

Universidad de las Ciencias Informáticas

Facultad 6



SIMDECC

**Título: Sistema de Manejo de Datos de Ensayos Clínicos
Cubano. Módulo Validación: Diseño del submódulo
“Derivación de las variables del Cuaderno de Recogida de
Datos”.**

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

**Autor(es): Marlien González Hernández
Jenely De Laosa Socarrás**

**Tutor(es): Ing. Lucía Rodríguez García
Ing. Aislein Blanco González**

Ciudad de La Habana, Junio 2008



“.....He aprendido que todo el mundo quiere vivir en la cima de la montaña, sin saber que la verdadera felicidad está en la forma de subir la escarpada....”

Gabriel García Márquez

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 _____.

Marlien González Hernández

Jenely De Laosa Socarrás

Firma del Autor

Firma del Autor

Ing. Lucía Rodríguez García

Aislein Blanco González

Firma del Tutor

Firma del Tutor

Datos de contacto

Tutora: Lucía Rodríguez García

Correo electrónico: lrodriguezg@uci.cu

Tutora: Aislein Blanco González

Correo electrónico: ablanco@uci.cu

Agradecimientos

- A Fidel Castro y a la Revolución por la increíble oportunidad de estudiar en esta universidad de excelencia y por tanta confianza depositada en nosotras.

-A nuestros padres, por nutrirnos día a día con todo el amor, el apoyo y la dedicación que un hijo puede desear. Por ser quienes en todos estos años se han mantenido a nuestro lado, alentándonos al esfuerzo continuo por alcanzar siempre metas más altas. Por deberles lo que somos y ver en ellos el faro que guía y alumbra el camino hacia el futuro.

-A nuestras tutoras, especialmente a Lu por ayudarnos desde el comienzo con su paciencia y carisma. Por ser ya para nosotras, más que tutora, amiga.

-Al profe Ballester, por inspirar en nosotras una gran admiración, por toda esa sabiduría a pesar de su juventud y por ser un libro abierto lleno de conocimientos para todos nosotros.

-A Richard, Osvaldo, Raúl, Yasiel, por su eterna paciencia, comprensión, por tener siempre una respuesta para todas nuestras preguntas. Por el cariño que han inspirado y lo que significan hoy para nosotras.

- A Rosy e Ide, por ser las hermanas que vivieron día a día nuestros logros y fracasos, alegrías y tristezas. Por formar parte de este cuarteto de “pijas” que ni tiempo ni distancia podrán destruir jamás. Recuerden que las queremos mucho.

- A las chicas del 67204, a Yayi, Yane, Evy, Keky, Ailyn, Yune, Zoila, Lorna, por su presencia a lo largo de estos años cada mañana al despertar y cada noche al finalizar el día. Por estar siempre al tanto de nuestras vidas, en lo personal y profesional. Por tantos recuerdos imborrables....

- A Adonis, Yaniel, Dayamis, a todos aquellos amigos que mañana serán parte insustituible de nuestros mejores años como estudiantes y siempre tuvieron presente

nuestro desempeño en esta etapa que no vuelve atrás.

- A nuestras familias, por ser la raíz de este árbol que hoy florece y mañana dará nuevos frutos. Por esperar siempre lo mejor de nosotras, alentarnos y estimular cada victoria alcanzada. Por confiar en nuestras decisiones aunque no siempre fueran las mejores. Por el espacio infinito dedicado a nuestros sueños.

- A todos nuestros seres queridos que durante estos años nos enseñaron que nada es imposible, que con esfuerzo y dedicación podemos alcanzar las metas más lejanas.

- Y más que nada, agradecemos a Dios, por todo cuanto tenemos y somos.

Dedicatoria

-A mi padre, por ser mi fuente inagotable de inspiración, el mejor de los consejeros y un amigo inigualable; por su amor sin comparación ni fronteras, por abrirme el camino a lo que hoy soy, por guiar cada uno de mis pasos a lo largo de los años y por las tantas veces que debí decir lo mucho que lo quiero.

-A mi madre, por ser la luz que ilumina mi mundo y mi tesoro más preciado, por su infinita ternura, sensibilidad y corazón de oro, por su entrega sin medidas ni descanso, por creer ciegamente en mí y regalarme la felicidad de vivir.

-A mi hermano, por su inmenso cariño y bondad sin igual, por motivarme al esfuerzo continuo y a llegar siempre lejos, por seguir día a día mi ejemplo, por ser la persona maravillosa que es.

-A Pedri, por los años de paciencia y amor a mi lado, por marcar mi vida y darle sentido a esta etapa inolvidable.

-A los que ya no están, por cuidarme, amarme y desear un día este sueño hecho realidad.

-A Ana María, Mailén, Anagel, Marlien, Rosayda e Idelsis, por ser más que amigas, las hermanas que no tuve.

