabilidad de crear un modelo a la clase **newAction** como se muestra a continuación:

**Fragmento del diagrama de secuencia (CU Gestionar MRD: Escenario Crear MRD)**

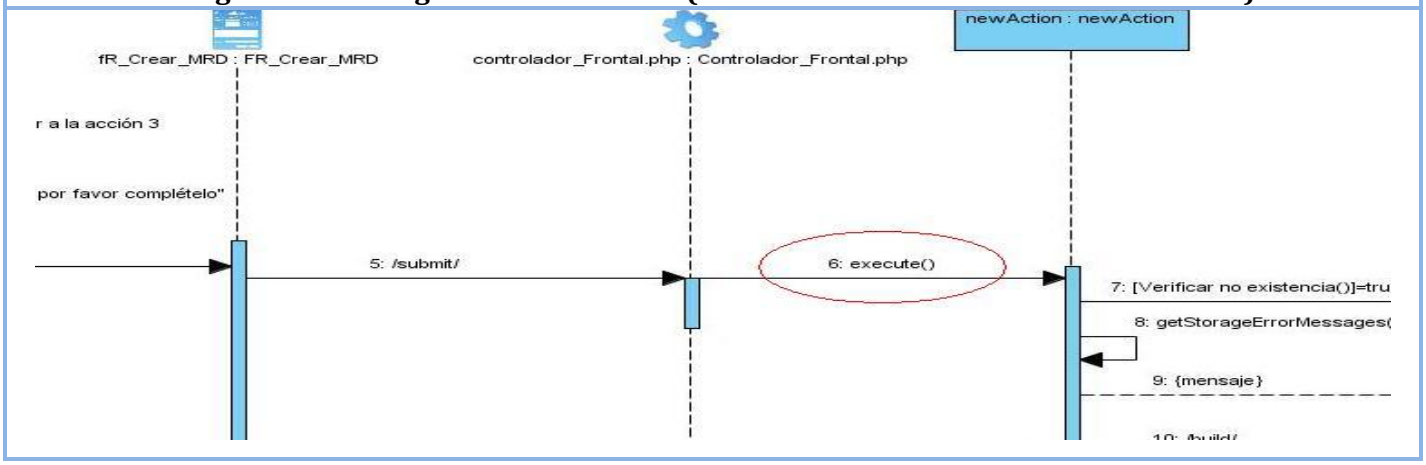


**Figura 7. Fragmento del diagrama de secuencia (CU Gestionar MRD: Escenario Crear MRD) del diseño donde se evidencia la utilización del patrón GRASP: Creador.**

**Patrón Bajo Acoplamiento:**

*"El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras".* (10)

Este patrón se pone de manifiesto principalmente en el diagrama de clases persistentes donde cada una de estas se comunica con el menor número de clases posible, logrando así un bajo acoplamiento entre las mismas.

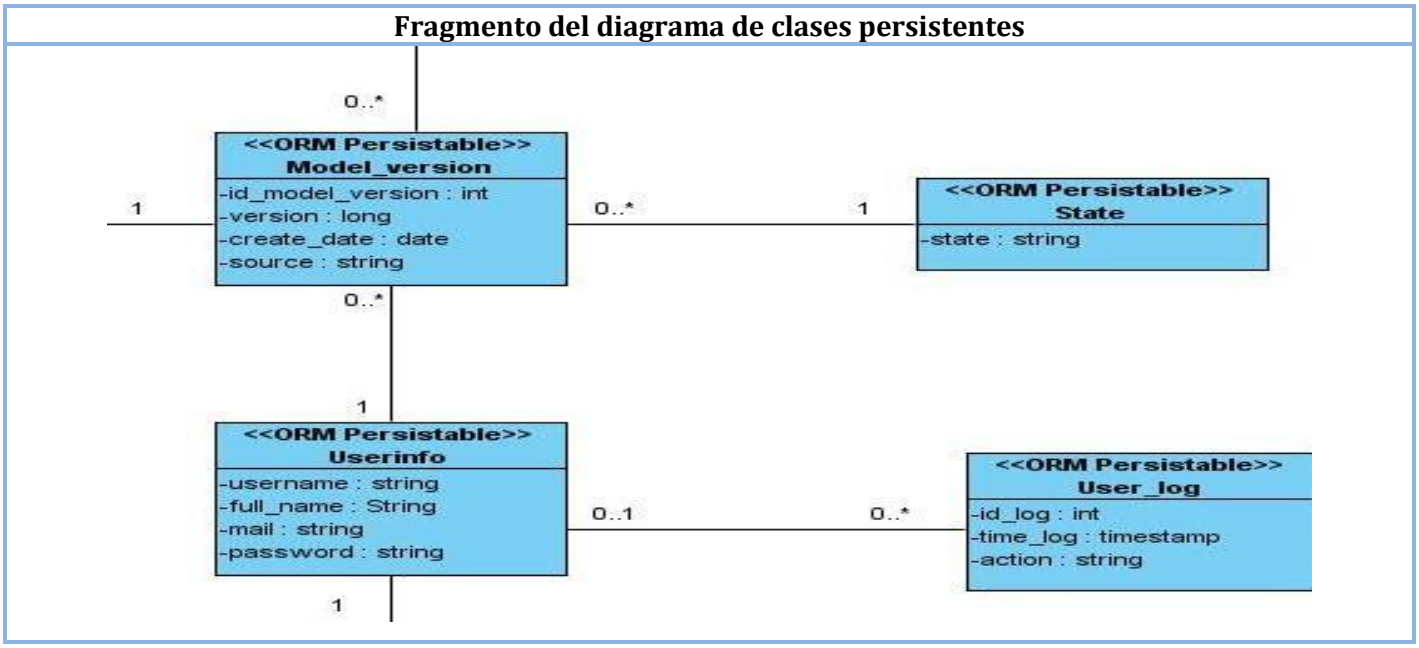


**Figura 8. Fragmento del diagrama de clases persistentes donde se evidencia la utilización del patrón GRASP: Bajo Acoplamiento.**

**Patrón Alta Cohesión:**

*"En la perspectiva del diseño orientado a objeto, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme".* (10)

En la solución se encuentran varias clases **Action** que contienen funcionalidades, las que poseen un único propósito, no desempeñado por el resto de los elementos, siendo estas funcionalidades las encargadas de controlar las acciones de las plantillas. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo. Este patrón quedó representado como se muestra a continuación:

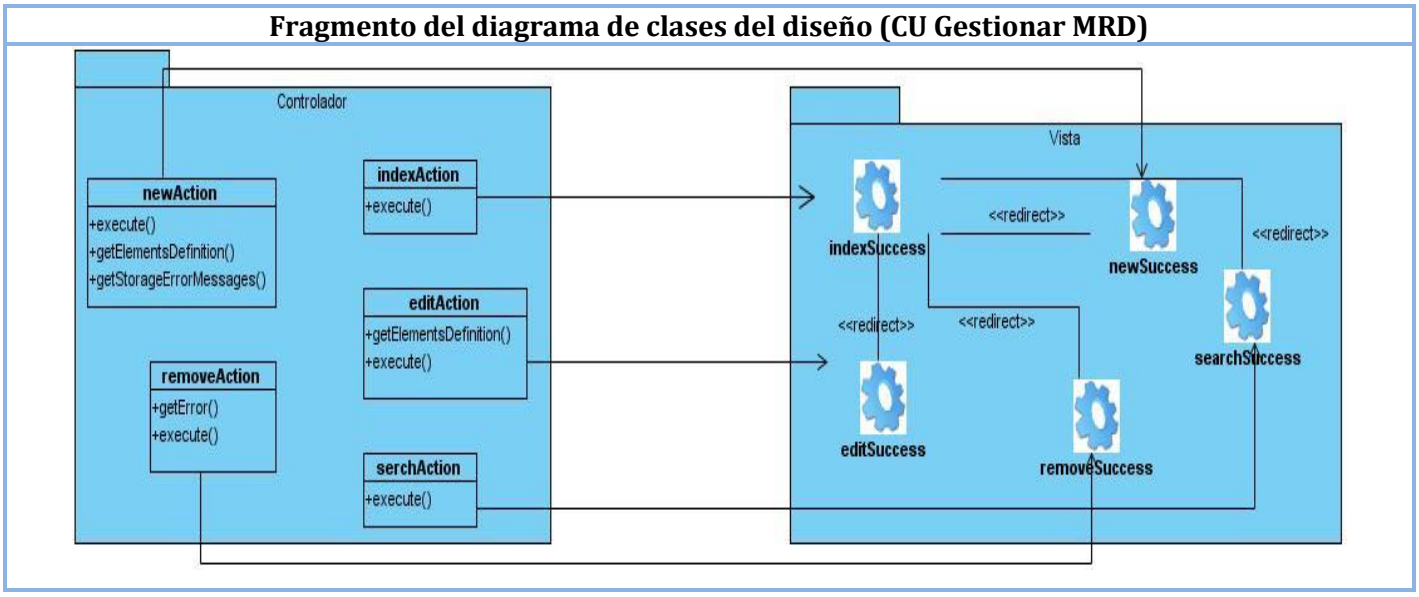


**Figura 9. Fragmento del diagrama de clases del diseño (CU Gestionar MRD) donde se evidencia la utilización del patrón GRASP: Alta Cohesión.**

**Patrón Controlador:**

*"Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación".* (10)

En la solución al utilizar la clase controlador_frontal.php para procesar los pedidos del usuario y realizar los cambios en el modelo y la vista, queda evidenciado este patrón.

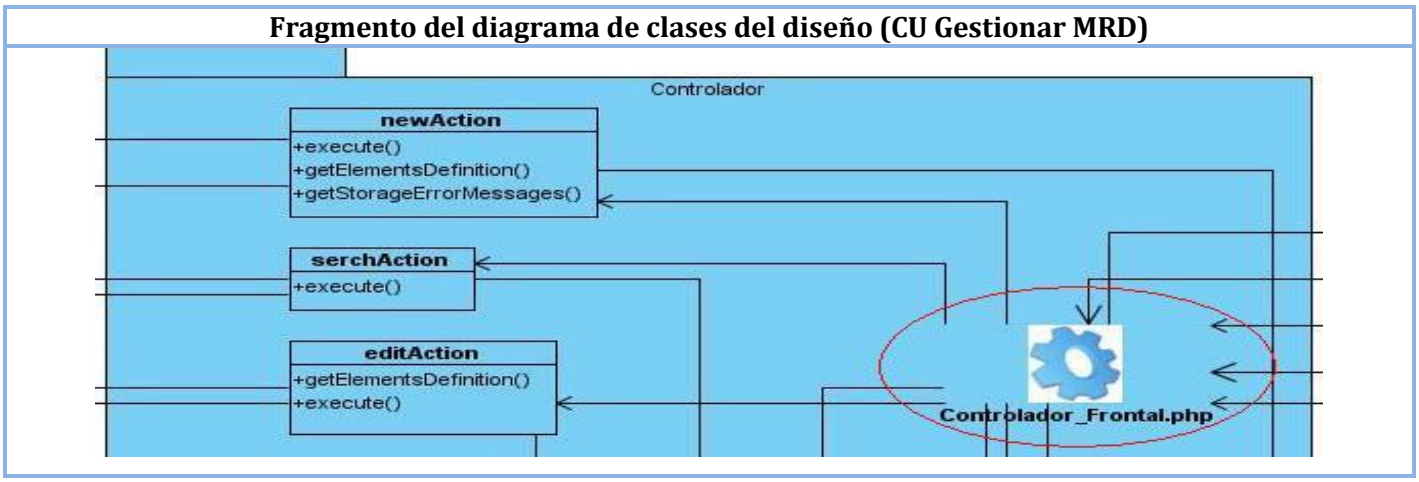


**Figura 10. Fragmento del diagrama de clases del diseño (CU Gestionar MRD) donde se evidencia la utilización del patrón GRASP: Controlador.**

## 3.2 Modelo de diseño

Es un modelo de objetos que describe la realización de los casos de uso, sirve como una abstracción del modelo de implementación y el código fuente y es usado como una entrada esencial para las actividades en la implementación y pruebas.