-A todos mis seres queridos, los que están lejos y cerca, simplemente por estar siempre ahí.

- A mi mamá, por ser a cada instante más que una madre, por ser mi amiga, mi hermana, mi consejera. Por todo el cariño que día a día me regala, por todos sus esfuerzos por lograr mi desarrollo profesional, por guiarme en la vida y encontrar siempre en ella todo el apoyo y el cariño que necesito para seguir adelante.

- A mi papá, por ser mi ejemplo a seguir, por confiar siempre en mí, por acompañarme en los momentos más difíciles de mi vida. Por haberme brindado junto a mami un hogar lindo, lleno de amor y comprensión. Por todo el apoyo y la dedicación que recibí durante todos estos años.

- A Esmel, por ser ya una parte de mí. Por ser mi inspiración, por su constante exigencia demostrándome que no debemos conformarnos, que siempre se puede dar más. Por todo su amor y sacrificio, por enseñarme a amar y por estar siempre conmigo a pesar de la distancia que nos separó durante esta etapa.

- A mis abuelos Bertha y Musa que aunque ya no están, sé que hubiesen estado orgullosos de mí en este momento. A ellos gracias por quererme tanto y por dedicar parte de sus vidas a ayudarme a crecer, a complacerme en mis caprichos de niña y entregarme todo su amor sin pedir nada a cambio.

- A todos mis seres queridos, mis familiares, mis amigos, y los que de una forma u otra han contribuido a mi superación.

Jenely De Laosa Socarrás

Marlien González Hernández

Resumen

La presente investigación surge en el marco de trabajo del proyecto Ensayos Clínicos perteneciente al polo científico Gestión de Información Biomédica (GIB), en colaboración con el Centro de Inmunología Molecular (CIM) y la Universidad de las Ciencias Informáticas (UCI). En este Trabajo de Diploma se realiza el diseño de un submódulo que garantiza el proceso de validación de las variables de los Cuadernos de Recogida de Datos (CRD) para la aplicación web Sistema de Manejo de Datos de Ensayos Clínicos Cubano (SIMDECC), tomando como base los resultados arrojados por una investigación precedente.

El proyecto en cuestión se ha venido desarrollando durante algunos años en aras de obtener un aplicación web de gran importancia para el país, no solo por agilizar la recogida de datos de aquellos pacientes que padecen de cáncer, sino también porque en el mundo existen software dedicados a este fin, pero su costo elevado los sitúa lejos del alcance de naciones subdesarrolladas como es el caso de Cuba.

Palabras claves:

Ensayos Clínicos, diseño, validación, variables, Cuadernos de Recogida de Datos, submódulo.

Índice

Agradecimientos	I
Dedicatoria.....	III
Resumen.....	IV
Introducción	1
Capítulo 1: Fundamentación teórica	4
1.1 ¿Qué es un ensayo clínico?	4
1.2 ¿Qué es un CRD?	4
1.3 Módulo Validación.....	4
1.4 ¿Qué es la validación de las variables?.....	5
1.5 Derivación de variables del CRD	5
1.6 Situación actual.....	6
1.7 Sistemas informáticos vinculados al campo de acción.....	6
1.7.1 OpenClínica	7
1.7.2 Pivotal	7
1.7.3 Hypocrates.....	8
1.8 Metodologías, tecnologías y herramientas.....	8
1.8.1 Metodología	8
1.8.1.1 RUP.....	9
1.8.1.2 Lenguaje Unificado de Modelado (UML).....	12
1.8.2 Tecnologías	12
1.8.2.1 Lenguaje de programación	12
1.8.2.2 Servidor web	14
1.8.2.2.1 AOLServer.....	14
1.8.2.2.2 IIS (Information Server).....	15
1.8.2.2.3 Apache	15
1.8.3 Herramientas	16
1.8.3.1 Herramienta CASE.....	16
1.8.3.2 Komposer	18
1.8.3.3 Framework	19

1.8.3.3.1 Djanjo	20
1.8.3.3.2 Kumbia	20
1.8.3.3.3 Symfony	21
1.8.3.4 Herramienta para el control de versiones	23
Conclusiones	25
Capítulo 2: Diseño del Sistema.....	26
2.1 Modelo de Diseño	26
2.2 Patrones.....	27
2.2.3 Patrones de diseño.....	27
2.2.3.1 Patrón Singleton.....	28
2.2.3.2 Patrón Decorator.....	29
2.2.3.3 Patrón Fachada.....	29
2.2.3.4 Patrones GRASP	29
2.2.3.5 Más patrones para asignar responsabilidades.....	30
2.2.4 Patrón de arquitectura	31
2.2.4.1 Patrón Modelo-Vista- Controlador.....	31
2.2.5 Uso de los patrones en la investigación	32
2.3 Realización de los casos de uso del diseño	33
2.3.1 Vista Lógica	33
2.3.2 Descripción de las clases del diseño.....	33
2.3.3 Diagramas de Clases del Diseño	39
2.3.3.1 Estructura del diagrama de clases del diseño con la utilización de Symfony	39
2.3.4 Diagramas de Interacción.....	47
2.4 Prototipos no funcionales.....	60
2.5 Mapa de Navegación	69
2.6 Clases persistentes.....	70
2.6.1 Diagrama de clases persistentes.....	70
2.7 Modelo de Datos	71
2.8 Diagrama de Despliegue	72
2.9 Validación del diseño	73
Conclusiones	76