### 3.2.1 Diagramas de clases del diseño

Representa una abstracción de una o varias clases en la implementación del sistema, dependiendo del lenguaje de programación. Las clases definen los objetos, con los cuales se implementan los casos de uso.

Para modelar las clases del diseño de un sistema que emplea Symfony, es necesario delimitar bien lo que es del framework y lo propio del caso de uso. Los componentes de Symfony se encapsulan en un subsistema que contiene todos los elementos de este, que actúan o intervienen en la realización de los casos de uso. Al ser estos propios del framework, la manera en que trabajan es transparente al programador. Para que el diagrama no se tornara complejo se decidió modelar estos componentes como un subsistema con el nombre de Componentes de Symfony.

Otro elemento fundamental en los diagramas es el mapeo objeto-relacional (ORM) que utiliza Symfony, el cual proporciona la persistencia de los objetos y un servicio de consulta, en este caso Propel y se representa también como un subsistema. Para lograr una mayor organización, en los elementos de diseño están ubicados en la capa a la que responde según las definidas por el patrón MVC.

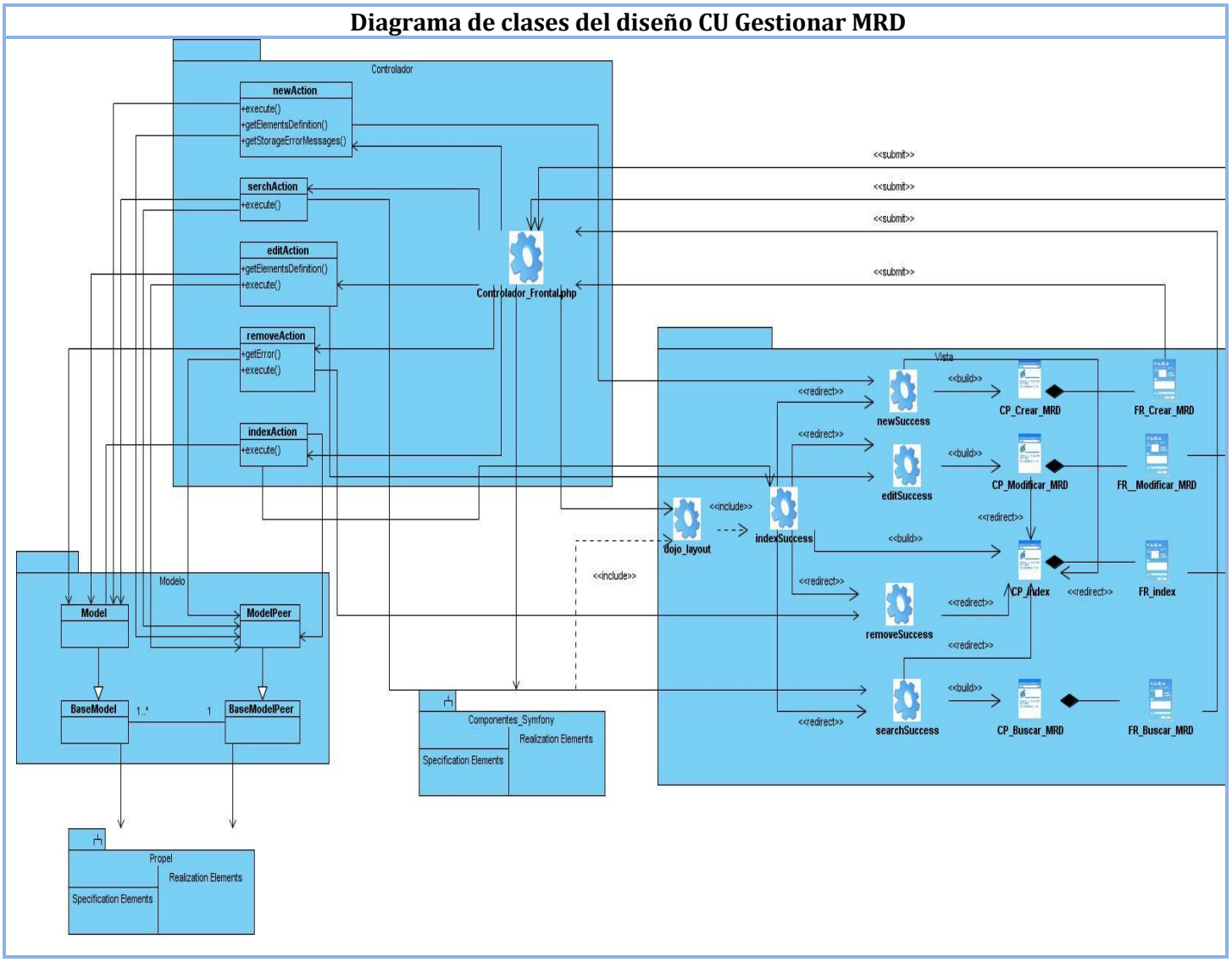


**Figura 11. Ejemplo de diagrama de clases del diseño CU Gestionar MRD.**

***El resto de los diagramas de clases del diseño se puede ver en: Documentos Complementarios.***

## 3.2.2 Diagramas de interacción

Muestra una interacción que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Estos diagramas pueden ser de dos tipos: diagramas de secuencia o diagramas de colaboración. En la figura 12 se representan un ejemplo de uno de los diagramas de secuencia correspondientes al diseño del sistema.

**Diagrama de secuencia CU Gestionar MRD. Escenario: Crear MRD**

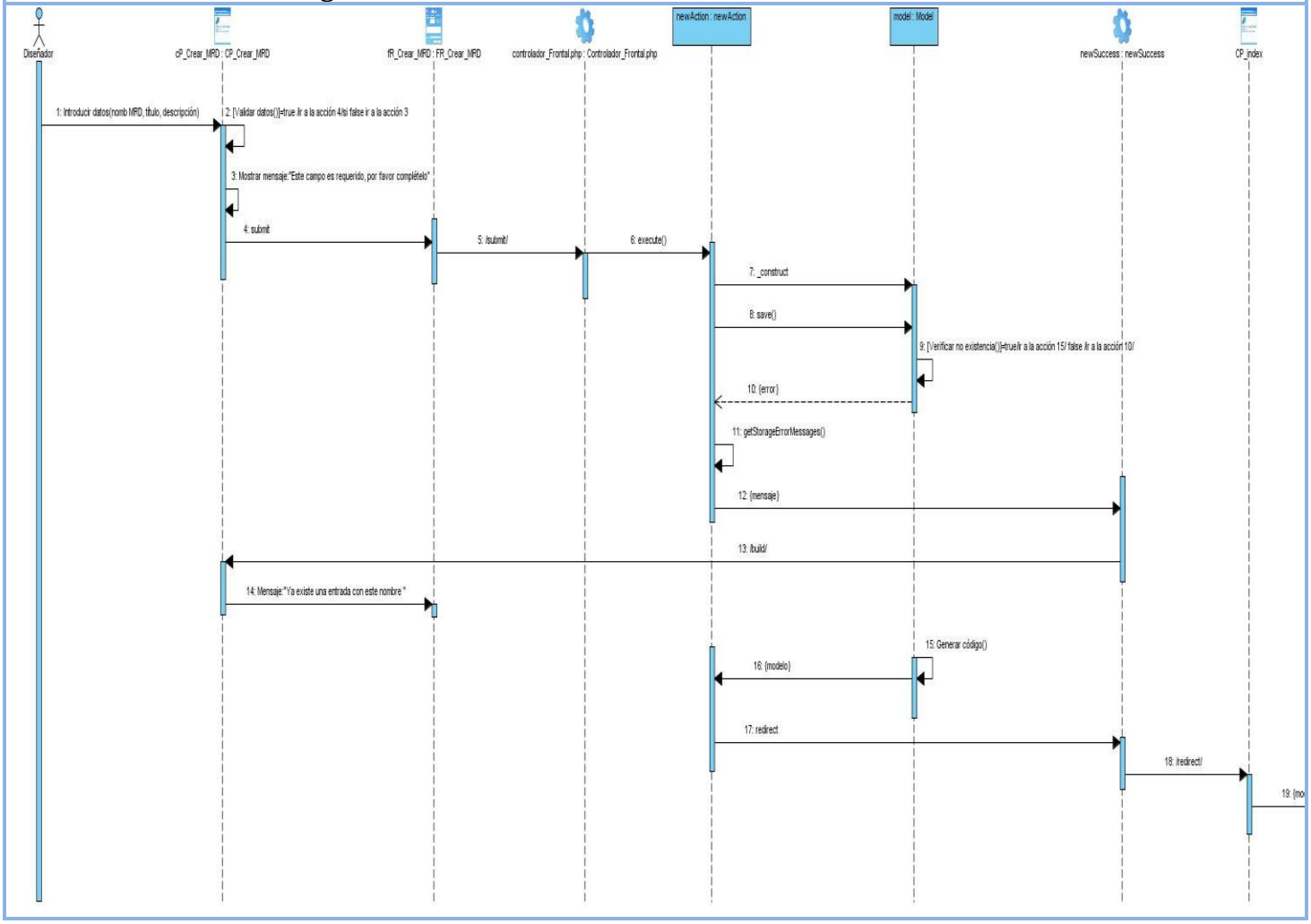


**Figura 12. Diagrama de secuencia CU Gestionar MRD. Escenario: Crear MRD.**

***Diagramas de secuencia del CU Gestionar MRD (Ver Anexo: 1) Los diagramas de secuencia correspondientes al resto de los CU del sistema, se pueden encontrar en: Documentos Complementarios.***

## 3.3 Diseño de la Base de Datos

El diseño de la base de datos tiene como propósito asegurar que los datos persistentes sean almacenados consistente y eficientemente, así como definir el comportamiento que debe ser implementado en la base de datos.