Conclusiones generales.....	77
Recomendaciones	78
Referencias bibliográficas.....	79
Bibliografía Consultada.....	81
Anexos	83
Glosario de términos.....	92

Índice de figuras

Figura 1 Estructura del proyecto "Ensayos Clínicos"	5
Figura 2 Rational Unified Process (RUP)	9
Figura 3 Unified Modeling Language (UML)	12
Figura 4 Servidor Apache	15
Figura 5 Visual Paradigm para UML	17
Figura 6 Integración del VP con los IDE	18
Figura 7 Integración del VP con Subversion	18
Figura 8 Komposer	19
Figura 9 Framework Symfony	21
Figura 10 El flujo de trabajo de Symfony	23
Figura 11 Patrón Modelo-Vista-Controlador	32
Figura 12 Diagrama de paquetes del diseño	33
Figura 13 Iniciar validación	40
Figura 14 Adicionar variables de dependencia	41
Figura 15 Eliminar variables de dependencia	42
Figura 16 Gestionar casos	43
Figura 17 Establecer condiciones	44
Figura 18 Derivar	45
Figura 19 Capa del Modelo	46
Figura 20 Listar modelos	47
Figura 21 Listar submodelos	48
Figura 22 Mostrar variables	49
Figura 23 Adicionar cantidad de variables de dependencia	50
Figura 24 Seleccionar variable de dependencia	51
Figura 25 Eliminar variables de dependencia	52
Figura 26 Adicionar cantidad de casos	53
Figura 27 Adicionar casos	54
Figura 28 Eliminar casos	55
Figura 29 Editar variables de dependencia	56
Figura 30 Establecer condiciones	57

Figura 31 Establecer condiciones para el tipo de dato nomenclador	58
Figura 32 Derivar	59
Figura 33 Caso de uso Iniciar validación: Listar modelos.....	60
Figura 34 Caso de uso Iniciar validación: Listar submodelos	61
Figura 35 Caso de uso Iniciar validación: Variables de un submodelo.....	62
Figura 36 Caso de uso Iniciar validación: variables y subvariables de un submodelo	63
Figura 37 Caso de uso Adicionar variables de dependencia.....	64
Figura 38 Caso de uso Gestionar casos.....	65
Figura 39 Caso de uso Establecer condiciones.....	66
Figura 40 Caso de uso Derivar	67
Figura 41 Resultado del caso de uso Derivar	68
Figura 42 Mapa de Navegación	69
Figura 43 Diagrama de clases persistentes.....	70
Figura 44 Modelo de datos	71
Figura 45 Modelo de despliegue de la aplicación	72

Índice de tablas

Tabla 2.1 Descripción de la clase del Diseño: valIniciarValidacionActions	33
Tabla 2.2 Descripción de la clase del Diseño: valAddVarDepActions	34
Tabla 2.3 Descripción de la clase del Diseño: valEliminarVarDepActions	35
Tabla 2.4 Descripción de la clase del Diseño: valGestionarCasosActions	35
Tabla 2.5 Descripción de la clase del Diseño: valDerivarActions	36
Tabla 2.6 Descripción de la clase del Diseño: valEstabCondicionesActions.....	37
Tabla 2.7 Descripción de la clase del Diseño: NomencladorComponents	38

Introducción

Se dice que vivimos en una era tecnológica, donde cada vez son más eminentes los avances que se experimentan. No son menos importantes los logros obtenidos en el ámbito científico. Sin embargo, la humanidad todavía no ha podido encontrar la cura para enfermedades que ponen en peligro su existencia.

Una de las mayores causas de muerte que enfrenta el hombre es el cáncer. La búsqueda de un tratamiento que pueda frenar este mal ha significado muchos años de arduo trabajo para miles de especialistas de todo el mundo. Cuba no está exenta de todas estas investigaciones para tratar de propiciarles una vida más estable y duradera a los pacientes que padecen de tan agobiante enfermedad. Con este objetivo fue creado el 5 de diciembre de 1994 el Centro de Inmunología Molecular cuyo propósito es mejorar la calidad de vida de la población cubana.