### 3.3.1 Diagrama de clases persistentes

*"El diagrama de clases persistentes muestra todas las clases capaces de mantener su valor en el espacio y en el tiempo".* (11)

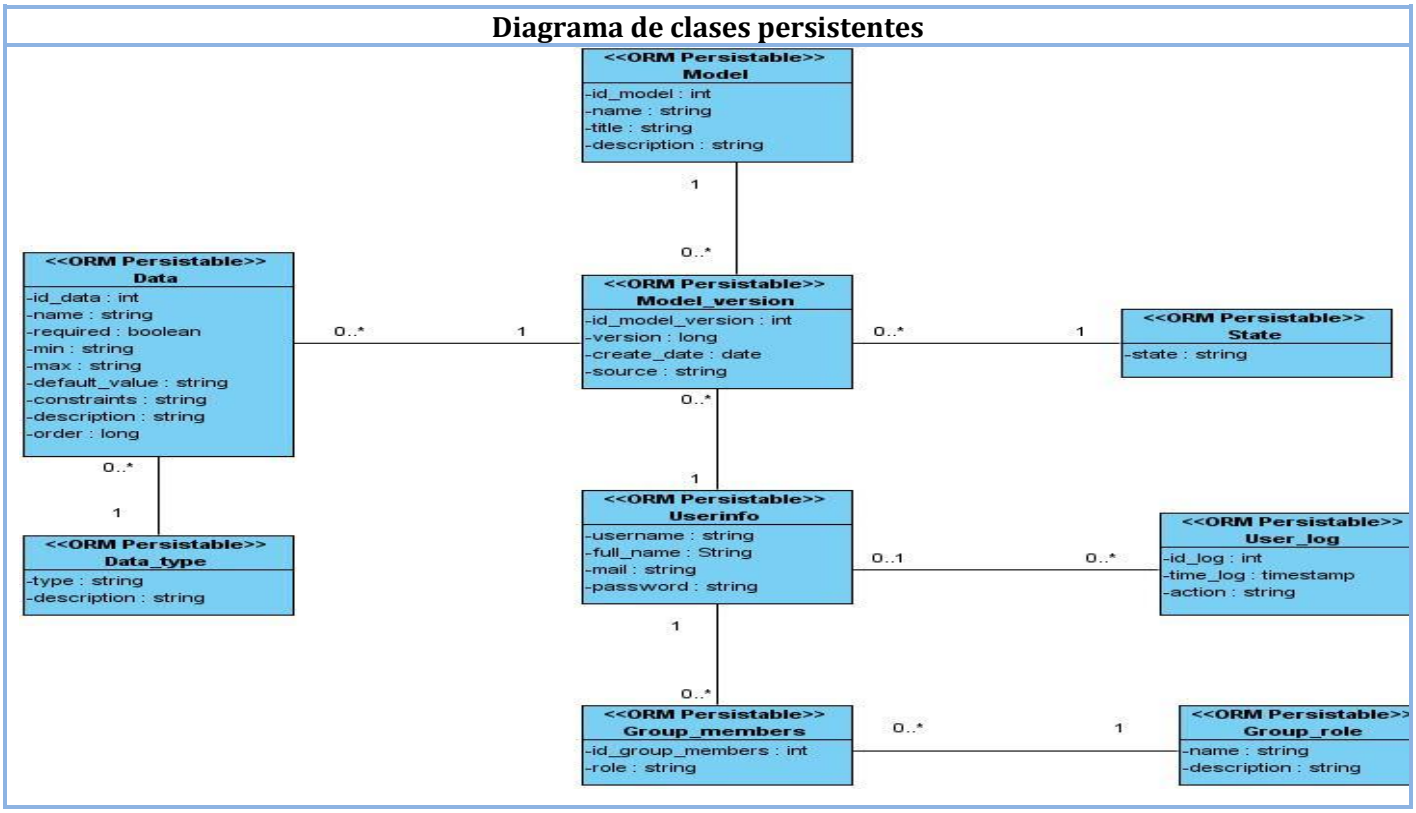


**Figura 13. Diagrama de clases persistentes.**

### 3.3.2 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. Para este caso el modelo se obtuvo a partir de una base de datos ya existente, quedando representado como se muestra en la figura.

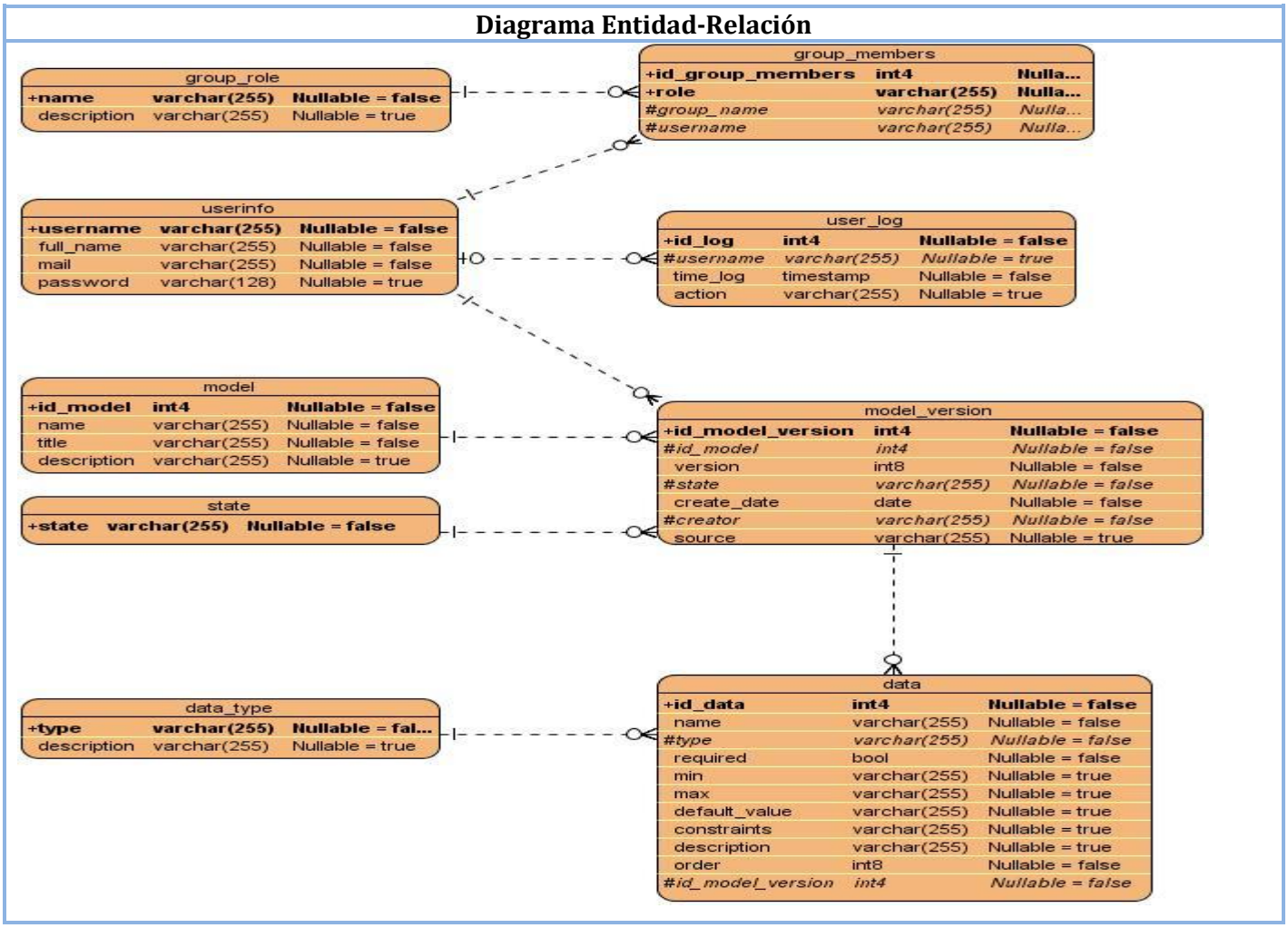


**Figura 14. Diagrama Entidad-Relación.**

***Descripción de las tablas de la base de datos (Ver: Anexo 2)***

### 3.3.3 Diagrama de despliegue

Un 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.

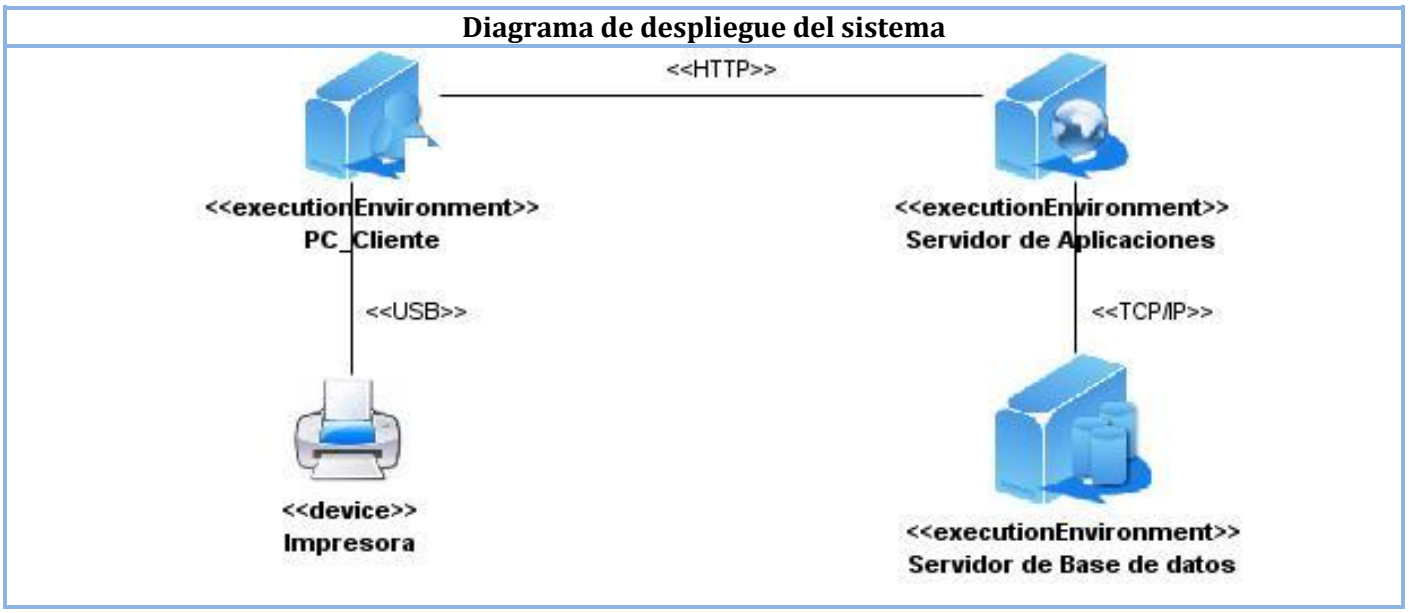


**Figura 15. Diagrama de despliegue del sistema.**

## Conclusiones parciales del capítulo

➢ El uso del patrón MVC, al igual que la aplicación de patrones de diseño, facilitó el diseño de las clases permitiendo que se obtengan clases mejor diseñadas, flexibles y reutilizables.

➢ La realización del diagrama de despliegue propició una visión de cómo está distribuido el sistema físicamente, quedando representado por tres nodos de procesamiento: 1 PC cliente, 1 servidor de aplicaciones y 1 servidor de base de datos; además 1 nodo dispositivo representando a la impresora conectada a la PC cliente.

# Capítulo 4: Implementación y Prueba

Se describe la implementación de los componentes a partir de los requerimientos identificados. Se presenta el diagrama de componentes correspondiente a cada uno de los casos de uso a implementar. Además, se muestran fragmentos relevantes del código y las vistas principales de la aplicación; así como ejemplos de las pruebas funcionales aplicadas, que permiten examinar las funcionalidades de la aplicación y garantizar la calidad del software.

## 4.1 Modelo de Implementación

"El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros". (11) Esta descripción es de gran utilidad a la hora de implementar el sistema, facilita la organización del trabajo y lo hace más entendible a los desarrolladores.

### 4.1.1 Diagrama de componentes

Este artefacto es usado para estructurar el modelo en términos de subsistemas de implementación y mostrar las relaciones entre los elementos. 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 en capas. A continuación se muestra un ejemplo del diagrama de componentes correspondiente al caso de uso del sistema Gestionar MRD.

**Diagrama de componentes CU Gestionar MRD**

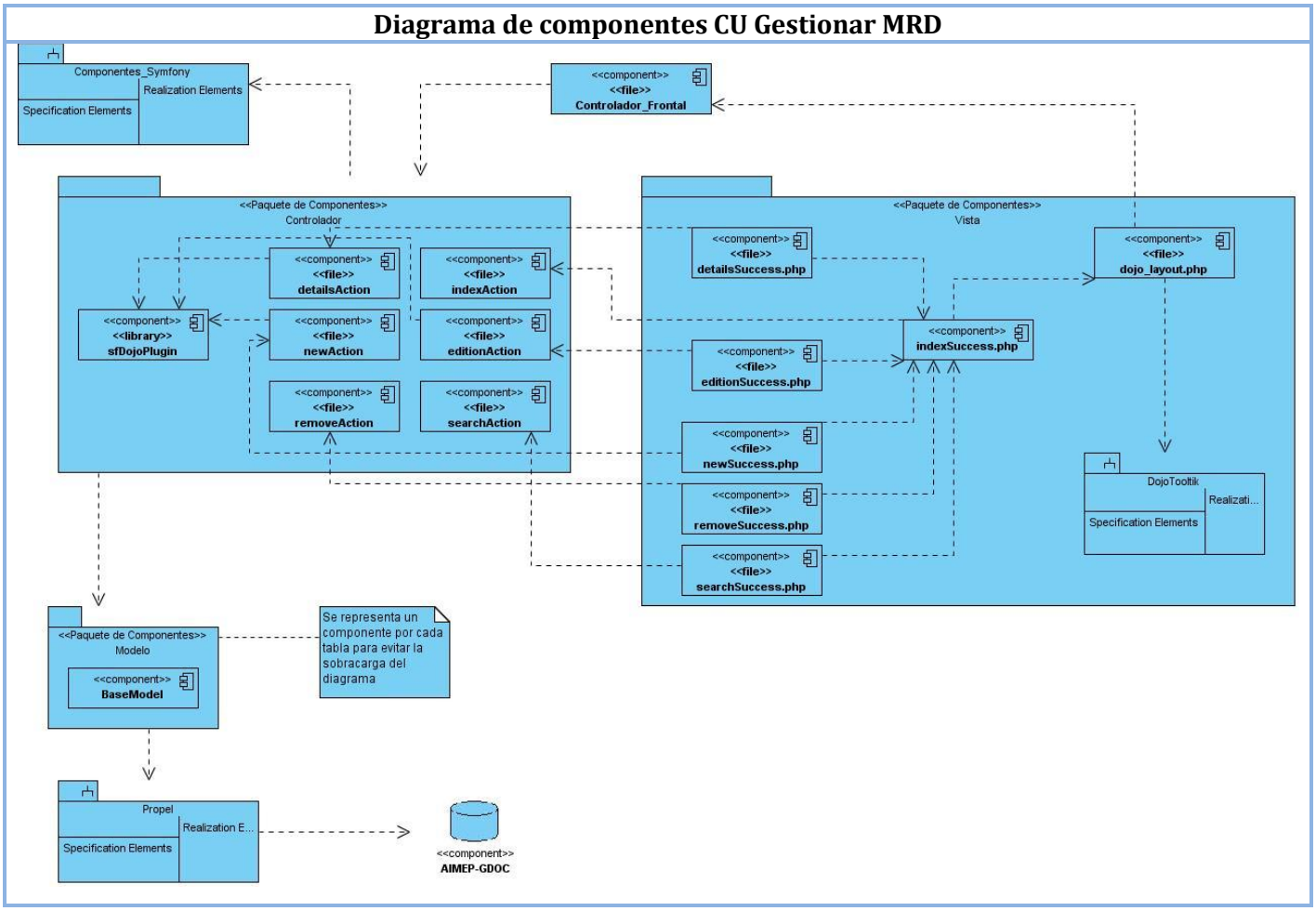


**Figura 16. Diagrama de componentes CU Gestionar MRD.**

***Los diagramas de componentes correspondientes a los casos de uso Administrar Versión y Editar Versión, se pueden encontrar en: Documentos Complementarios.***

## 4.2 Modificaciones realizadas al editor TinyMCE

El editor TinyMCE fue empleado para dar apoyo al desarrollo de la aplicación. Después del estudio realizado sobre sus características y funcionalidades se decidió realizar modificaciones para ajustarlo a las necesidades del equipo de desarrollo. Entre los cambios más relevantes se encuentran:

- Insertar y modificar elementos de formulario tales como: texto, cadena, selección, verificación, entero y real.

- Guardar y activar una versión, que permite establecer la correspondencia que existe entre las variables de una versión con el resto de las versiones pertenecientes a un modelo, manteniendo así los cambios controlados.

## 4.3 Código Fuente

La mayoría de los métodos implementados, responden a funciones básicas de inserción, actualización, eliminación y búsqueda. La implementación de estas acciones se vuelve realmente sencilla mediante el uso que hace Symfony de Propel para realizar el mapeo objeto-relacional. Este genera la mayor parte del código de acceso a datos proporcionando una abstracción al punto, que permite que los implementadores tengan que utilizar muy pocas consultas a la base de datos.

Entre los métodos desarrollados está el método **save (),** que se encarga de guardar y activar una versión, para ello verifica si existe alguna versión activa y cambia su estado a obsoleto. Por último, a la versión actual le pone su estado en activo, cumpliéndose así la restricción de que sólo puede existir una versión activa.

**Clase ModelVersion, método save()**

```
public function save(PropelPDO $con = null) {
    if ($this->isColumnModified( ModelVersionPeer::STATE ) && $this->getState() == 'active') {
        $model = $this->getModel();

        if ($model) {
            $C = new Criteria();
            $C->add(ModelVersionPeer::STATE, 'active', Criteria::EQUAL);

            $activeVersions = $model->getModelVersions( $C );

            foreach ($activeVersions as $activeVersion) {
                $activeVersion->setState('obsolete');
                $activeVersion->save();
            }
        }
    }
    parent::save($con);
}
```

**Figura 17. Ejemplo de la funcionalidad guardar y activar versión incorporada al editor TinyMCE.**

## 4.4 Pantallas principales de la aplicación

### Área de los Modelos

La aplicación desarrollada consta de una interfaz principal. En el centro de la misma se encuentran listados los modelos de recogida de datos existentes; mostrando su código, nombre, título y descripción. A la izquierda se tiene la opción ir a la página principal, donde se encontrarán el resto de los módulos luego de ser integrada la aplicación. En la parte superior se encuentra ubicada la barra de herramientas, la cual contiene las funcionalidades básicas de la aplicación: actualizar listado, adicionar, modificar y eliminar modelos de recogida de datos, así como ir a las versiones pertenecientes a un modelo seleccionado. En la siguiente figura se muestra como se adiciona un nuevo modelo.

**Interfaz para la inserción de un nuevo modelo**

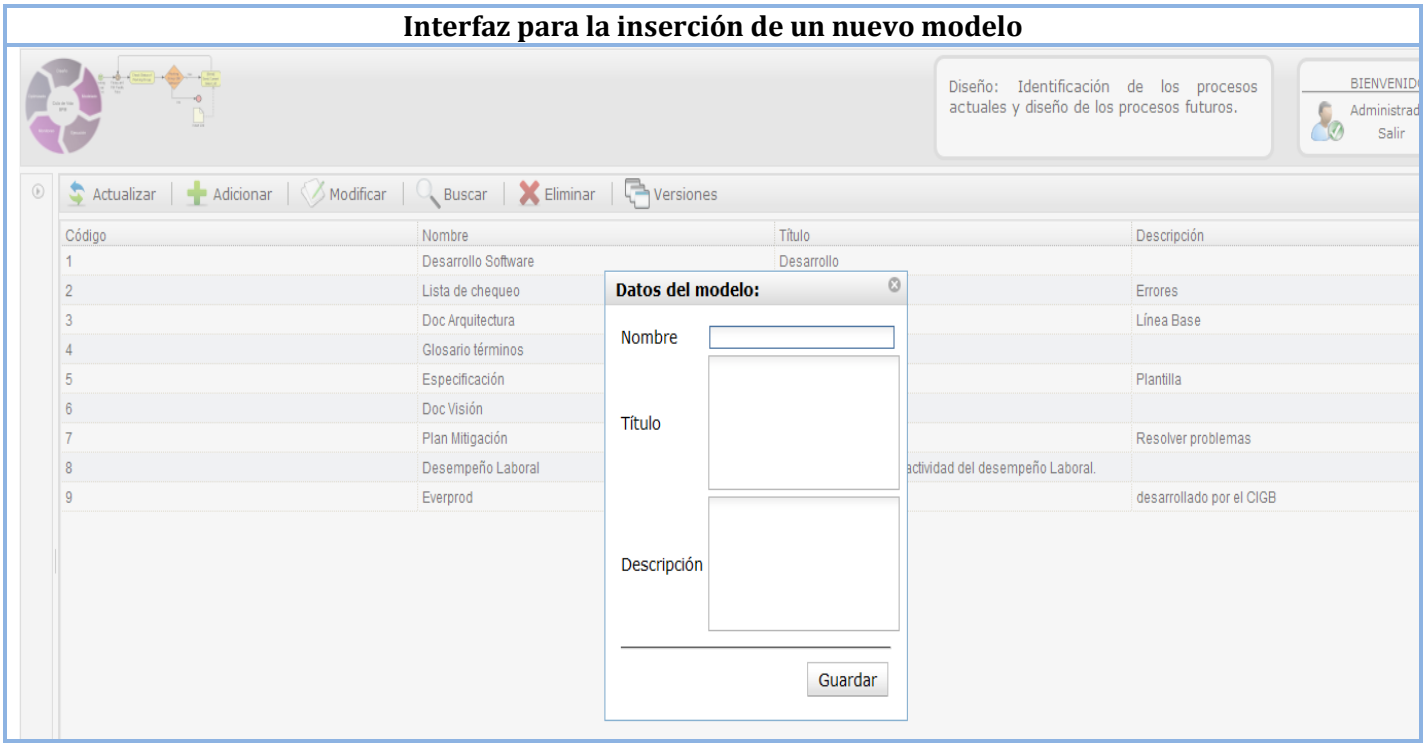


**Figura 18. Ejemplo de la interfaz para insertar un modelo.**

### Área de las Versiones

La aplicación consta de una interfaz para manipular la información referente a las versiones. En su centro se encuentran listadas las versiones correspondientes al modelo seleccionado; mostrando de estas su código, modelo, número de versión, estado, fecha de creación y creador. A la izquierda se cuenta con la opción descrita en el área de los modelos. En la parte superior se encuentra ubicada la barra de herramientas, la cual contiene las siguientes funcionalidades: regresar, actualizar listado, adicionar, buscar, eliminar, editar y mostrar las versiones, como se muestra en la figura 19.