Actualmente en el CIM se lleva a cabo un proceso de desarrollo de nuevos fármacos utilizados fundamentalmente para aplicarles tratamientos a pacientes que padecen de cáncer y otras enfermedades mortales. Para ello se realizan los llamados Ensayos Clínicos (estudios clínicos) que consisten en la evaluación de estos fármacos a través de su aplicación a seres humanos.

Para la recogida de los datos de estos ensayos, los científicos se auxilian de los Cuadernos de Recogida de Datos, los cuales contienen todas las variables recogidas durante el proceso.

Debido a que toda esta información recogida se encuentra en papeles, a que hay un gran número de variables asociadas a un solo CRD, corriéndose el riesgo de que hayan muchos errores a la hora de entrar los datos, además del complejo proceso de validación de cada una de esas variables, por no existir un estándar para la validación de las mismas, se llevó a cabo una investigación perteneciente al módulo de Diseño del proyecto Ensayos Clínicos, la cual propuso una solución a estos problemas y cuyo alcance fue el Modelamiento del Negocio y el Levantamiento de Requerimientos Funcionales y no Funcionales.

Después de realizar un estudio de la investigación antes mencionada, específicamente de los requerimientos relacionados con la derivación de las variables en un CRD y analizando además la importancia de la culminación de este proyecto para el país y para los pacientes que padecen de cáncer, se plantea el problema científico siguiente:

¿Cómo traducir los requerimientos identificados en el módulo Validación: submódulo “Derivación de las variables del CRD” en elementos que puedan ser implementados?

El objeto de estudio definido es:

Los procesos de gestión de la información de aplicaciones web para Ensayos Clínicos.

A partir de este objeto de estudio se delimita el campo de acción de la investigación como:

Los procesos de gestión de la información de aplicaciones web para la derivación de las variables del CRD de los Ensayos Clínicos.

Luego de haber definido el problema científico y el campo de acción, se tiene como objetivo general:

Diseñar el submódulo: “Derivación de las variables del Cuaderno de Recogida de Datos” perteneciente al módulo Validación para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano en el CIM.

Para darle cumplimiento a este objetivo se trazan las **tareas** siguientes:

- Estudio de los resultados de la investigación precedente relacionados con la validación de las variables del Cuaderno de Recogida de Datos
- Estudio y selección de las herramientas a utilizar en el diseño del módulo Validación
- Estudio de los patrones de diseño a utilizar
- Estudio y selección del framework a emplear
- Realización de los casos de uso del diseño
- Elaboración del diagrama de clases persistentes
- Confección del Mapa de Navegación
- Realización de los Prototipos no funcionales
- Elaboración del diagrama de Despliegue
- Validación del diseño

A continuación se muestra una breve descripción de los capítulos de la presente investigación:

En el **Capítulo 1: Fundamentación teórica**, se realiza un estudio sobre el estado del arte de los procesos de gestión de la información de aplicaciones web para Ensayos Clínicos, siendo este el objeto de estudio de la investigación, así como la metodología, tecnologías y herramientas utilizadas para dar solución al problema planteado.

En el **Capítulo 2: Diseño del Sistema**, se realiza el diseño del submódulo “Derivación de las variables del CRD”, con el objetivo de crear un modelo físico que sirva como entrada al modelo de implementación. Con el diseño se logra un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, o sea, cómo cumple el sistema sus metas. La realización de este capítulo es de suma importancia puesto que de él depende que la futura aplicación se implemente sin ambigüedades.

Capítulo 1: Fundamentación teórica

El presente capítulo brinda el estado del arte de los procesos de gestión de la información de aplicaciones web para Ensayos Clínicos, siendo este el objeto de estudio de la investigación, así como la metodología, tecnologías y herramientas utilizadas para dar solución al problema planteado.

1.1 ¿Qué es un ensayo clínico?

Tipo de estudio clínico en el que se evalúan nuevos fármacos o tratamientos médicos que a través de su aplicación a seres humanos pretende valorar la eficacia y seguridad de los mismos con un protocolo de investigación estrictamente controlado, permite a los médicos determinar si un nuevo tratamiento, medicamento o dispositivo contribuirá a prevenir, detectar o tratar una enfermedad.

Un ensayo clínico medicamentoso es toda evaluación experimental de una sustancia o medicamento, a través de su administración o aplicación a seres humanos, orientada a alguno de los siguientes fines:

- *Poner de manifiesto sus efectos farmacodinámicos o recoger datos referentes a su absorción, distribución, metabolismo y excreción en el organismo humano.*
- *Establecer la eficacia para una indicación terapéutica, profiláctica o diagnóstica determinada.*
- *Conocer el perfil de sus reacciones adversas y establecer su seguridad. (1)*

1.2 ¿Qué es un CRD?

Para la realización de los ensayos clínicos se maneja gran cantidad de información relacionada con los datos personales del paciente, además de un conjunto de exámenes que tienen como objetivo analizar el estado de la enfermedad y de este modo dar tratamiento a la misma. Estos datos solicitados por un protocolo son recogidos en un documento donde se registran y organizan por modelos, que a su vez son constituidos por submodelos portadores de las variables relacionadas con un paciente, representando un historial de su enfermedad y una prueba fehaciente de la calidad de un posible producto farmacéutico. Este documento donde es recogida toda la información se denomina Cuaderno de Recogida de Datos.

1.3 Módulo Validación

El proyecto Ensayos Clínicos perteneciente al polo Gestión de Información Biomédica de la facultad 6, surge con el objetivo de informatizar la gestión de la información de un Cuaderno de Recogida de

Datos específicamente para el Centro de Inmunología Molecular. El mismo consta de 5 módulos y 5 submódulos:



Figura 1 Estructura del proyecto "Ensayos Clínicos"

El módulo Validación está dividido en dos submódulos:

1. "Derivación de las variables del Cuaderno de Recogida de Datos" (Validación 1).
2. "Validación de las variables en el Cuaderno de Recogida de Datos y el control de errores" (Validación2).

La presente investigación se centra en el primer submódulo encargado de la derivación de las variables del CRD.

1.4 ¿Qué es la validación de las variables?

La validación de las variables de un CRD consiste en establecer reglas que definan el valor de los datos en los cuadernos, y se efectúa con el objetivo de que la información almacenada en el CRD contenga la menor cantidad de errores posible.

El registro de los datos en el cuaderno puede efectuarse de dos formas diferentes: el dato correspondiente a una variable puede ser entrada directamente por el usuario en el CRD o la variable puede obtener un valor determinado que haya sido derivado a partir del valor de otras variables, este último proceso de recogida de datos se realiza a través de la derivación de variables.

1.5 Derivación de variables del CRD

El valor de una variable puede depender del valor de otras variables de diferentes formas, por ejemplo, una variable puede tomar el valor dado por la concatenación de otras tres variables que hayan sido

entradas anteriormente al CRD, o puede tomar el valor dado por una fórmula realizada entre otras dos variables del CRD. De esta forma el valor de una variable no será entrado directamente por un usuario en el cuaderno, sino que será derivado a partir del valor de otras variables.

Así mismo, la forma en que será derivado el valor de una variable está determinada por el tipo de dato de la misma y por el tipo de dato y la cantidad de variables de las cuales depende. Este proceso de dependencia entre los valores de las variables que permitirán derivar un dato a partir de otros, constituye la derivación de las variables del CRD, la cual tiene como objetivo definir un conjunto de reglas que permitan obtener el valor de una variable de forma correcta.

La presente investigación pretende diseñar un submódulo para una aplicación web que sirva como herramienta para obtener un conjunto de reglas de derivación previamente definidas con vista a registrar los datos en el CRD con la menor cantidad de errores posible.

1.6 Situación actual

El proceso de recogida de datos de todos los pacientes para un ensayo clínico, se realiza a través del Cuaderno de Recogida de Datos, como se expresó anteriormente, lo que no significa que esto disminuye la complejidad del trabajo, pues se gana en organización pero los riesgos que se corren son muchos, incluyendo lo tedioso y agotador que resulta la tarea. Esto se debe a que toda la información existente se encuentra en papeles, existe un gran número de variables asociadas a un solo CRD, por lo que la posibilidad de que se cometan errores es mayor y el proceso de validación de estas variables se vuelve complejo dada la falta de estandarización para validar las variables del CRD.

Para resolver todas estas dificultades un grupo de analistas pertenecientes al módulo Validación del proyecto Ensayos Clínicos se reunió en torno a desarrollar una investigación que concluyó con el Modelamiento del Negocio y el Levantamiento de Requisitos, arrojando varios casos de uso del sistema y un prototipo no funcional. Dada la situación presentada y la importancia acreditada a la terminación del proyecto se presenta la problemática a resolver en la actual investigación:

¿Cómo traducir los requerimientos identificados en el módulo Validación: submódulo “Derivación de las variables del CRD” en elementos que puedan ser implementados?

1.7 Sistemas informáticos vinculados al campo de acción.

En el mundo no existe una herramienta que permita al usuario obtener reglas para la derivación de una variable, puesto que los sistemas informáticos que tratan los Ensayos Clínicos en la actualidad

abordan la validación de los datos registrados de forma sencilla, por lo general a través de código JavaScript, pero no responden a la necesidad de establecer reglas para el correcto almacenamiento de la información proveniente del estudio de los pacientes. OpenClínica, Pivotal, Hypocrates, entre otros, son sistemas que tratan la gestión de la información relacionada con los ensayos clínicos, pero no cuentan con una herramienta capaz de ofrecer al usuario la posibilidad de seleccionar las reglas adecuadas para validar una variable.