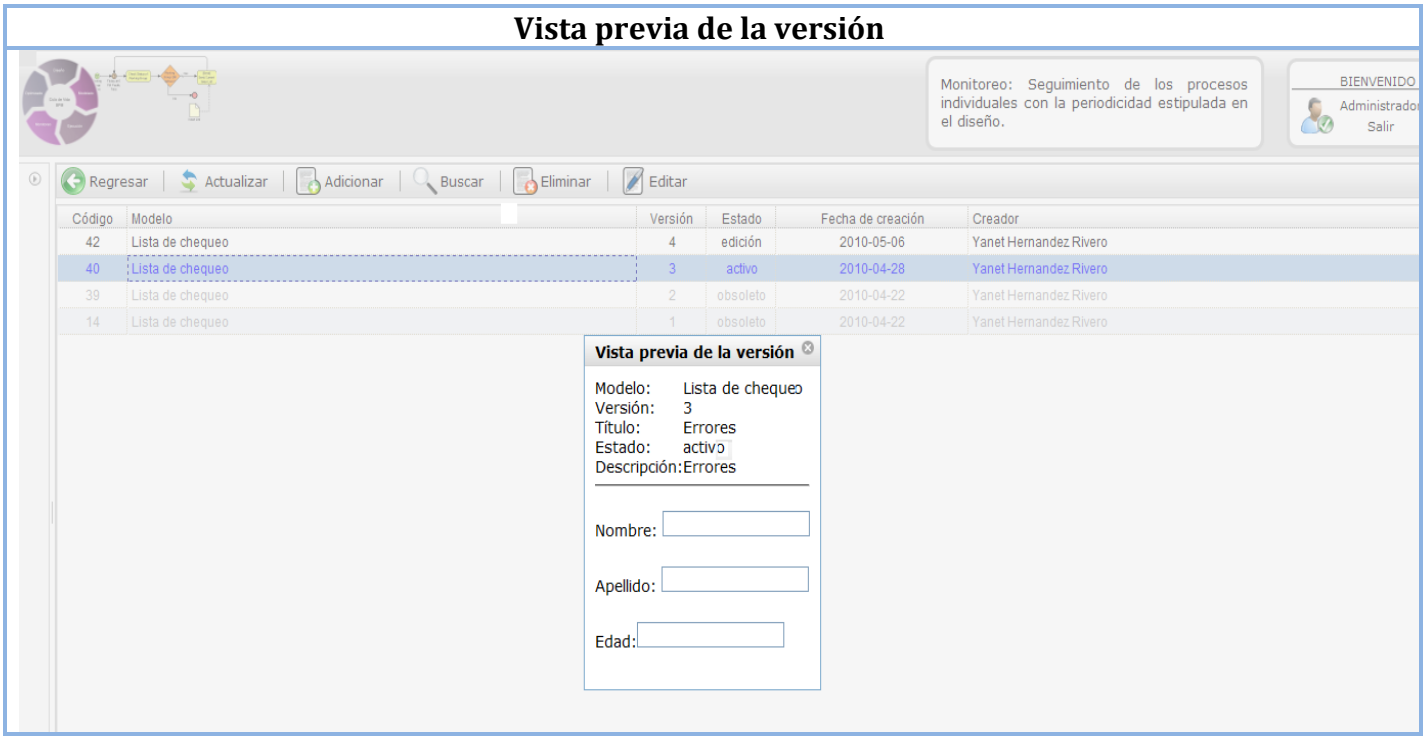


**Figura 19. Ejemplo de la vista previa de una versión**

### Área de diseño

La aplicación también cuenta con un editor de texto para diseñar las versiones. En su centro se encuentra el área de trabajo, donde se insertarán los elementos seleccionados por el diseñador. A la izquierda se cuenta con la opción antes descrita. En la parte superior está ubicado el editor, que además de sus funcionalidades básicas se añadieron las siguientes: regresar, guardar, guardar y activar versión, pantalla completa así como insertar y modificar elementos de diseño.

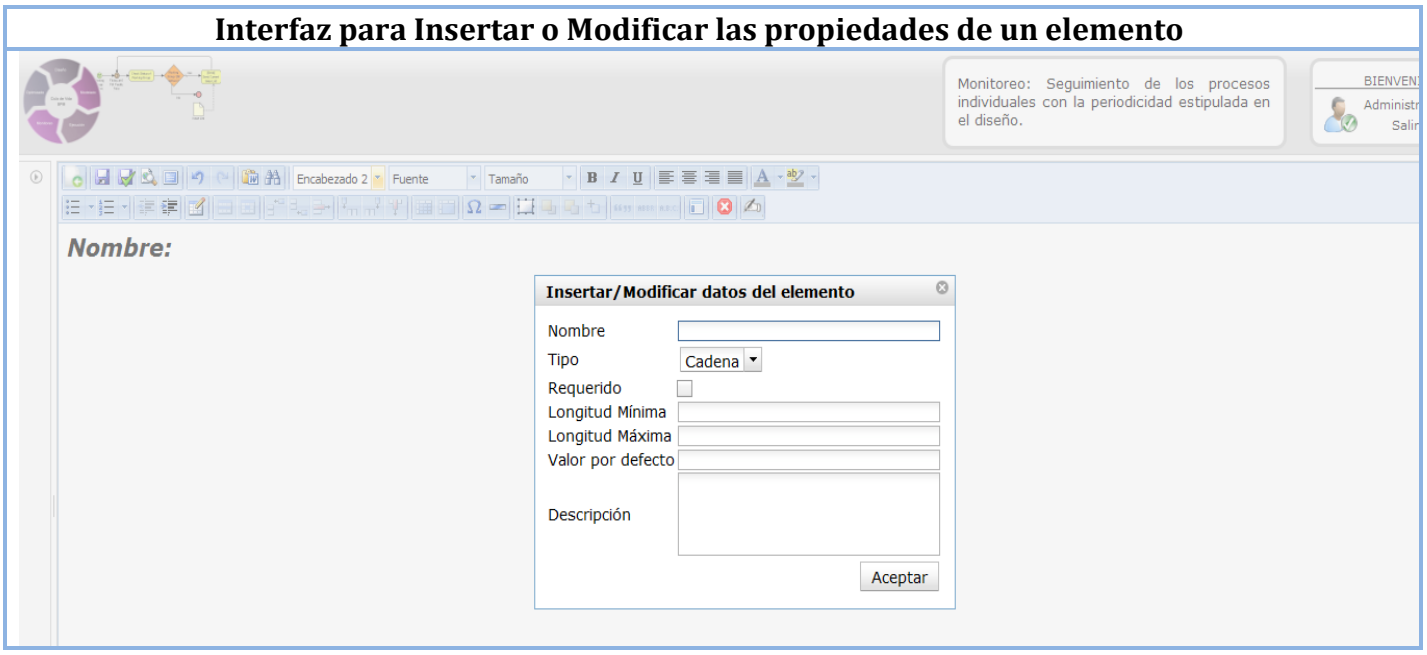


**Figura 20. Ejemplo de la interfaz para Insertar o Modificar las propiedades de un elemento.**

## 4.5 Diseño de casos de prueba. Pruebas funcionales

*"Las pruebas de software constituyen un pilar indispensable para evaluar y determinar la calidad de un software. Concretamente se puede definir pruebas de software como:*

- *El proceso de ejecución de un programa con la intención de descubrir errores previos a la entrega al usuario final.*
- *Una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones específicas, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente".*(12)

Entre los tipos de pruebas que se realizan en un sistema está la que evalúa la funcionalidad de este, denominada prueba funcional, que tiene por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para las cuales han sido creados.

El proceso de realización de pruebas a un software, está precedido por el diseño de los casos de prueba, que se definen según las funcionalidades descritas en los casos de uso. Se parte de las descripciones de los casos de uso del sistema, como apoyo para las revisiones.

Existen casos de pruebas para los diferentes métodos: Caja Negra y Caja Blanca. En el proceso de pruebas en cuestión se hace uso de los casos de prueba de Caja Negra, de ahí el motivo por el que se diseña un caso de prueba por caso de uso del sistema. El siguiente es un ejemplo donde se detalla el caso de uso **Gestionar MRD**.

| Nombre de la sección | Escenarios de la sección | Descripción de la funcionalidad |
|---|---|---|
| SC1: Crear MRD. | EC1.1: Crear MRD. | En este escenario el diseñador indica crear un MRD e introduce los datos necesarios para su construcción. |
| | EC1.2: Crear MRD con campos vacíos. | Es la misma operación que se realiza en el EC 1.1, en caso de que no se completen los campos requeridos. |
| | EC1.3: Crear un MRD que ya existe. | Es la misma operación que se realiza en el EC 1.1, en caso de que el nombre se duplique. |
| SC2: Actualizar listado de MRD. | EC2.1: Actualizar listado de MRD. | En este escenario el diseñador selecciona la opción de actualizar el listado de los MRD. |
| SC3: Buscar y Visualizar MRD. | EC3.1: Buscar y Visualizar MRD. | En este escenario el diseñador selecciona los datos por los cuales desea realizar la búsqueda e indica un valor. |
| | EC3.2 Buscar y Visualizar un MRD introduciendo un valor incorrecto. | Es la misma operación que se realiza en el EC3.1, en caso de que el valor por el que se desea buscar no sea correcto. |

| | | |
|---|---|---|
| | EC3.3: Buscar y Visualizar un MRD que no existe. | Es la misma operación que se realiza en el EC3.1, en caso de que el MRD no exista. |
| SC 4: Modificar MRD | EC4.1: Modificar MRD. | En este escenario el diseñador selecciona el modelo a modificar e introduce los datos. |
| | EC4.2: Modificar MRD con campos vacíos. | Es la misma operación que se realiza en el EC 4.1, en caso de que los campos requeridos queden vacíos. |
| | EC4.3: Modificar MRD introduciendo un nombre que ya existe. | Es la misma operación que se realiza en el EC 4.1, en caso de que el nombre se duplique. |
| SC5: Eliminar MRD. | EC5.1: Eliminar MRD. | En este escenario el diseñador selecciona el modelo e indica eliminar. |
| | EC5.2: Eliminar MRD con versiones asociadas. | Es la misma operación que se realiza en el EC 5.1, en caso de que el modelo tenga versiones. |

**Tabla 2. Secciones a probar en el Caso de Uso.**

A partir de esta descripción se detallan las variables que se encuentran en las interfaces asociadas al caso de uso Gestionar MRD. (Ver tabla 3)

| No. | Nombre de campo | Clasificación | Puede ser nulo | Descripción |
|---|---|---|---|---|
| **1.** | Nombre | Texto | No | Nombre del modelo. |
| **2.** | Título | Texto | Sí | Título del modelo. |
| **3.** | Descripción | Texto | Sí | Descripción asociada al modelo. |
| **4.** | Variable | Lista de selección | No | Se escoge la variable por la que se desea realizar la búsqueda. |
| **5.** | Criterio | Lista de selección | No | Se escoge el criterio por el que se desea |

| | | | | |
|---|---|---|---|---|
| | | | | realizar la búsqueda. |
| **6.** | Valor | Texto | Sí | Se indica el valor de la variable para realizar la búsqueda. |

**Tabla 3. Descripción de variables.**

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los valores introducidos en el sistema. Para ello se utilizó un juego de datos válidos e inválidos, empleando la técnica de partición de equivalencia.