1.7.1 OpenClínica

OpenClínica es una plataforma de código abierto para la recogida de datos electrónicos usados en investigaciones clínicas y en el diseño de cuadernos para la captura de datos de forma tal que pueden ser creados como se desee sin tener conocimientos de programación. Proporciona además una interfaz fácil de usar para la presentación y validación de los datos facilitando su uso a los médicos e investigadores que participan en la inscripción de los pacientes. Permite la extracción de datos y filtrado de archivos para ser empleados por los investigadores, estadísticos, y los directores de estudio, así como la administración de cuentas de usuario, presentación de informes por los administradores, auditorías, y configuración. Posibilita el intercambio de recursos de una forma segura y transparente y el manejo de los datos de importación y exportación de herramientas para la migración de datos clínicos en las hojas de cálculo Excel y bases de datos locales. Ofrece amplias interfaces para la consulta y recuperación de datos, el uso de guías de estudios específicos de las funciones y privilegios de usuario, la encriptación SSL y el control de acceso y cambios por los usuarios. Cuenta con una robusta infraestructura de tecnología escalable y desarrollada utilizando Java J2EE, con bases de datos relacionales PostgreSQL incluidas (de código abierto) y Oracle 10G para apoyar las necesidades de la empresa de investigación clínica.

1.7.2 Pivotal

Pivotal es una consultoría médico-farmacéutica y de investigación clínica, especializada y organizada en áreas terapéuticas: oncología, cardio-vascular, endocrinología, medicina interna, antiinfecciosos, psiquiatría, neurología, reumatología y enfermedades óseas.

Servicios que brinda:

- *Diseño e implementación de bases de datos de ensayos clínicos*
- *Diseño y programación de rutinas de validación de datos con PL/SQL y Perl-SQL*
- *Diseño de CRDs en papel o electrónicos*
- *Entrada de datos: doble entrada., con validación por tercera persona.*

- *Adaptación a los sistemas del cliente entrando en su intranet mediante una conexión por red privada segura y autenticada, con sistemas de emulación de terminal. (2)*

Pivotal proporciona una herramienta fácil de usar para la recogida y transmisión de datos, así como también se ha dotado de sistemas de seguridad, de manera que consigue un posterior procesamiento fiable.

1.7.3 Hypocrates

Hypocrates es un sistema de archivo informático de Historias Clínicas.

Se usa para el control burocrático y archivo total de consultas médicas adaptado a la dinámica habitual de una consulta médica y sin necesidad de conocimientos informáticos.

Aporta un sistema de registro adaptado, perfectamente, al tiempo y mecánica de la consulta médica. Registra absolutamente toda la actividad de consulta, datos, historia, pruebas complementarias, prescripciones (...) de los enfermos.

Sin tener conocimiento alguno de informática, usando el programa, se pueden informatizar todas las actividades burocráticas y de archivo de la consulta. (3)

Una vez analizados estos sistemas, se concluye que no son realmente factibles para utilizar en el proyecto, puesto que no cumplen con las funcionalidades requeridas por el polo científico al presentar limitaciones en cuanto al manejo de datos y no definir reglas que eviten la entrada incorrecta de datos en los Ensayos Clínicos.

De ahí que se ratifica la necesidad de desarrollar un sistema para el país, que a la vez sea libre, y garantice la fiabilidad de los datos en el CRD para Ensayos Clínicos y para ello se proponen herramientas, tecnologías y metodologías que permitan obtener el producto que se desea.

1.8 Metodologías, tecnologías y herramientas.

1.8.1 Metodología

El mundo de la informática no para de hablar de procesos de desarrollo, el modo de trabajar eficientemente para evitar catástrofes que llevan a que un gran porcentaje de proyectos se termine sin éxito. El objetivo de un proceso de desarrollo es elevar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso. Hay que producir lo esperado en el tiempo esperado y con el coste esperado.

Todo desarrollo de software es complicado y difícil de controlar, más aún cuando no se tiene en cuenta

una metodología adecuada capaz de arrojar resultados satisfactorios para el cliente.

En la actualidad se cuenta con varias metodologías utilizadas indistintamente en dependencia del grado de complejidad y dimensión del proyecto en cuestión.

Tres de las metodologías más importantes que existen son: RUP, XP y MSF.

Para el diseño del submódulo se reafirma la decisión de utilizar el Rational Unified Process (RUP) propuesto por la investigación precedente, como la más adecuada según los requerimientos planteados por el cliente, ya que si bien el proceso de desarrollo del software es un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema, así también el proceso unificado es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software.

1.8.1.1 RUP

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML. (4)

RUP define las actividades en grupos lógicos representados en 9 flujos de trabajo y 4 fases.