Este es un ejemplo de la matriz de datos del caso de uso Gestionar MRD, de la SC1 Crear MRD.

| Id del Escenario | Escenario | Variables | | | Respuesta del sistema | Resultado de la prueba | Flujo central |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | | | |
| EC1.1 | Crear MRD | V | V | V | Registra y visualiza el nuevo modelo creado. | Satisfactorio | 1. El diseñador introduce los datos para crear el modelo.<br>2. El sistema verifica que los campos requeridos hayan sido completados.<br>3. El sistema verifica que no exista un modelo con el mismo nombre.<br>4. El sistema genera un código único al modelo a crear.<br>5. El sistema registra y visualiza el nuevo modelo creado. |
| EC 1.2 | Crear MRD con campos | I | V | V | Emite un mensaje: "Este | Satisfactorio | 2.1 Si el campo nombre quedó vacío el sistema |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | vacíos. | | | | campo es requerido, por favor complételo". | | emite un mensaje: "Este campo es requerido, por favor complételo". |
| EC 1.3 | Crear un MRD que ya existe. | I | V | V | Emite un mensaje: "Ya existe una entrada con este nombre ". | Satisfactorio | 3.1 Si el modelo ya existe el sistema emite un mensaje: "Ya existe una entrada con este nombre ". |

**Tabla 4. Matriz de datos**

Los resultados de las pruebas que no fueron satisfactorios pasaron a ser no conformidades y se emitieron en el registro de defectos y dificultades detectados.

En la tabla 5 se muestran las no conformidades detectadas durante las pruebas a la aplicación.

| Elemento | No. | No conformidad | Aspecto correspondiente | Etapa de detección | Significativa | No Significativa | Estado NC | Resp. equipo de desarrollo |
|---|---|---|---|---|---|---|---|---|
| Aplicación | 1 | En el CU Gestionar MRD, en la sección Buscar y Visualizar, cuando se busca un modelo que no existe el sistema no muestra el mensaje indicado. | http://10.36.18.4:5800/DataModelDesigner.php | Pruebas de funcionalidad | | X | PD 22/4/10 RA 23/4/10 | Se agregó una nueva validación para mostrar el mensaje. |
| Aplicación | 2 | En el CU Editar Versión, en la sección Insertar elemento, cuando se deja el campo opciones | http://10.36.18.4:5800/DataModelDesigner.php | Pruebas de funcionalidad | | X | PD 22/4/10 RA 23/4/10 | Se cambió el mensaje. |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | vacío, no se muestra el mensaje esperado. | | | | | | |
| Aplicación | 3 | En el CU Editar Versión, en la sección Insertar elementos (real o entero), cuando se introducen datos en los campos valor min y valor máx, se admiten rangos no válidos. | http://10.36.18.4:5800/DataModelDesigner.php | Pruebas de funcionalidad | X | | PD 22/4/10<br>RA 24/4/10 | Se validaron los campos. |
| Aplicación | 4 | En el CU Editar Versión, en la sección Insertar elementos (texto o cadena), cuando se introducen datos en los campos longitud min y longitud máx, se admiten rangos no válidos. | http://10.36.18.4:5800/DataModelDesigner.php | Pruebas de funcionalidad | X | | PD 22/4/10<br>RA 24/4/10 | Se validaron los campos. |

**Tabla 5. Registro de defectos y dificultades detectados.**

***El resto de los casos de prueba se pueden ver en: Documentos Complementarios.***

## Conclusiones parciales del capítulo

➢ Se implementaron 66 componentes identificados haciendo uso de NetBeans, PHP5, JavaScript, Symfony, Dojo Toolkit, AJAX, PostgreSQL y Apache.

➢ Se realizaron modificaciones sobre el editor TinyMCE como son: insertar o modificar elementos de formulario, pantalla completa, guardar, guardar y activar versión y regresar, cumpliéndose así las expectativas de clientes y desarrolladores.

➢ La realización de pruebas funcionales a la aplicación a través del diseño de 3 casos de prueba, arrojaron como resultado 4 no conformidades, de ellas, 2 significativas y 2 no significativas, que fueron resueltas en un corto período de tiempo.

.

# Conclusiones

- Se implementó la primera versión del módulo “Diseño de Modelos de recogida de Datos”, que permite el diseño y control de los modelos en la industria biotecnológica.

- Se realizaron modificaciones sobre el editor TinyMCE tales como, la inserción y modificación de elementos de formulario, además de establecer la correspondencia que existe entre las variables de una versión con el resto de las versiones pertenecientes a un modelo, logrando así mantener los cambios controlados.

- Se realizaron pruebas funcionales a la aplicación, que arrojaron resultados satisfactorios.

# Recomendaciones

Después de lograr los objetivos que se trazaron al principio de este trabajo, se plantean las siguientes recomendaciones:

- Continuar el desarrollo de la herramienta con el objetivo de incorporar otras funcionalidades que una vez realizado el diseño de un modelo, el sistema permita introducir datos en ellos.

- Incorporar nuevos elementos de diseño al editor de texto como campos de contraseña y campos de radio, con el fin de enriquecer el diseño.

# Referencias Bibliográficas

1. **Larman, Craig.** *UML y Patrones.* p. 15. Vol. 1, epígrafe 1.9. [Online] http://bibliodoc.uci.cu/pdf/reg00061.pdf

2. Capitulo-I-HERRAMIENTAS-CASE. *Capitulo-I-HERRAMIENTAS-CASE.* [Online] [Cited: 10 25, 2009.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE

3. kioskea. *kioskea.* [Online] 10 16, 2008. [Cited: 10 25, 2009.] http://es.kioskea.net/contents/langages/langages.php3

4. Cibernetia. *Cibernetia .* [Online] [Cited: 11 30, 2009.] http://www.cibernetia.com/manuales/instalacion_servidor_web/1_conceptos_basicos.php

5. librosweb. *librosweb.* [Online] 8 31, 2009. [Cited: 12 1, 2009.] http://www.librosweb.es/ajax/capitulo1.html

6. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado del Software.* s.l. : Addison Wesley, 2000. p. 107. Cap 6, Epígrafe 6.2. [Online] http://bibliodoc.uci.cu/pdf/8478290362.pdf

7. **Ivar Jacobson, Grady Booch,James Rumbaugh**. *El Proceso Unificado del Software*. s.l. : Addison Wesley, 2000. p. 110. Cap 6, Epígrafe 6.3. [Online] http://bibliodoc.uci.cu/pdf/8478290362.pdf

8. **Reynoso Billy, Carlos.** [En línea] Marzo 2004. [Citado: Enero 14, 2009.] http://www.willydev.net/descargas/prev/IntroArq.pdf

9. **Larman, Craig.** *UML y Patrones.* p. 189. Vol. 1, epígrafe 18.6. [Online] http://bibliodoc.uci.cu/pdf/reg00061.pdf

10. **Larman, Craig.** *UML y Patrones.* p. 191. Vol. 1, epígrafe 18.7. [Online] http://bibliodoc.uci.cu/pdf/reg00061.pdf

11. **Acuña, Karenny Brito.** Diseño e implementación del sistema. *Diseño e implementación del sistema.* [Online]http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm

12. **Linet Lores Sánchez, Diana Monné Roque.** *Aplicación de las pruebas de liberación al sistema informático de Genética Médica.* UCI. Ciudad de la Habana : s.n., 2009. p. 8. [Online] [Online] http://bibliodoc.uci.cu/TD/TD_2457_09.pdf

# Bibliografía

1. **Acuña, Karenny Brito.** Diseño e implementación del sistema. *Diseño e implementación del sistema.* [Online]http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm

2. anieko2k. *anieko2k.* [Online] [Cited: 12 03, 2009.] http://www.anieto2k.com/2009/08/29/ckeditor-la-reencarnacion-de-fckeditor/.

3. BALUART.NET. *BALUART.NET.* [Online] 07 31, 2005. [Cited: 11 27, 2009.] http://www.baluart.net/articulo/cual-es-el-mejor-editor-wysiwyg-on-line.

4. Beez Nest Latino. *Beez Nest Latino.* [Online] 01 14, 2009. http://dokeoslatinoamerica.wordpress.com/2009/01/14/algunos-frameworks-para-php-mas-usados/

5. BioCen. *BioCen.* [Online] [Cited: 10 14, 2009.] http://www.biocen.cu/.

6. Capitulo-I-HERRAMIENTAS-CASE. *Capitulo-I-HERRAMIENTAS-CASE.* [Online] [Cited: 10 25, 2009.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

7. Centro de Ingeniería Genética y Biotecnología. *Centro de Ingeniería Genética y Biotecnología.* [Online] [Cited: 10 14, 2009.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=105&Itemid=184.

8. Cibernetia. *Cibernetia .* [Online] [Cited: 11 30, 2009.] http://www.cibernetia.com/manuales/instalación_servidor_web/1_conceptos_básicos.php.

9. CKEditor. *CKEditor.* [Online] [Cited: 12 03, 2009.] http://ckeditor.com/what-is-ckeditor

10. *Clase teórico Práctica 1. Profundización del flujo de trabajo de requerimientos.* **Colectivo de autores Dpto ISW, UCI.** curso 2009-2010. p. 8.

11. *Conferencia # 5.Culminación del flujo de trabajo Análisis y Diseño. Diseño de la Base de Datos.* **Colectivo de autores Dpto ISW, UCI.** curso 2009-2010. p. 4.

12. *Conferencia # 2. Arquitectura y Patrones de diseño.* **Colectivo de autores Dpto ISW, UCI.** curso 2009-2010. p. 3.

13. *Conferencia6. Fase de Inicio.Disciplina de Requisitos.* **Colectivo de autores Dpto ISW, UCI.** curso 2009-2010.

14. **Corral, Rodrigo.** Geeks•ms. *Geeks•ms.* [Online] 01 15, 2007. [Cited: 12 01, 2009.] http://geeks.ms/blogs/rcorral/archive/2007/01/15/iquest-que-metodolog-iacute-a-de-desarrollo-elegir.aspx.

15. **Cuenca, Carlos Luis.** DesarrolloWeb.com. *DesarrolloWeb.com.* [Online] 03 20, 2003. [Cited: 11 12, 2009.] http://www.desarrolloweb.com/articulos/1112.php

16. Desarrolloweb.com. *Desarrolloweb.com.* [Online] 09 13, 2006. [Cited: 11 30, 2009.] http://www.desarrolloweb.com/scripts/protopad-wysiwyg-editor-html.html

17. **Fdez, David Carrero.** Carrero. *Carrero.* [Online] 08 27, 2009. [Cited: 12 05, 2009.] http://carrero.es/fckeditor-se-reescribe-en-el-nuevo-ckeditor-3-0/3775

18. Free Download Manager. *Free Download Manager.* [Online] [Cited: 12 02, 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/

19. GSINNOVA. *GSINNOVA.* [Online] [Cited: 11 29, 2009.] http://www.rational.com.ar/herramientas/roseenterprise.html

20. **Hernández, Claudia.** geocities. *geocities.* [Online] [Cited: 10 22, 2009.] http://www.geocities.com/claudiahernandez/EDITWEB.htm

21. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado del Software.* s.l. : Addison Wesley, 2000. [Online] http://bibliodoc.uci.cu/pdf/8478290362.pdf

22. kioskea. *kioskea.* [Online] 10 16, 2008. [Cited: 10 25, 2009.] http://es.kioskea.net/contents/langages/langages.php3

23. **Larman, Craig.** *UML y Patrones.* p. 15. Vol. 1, epígrafe 1.9. [Online] http://bibliodoc.uci.cu/pdf/reg00061.pdf

24. librosweb. *librosweb.* [Online] 8 31, 2009. [Cited: 12 1, 2009.] http://www.librosweb.es/ajax/capitulo1.html

25. **Linet Lores Sánchez, Diana Monné Roque.** *Aplicación de las pruebas de liberación al sistema informático de Genética Médica.* UCI. Ciudad de la Habana : s.n., 2009. p. 8. [Online] http://bibliodoc.uci.cu/TD/TD_2457_09.pdf

26. maestros del web. *maestros del web.* [Online] 07 31, 2007. [Cited: 12 01, 2009.] http://www.maestrosdelweb.com/editorial/los-frameworks-de-php-agilizan-tu-trabajo/

27. Notas sobre Metodologías Ágiles. *Notas sobre Metodologías Ágiles.* [Online] 04 27, 2008. [Cited: 01 21, 2010.] http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio?tmpl=component&print=1&page=

28. **Oktaba, Hanna.** UNAM POSTGRADO. *UNAM POSTGRADO.* [Online] [Cited: 10 11, 2009.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html

29. pixelco.us. *pixelco.us.* [Online] 06 05, 2008. [Cited: 12 02, 2009.] http://pixelco.us/blog/4-excelentes-editores-de-texto-para-agregar-en-cualquier-

30. **Prieto, Felix.** Programación III.I.T.I. de Departamento de Informática. *Programación III.I.T.I. de Departamento de Informática.* [Online] curso 2008-2009. [Cited: 1 14, 2010.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

31. **Razziel.** symfony-users. *symfony-users.* [Online] 5 18, 2008. [Cited: 1 15, 2010.] http://www.mail-archive.com/symfony-users@googlegroups.com/msg07161.html

32. **Reynoso Billy, Carlos.** [Online] Marzo 2004. [Citado: Enero 14, 2009.] http://www.willydev.net/descargas/prev/IntroArq.pdf

33. riie. *riie.* [Online] 07 02, 2006. [Cited: 12 01, 2009.] http://riie.com.pe/?a=48295