Las fases son:

- *Inicio: El objetivo en esta etapa es determinar la visión del proyecto.*
- *Elaboración: En esta etapa la meta es determinar la arquitectura óptima.*
- *Construcción: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.*
- *Transmisión: El propósito es llegar a obtener el release del proyecto. (4)*

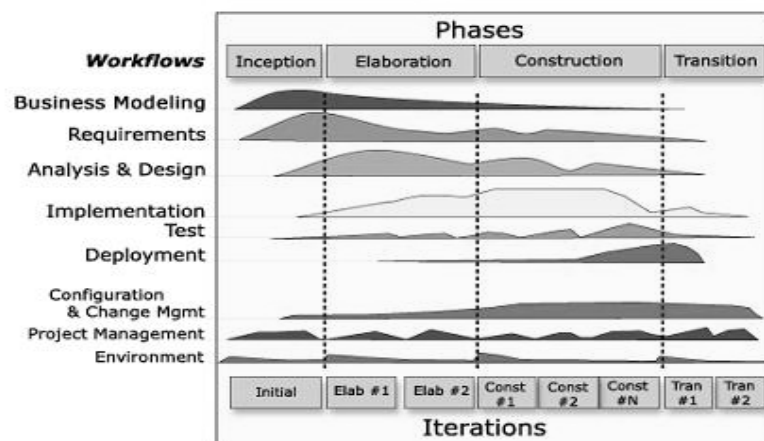


Figura 2 Rational Unified Process (RUP)

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, lo cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en

función de la evaluación de las iteraciones precedentes.

Los flujos de trabajo son:

- *Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.*
- *Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.*
- *Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.*
- *Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.*
- *Prueba (Testeo): Busca los defectos a lo largo del ciclo de vida.*
- *Instalación: Produce release del producto y realiza actividades (empaquete, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.*
- *Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.*
- *Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.*
- *Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización. (4)*

Los principales elementos que define RUP son:

- *Trabajadores (“quién”)*
- *Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.*
- *Actividades (“cómo”)*
- *Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.*

- *Artefactos ("qué")*
- *Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.*
- *Flujo de actividades ("Cuándo")*
- *Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable. (4)*

Roles y artefactos

Como se planteó anteriormente RUP propone 9 flujos de trabajo y para cada uno de ellos se definen roles y artefactos. A continuación se muestran los roles que serán desempeñados en el presente trabajo, así como los artefactos a realizar.

Análisis y Diseño:

Roles:

- Arquitecto
- Diseñador.
- Diseña de interfaz de usuario.
- Diseñador de base de datos.

Artefactos:

- Clases de diseño.
- Realización de los casos de uso del diseño.
- Vista lógica.
- Diagrama de clases persistentes.
- Modelo de Datos.
- Prototipos de interfaz de usuario
- Mapa de navegación.
- Diagrama de despliegue

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Por todo lo anteriormente planteado se reafirma la utilización de dicha metodología para guiar el proceso de desarrollo del software.

1.8.1.2 Lenguaje Unificado de Modelado (UML)

A lo largo de los años, el desarrollo de los proyectos de software causan grandes confusiones y malas interpretaciones en los requerimientos de los clientes y usuarios, en parte debido a la abundancia de notaciones, metodologías y conceptos que hacen que los desarrolladores de sistemas no se pongan de acuerdo en qué es lo que realmente están elaborando. En un esfuerzo para estandarizar las notaciones y procesos a utilizar, se conformó el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) siendo hoy el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir.



Figura 3 Unified Modeling Language (UML)

Se puede aplicar además en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el RUP) pero no especifica en sí mismo qué metodología o proceso usar.

1.8.2 Tecnologías

1.8.2.1 Lenguaje de programación

Para llevar a cabo la implementación del submódulo que formará parte del sitio web resultante se mantiene el uso de PHP 5.2.0 como lenguaje de programación potente y eficaz.

Usamos PHP por 10 razones fundamentales enunciadas a continuación:

1.- La Comunidad PHP

Dado el hecho de poseer una gran comunidad de desarrolladores es posible encontrar documentación en cualquier lugar, así como respuestas a determinados problemas que se puedan presentar relacionadas con el lenguaje. Dígase foros de discusión online, ejemplos de códigos, tutoriales, videos con descarga gratis para el auto aprendizaje, entre otros.

2.- Fácil aprendizaje

Su sintaxis primaria está basada en Perl por lo que resulta ser semejante a C y Java. Tiene librerías especializadas en determinadas funcionalidades que hacen más sencillo el trabajo del programador.

3.- Alto Rendimiento

PHP se destaca por su gran eficiencia, usando un modesto servidor capaz de atender millones de peticiones al día.

4.- Bajo Costo

Una de las mayores ventajas del PHP es precisamente su carácter gratuito

5.-Es Open Source

PHP es Open Source es decir que se tiene acceso al código fuente, se puede modificar en dependencia de las funcionalidades que se necesiten. Además, al ser muchos los desarrolladores que tiene acceso al código es posible mejorar el lenguaje.