# Anexos

**Anexo 1:** Diagramas de secuencia.

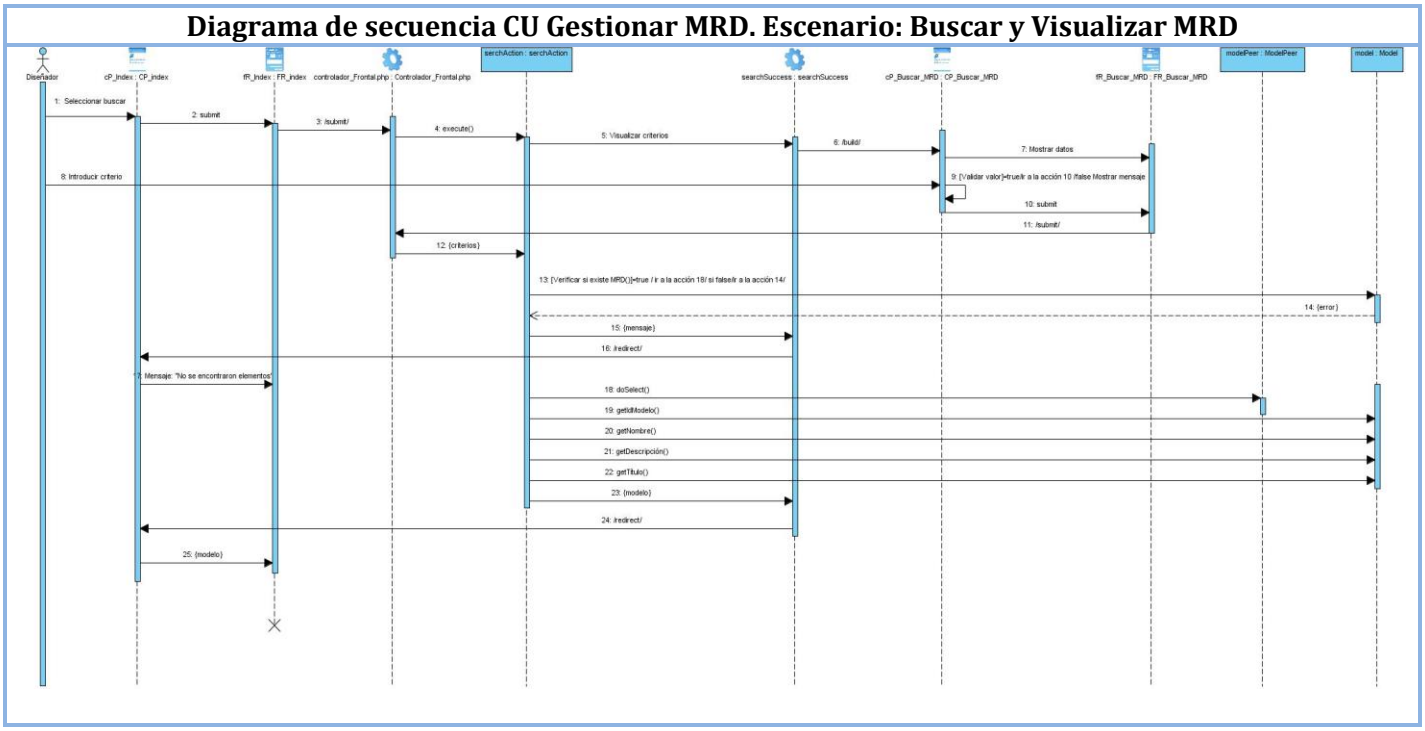


**Figura 21. Diagrama de secuencia CU Gestionar MRD. Escenario: Buscar y Visualizar MRD.**

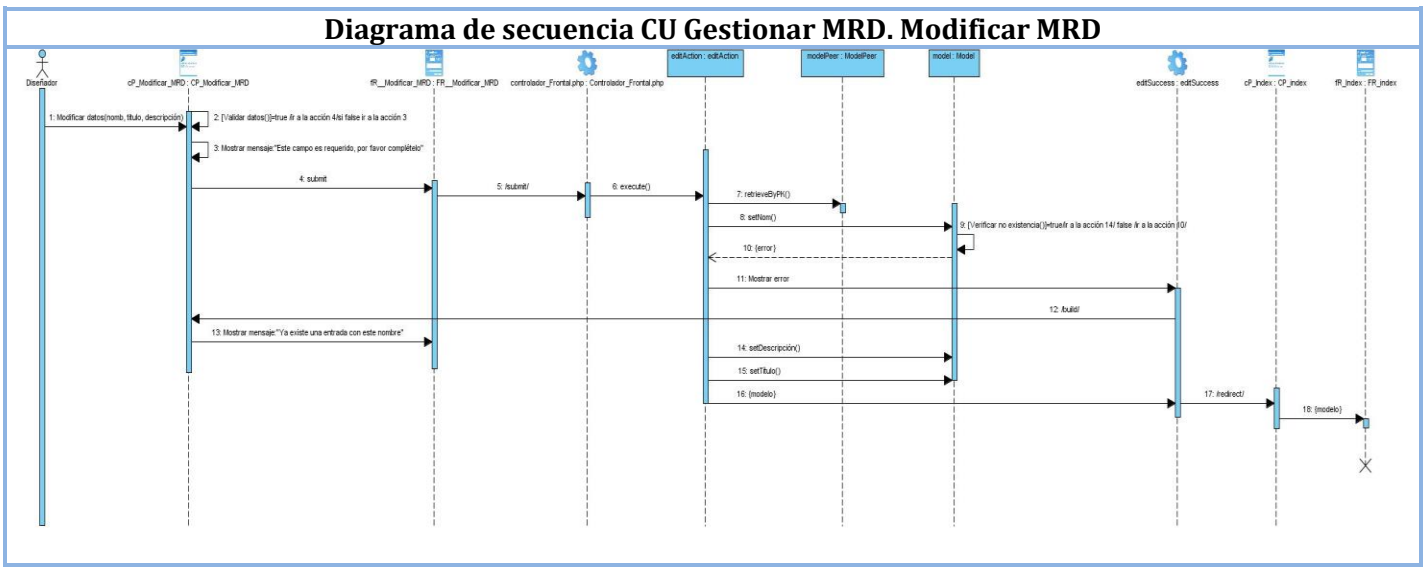


**Figura 22. Diagrama de secuencia CU Gestionar MRD. Escenario: Modificar MRD.**

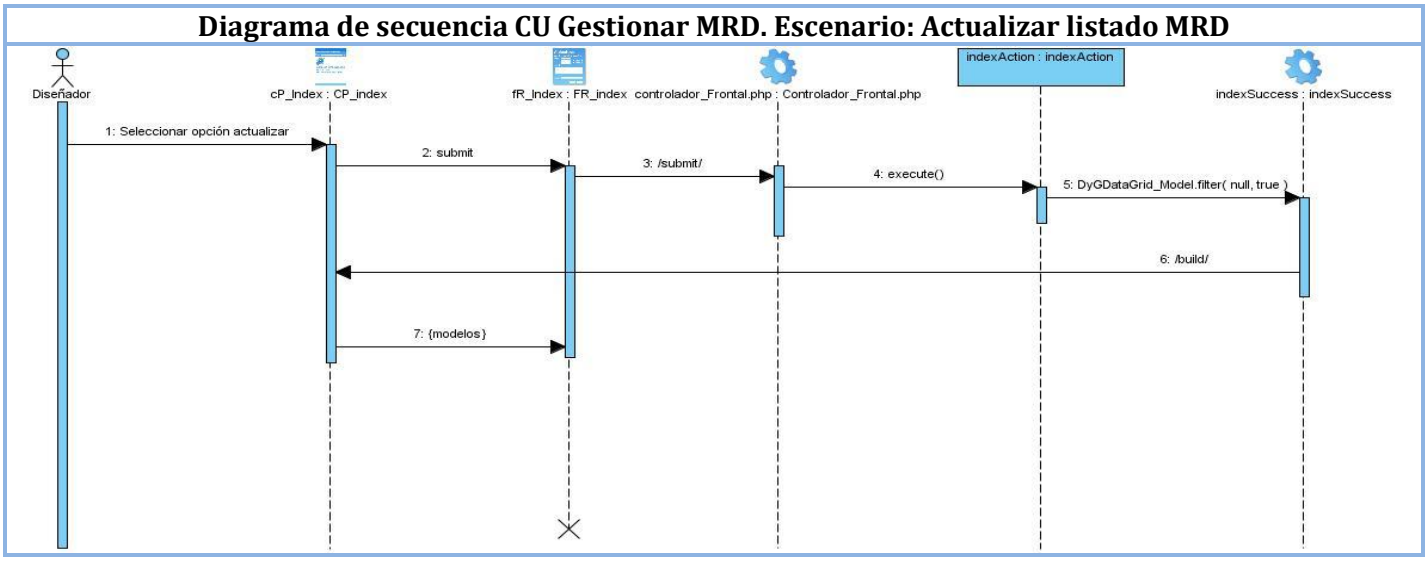


**Figura 23. Diagrama de secuencia CU Gestionar MRD. Escenario: Actualizar listado MRD.**

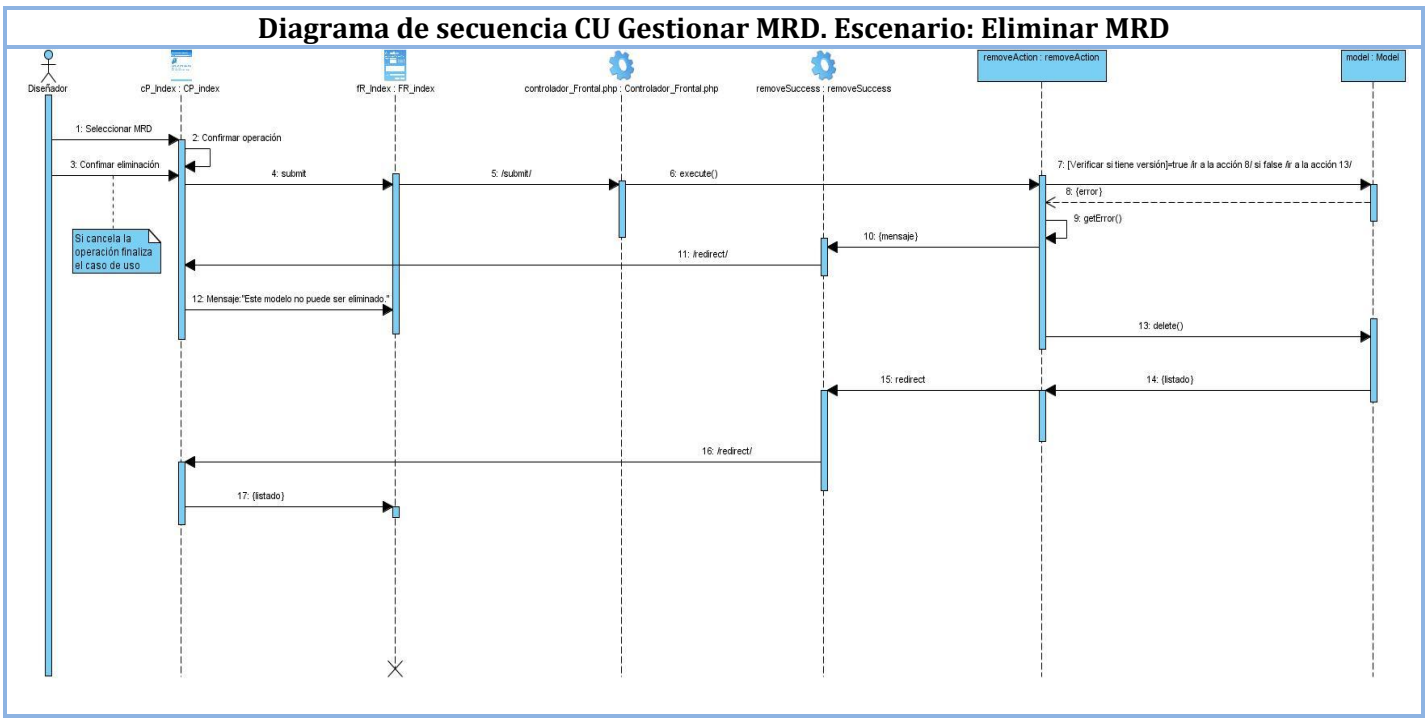


**Figura 24. Diagrama de secuencia CU Gestionar MRD. Escenario: Eliminar MRD.**

**Anexo 2:** Descripciones de las tablas de la base de datos.

| Tabla: model | | |
|---|---|---|
| **Descripción: Encargada de guardar los datos de los modelos.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| Id_model | INTEGER | Identificador de la tabla. |
| name | VARCHAR(255) | Nombre del modelo |
| title | VARCHAR(255) | Título del modelo. |
| description | VARCHAR(255) | Descripción del modelo. |

**Tabla 5. Descripción de la tabla model del Modelo de Dato.**

| Tabla: model_version | | |
|---|---|---|
| **Descripción: Encargada de guardar los datos de las versiones correspondientes a un modelo.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| Id_model_version | INTEGER | Identificador de la tabla. |
| Id_model | INTEGER | Identificador de la tabla model. |
| version | INTEGER | Número de versión. |
| state | VARCHAR(255) | Identificador de la tabla state. |
| create_date | DATE | Fecha de creación. |
| creator | VARCHAR(255) | Nombre del creador de la versión. |
| source | VARCHAR(255) | Todo el contenido de una versión. |

**Tabla 6. Descripción de la tabla model_version del Modelo de Dato.**

| Tabla: state | | |
|---|---|---|
| **Descripción: Encargada de guardar el estado de las versiones.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| state | VARCHAR(255) | Identificador de la tabla. |

**Tabla 7. Descripción de la tabla state del Modelo de Dato.**

| Tabla: data | | |
|---|---|---|
| **Descripción: Encargada de guardar los datos de los elementos que componen una versión.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| id_data | INTEGER | Identificador de la tabla. |
| name | VARCHAR(255) | Nombre del elemento. |
| type | VARCHAR(255) | Identificador de la tabla datatype. |
| required | BOOLEAM | Campo requerido. |
| min | VARCHAR(255) | Longitud mínima de caracteres. |

| | | |
|---|---|---|
| max | VARCHAR(255) | Longitud máxima de caracteres. |
| default_value | VARCHAR(255) | Valor por defecto. |
| constraints | VARCHAR(255) | Restricciones del tipo de dato. |
| description | VARCHAR(255) | Descripción del elemento. |
| order | INTEGER | Orden de un elemento. |
| id_model_version | INTEGER | Identificador de la tabla model_version. |

**Tabla 8. Descripción de la tabla data del Modelo de Dato.**

| **Tabla: datatype** | | |
|---|---|---|
| **Descripción: Encargada de guardar los tipos de datos de los elementos.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| type | VARCHAR(255) | Identificador de la tabla. |
| description | VARCHAR(255) | Descripción del elemento. |

**Tabla 9. Descripción de la tabla datatype del Modelo de Dato.**

| **Tabla: userinfo** | | |
|---|---|---|
| **Descripción: Encargada de guardar los datos de los usuarios.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| username | VARCHAR(255) | Identificador de la tabla. |
| full_name | VARCHAR(255) | Nombre completo del usuario. |
| mail | VARCHAR(255) | Dirección de correo electrónico. |
| password | VARCHAR(128) | Contraseña de acceso al sistema. |

**Tabla 10. Descripción de la tabla userinfo del Modelo de Dato.**

| **Tabla: userlog** | | |
|---|---|---|
| **Descripción: Encargada de registrar todas las acciones que realizan los usuarios.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| id_log | INTEGER | Identificador de la tabla. |
| username | VARCHAR(255) | Identificador de la tabla userinfo. |
| time_log | TIMESTAMP | Tiempo que transcurre entre cada acción. |
| action | VARCHAR(255) | Acciones que realiza el usuario. |

**Tabla 11. Descripción de la tabla userlog del Modelo de Dato.**

| **Tabla: group_members** | | |
|---|---|---|
| **Descripción: Encargado de registrar el rol que desempeña un usuario en el grupo.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |

| | | |
|---|---|---|
| id_group_members | INTEGER | Identificador de la tabla. |
| role | VARCHAR(255) | Rol del usuario en la aplicación. |
| group_name | VARCHAR(255) | Nombre del grupo al que pertenece. |
| username | VARCHAR(255) | Identificador de la tabla userinfo. |

**Tabla 12. Descripción de la tabla group_members del Modelo de Dato.**

| **Tabla: group_role** | | |
|---|---|---|
| **Descripción: Encargada de registrar los roles existentes en el sistema.** | | |
| **Nombre atributo** | **Tipo de dato** | **Comentario** |
| name | VARCHAR(255) | Identificador de la tabla. |
| description | VARCHAR(255) | Descripción del rol. |

**Tabla 13. Descripción de la tabla group_role del Modelo de Dato.**

# Glosario de Términos

**CSS:** Hojas de Estilo en Cascada (CSS por sus siglas en inglés). Es una tecnología que permite crear páginas web de una manera más exacta. Proporciona hacer cosas como incluir márgenes, tipos de letra, fondos, colores.

**DOM:** (Document Object Model o Modelo de objetos en documentos). Es una interface independiente de la plataforma y del lenguaje que permite que los programas y scripts tengan acceso dinámicamente y actualicen el contenido, la estructura y estilo de los documentos.

**EC:** Escenario. Abreviatura utilizada para referirse a los diferentes escenarios de una sección.

**HTTP:** Protocolo de Transferencia de Hipertexto (HTTP por sus siglas en inglés). Es un protocolo de comunicaciones usado en cada transacción de la web, es decir, este define la comunicación entre un servidor y otra máquina.

**HTTPS:** Protocolo Seguro de Transferencia de Hipertexto (HTTPS por sus siglas en inglés). Está basado en el HTTP, es decir, es la versión segura de este. Utilizado principalmente por entidades bancarias, tiendas en línea, y cualquier tipo de servicio que requiera el envío de datos personales que requieran más seguridad.

**LGPL:** Licencia Pública General Reducida (LGPL por sus siglas en inglés). Es un tipo de licencia que permite que el software licenciado bajo ella esté integrado en software's privativos.

**MVC:** Patrón de arquitectura de software que se utiliza mucho en aplicaciones web, separándola en tres capas el Modelo, la Vista y el Controlador.

**MRD:** Término referido a un Modelo de Recogida de Datos. Planilla con un estándar o formato previamente definido, que contienen un conjunto de variables y campos que pueden cambiar o actualizarse a partir de nuevas investigaciones.

**NC:** No Conformidad. Abreviatura utilizada para referirse a los errores detectados durante la aplicación de pruebas funcionales.

**PD:** Pendiente**.** Abreviatura utilizada para definir el estado de una No conformidad.

**Propel:** Propel es un servicio de objeto persistente y de consulta, lo que significa que provee un sistema para almacenar objetos en una base de datos y un sistema para búsqueda y restauración de objetos desde una base de datos.

**RA**: Resuelta. Abreviatura utilizada para definir el estado de una No conformidad una vez resuelta.

**RIA:** Aplicaciones Ricas de Internet (RIA por sus siglas en inglés). Estas son ejecutadas en el interior de un navegador y se caracterizan por presentar un comportamiento similar al de las de escritorio, el aspecto fundamental es que no se precisa una actualización de la página completa.

**SC:** Abreviatura de la palabra Sección utilizada para enumerar las secciones de los casos de pruebas.

**Script:** Pequeño programa para realizar efectos especiales en las páginas.

**TCP:** Protocolo de Control de Transmisión (TCP por sus siglas en inglés). Este permite a dos anfitriones establecer una conexión e intercambiar datos, garantiza su entrega, es decir, que los datos no se pierdan durante la transmisión.

**Tooltips:** Un tooltip es básicamente un cuadro que se puede rellenar con distinta información, el cual será visto por un usuario cuando posicione el puntero del mouse sobre un elemento en particular de un documento web. La gran utilidad de estos es presentar información extra sobre ciertos elementos.

**XML:** Lenguaje de Marcas Extensible (XML por sus siglas en inglés)**.** Es un metalenguaje, es decir, un lenguaje que se usa para hablar acerca de otro lenguaje, pues no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

**XMLHttpRequest:** Es una interfaz para realizar llamadas mediante http.

**XSLT:** Lenguaje de Transformación de Etiquetas Extensible de hojas de estilo (XSLT por sus siglas en inglés) Presenta una forma de transformar documentos XML en otros, incluso a formatos que no son XML.