6.- Librerías incluidas

PHP fue creado para el trabajo en aplicaciones web, por lo que incluye un conjunto de funciones y librerías para este tipo de tareas. Se puede conectar con base de datos, a web services, enviar email, generar imágenes, etc.

7. - Portabilidad

PHP está disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros. Una vez desarrollada una aplicación PHP esta puede funcionar para cualquiera de estos sistemas operativos sin necesidad de modificar el código.

8. - Soporte para POO

La versión 5 de PHP está diseñada para el soporte de características de programación orientada a objetos, características como herencia, métodos y atributos públicos o privados, clases y métodos abstractos, constructores, interfaces y destructores.

9. - Soporte para gran variedad de Bases de Datos

PHP tiene soporte para conectarse a una gran variedad de base de datos como: MySQL, PostgreSQL, mySQL, Oracle entre otras.

10. -Soporte

Para los soportes que se puedan necesitar en el trabajo con PHP, existe la empresa patrocinadora Zend Technologies, la cual se encarga de ofrecer versiones comerciales y mejoras al lenguaje.

1.8.2.2 Servidor web

Un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios, que funciona en la máquina y maneja la entrega de los componentes de las páginas web como respuesta a peticiones de los navegadores de los clientes.

1.8.2.2.1 AOLServer

AOLServer al igual que Apache es un producto open-source pero a diferencia de este último, fue diseñado conociendo varias deficiencias que existían en el modelo inicial utilizado por Apache.

AOLServer desde sus versiones iniciales fue desarrollado con "Threading" en mente, esto es, compartir la memoria del Proceso general en varios sub-procesos o "Threads", esto no solo eficientiza las conexiones al servidor de páginas sino también reduce la carga sobre el mismo.

Además de "Threading" AOLServer integra un Interpretador en su estructura interna evitando generar un proceso nuevo por aplicación de servidor y mantiene grupos de conexiones latentes ("pools") hacia bases de datos también para evitar generar procesos nuevos.

Otra ventaja de AOLServer es el ofrecimiento de ADP ("Aol Dynamic Pages") que son muy similares a las ASP's (Active Server Pages) de Microsoft o JSP's (Java Server Pages) de Sun, la diferencia estriba que ADP's utilizan el lenguaje Tcl y un API especialmente diseñado para acceder los elementos del servidor, pero su funcionamiento es igual al de ADP y JSP: mezclar elementos de HTML con elementos de programación para generar contenido dinámico.

Como último punto es utilizado por una de las empresas con mayor tráfico en Internet América Online, el proveedor de Servicios de Internet (ISP) más grande del mundo. Un poco más en Ventajas y Desventajas comparado con Apache. (9)

1.8.2.2.2 IIS (Information Server)

IIS es el servidor de páginas desarrollado por Microsoft para Windows NT/2000. IIS solo puede operar en plataformas Windows. El punto más favorable de este servidor son ASP's que facilitan el desarrollo de aplicaciones y la "sencillez" de instalación, sin embargo, existen alternativas como ADP's de AOLServer y JSP's para Java. Desafortunadamente debido a la presencia de Microsoft en el Mercado seguirá siendo necesario interactuar con este producto a pesar de todas sus desventajas:

- *Plataforma: Solo esta disponible para Windows. Historia de Sistemas Operativos para Red y Porque es mas fácil y Económico configurar Unix que Windows en Red*
- *Costo: Porque pagar licencia si existen productos flexibles y open-source mejores.*
- *Confiabilidad: Menos confiable que otros productos , tan confiable que ni sus mejores técnicos podían utilizarlo cuando se encontraba bajo uno de los tantos ataques que sufren sitios de Internet*
- *Seguridad: Aún plagado de fallas en versiones de producción. (9)*

1.8.2.2.3 Apache

El servidor web por excelencia es el Apache, por su estabilidad, robustez, excelencia, y configurabilidad que incondicionalmente reiteran la confianza en este programa. *Apache es una muestra, al igual que el sistema operativo Linux de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar.*(10)



Figura 4 Servidor Apache

Muchas de las facilidades que brinda este servidor web se muestran a continuación:

- *Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.*
- *Apache es una tecnología gratuita de código fuente abierta: El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos*

instalando como servidor, lo podemos saber, sin ningún secreto, sin ninguna puerta trasera.

- *Apache es un servidor altamente configurable de diseño modular: Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un modulo para realizar una función determinada.*
- *Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de script: Perl se destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.*
- *Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.*
- *Tiene una alta configurabilidad en la creación y gestión de logs: Apache permite la creación de ficheros de log a medida del administrador.(10)*

Por todo lo antes expuesto se propone la utilización del Apache 2.2.3 como servidor web a utilizar para el desarrollo de la futura aplicación.

1.8.3 Herramientas

1.8.3.1 Herramienta CASE

Existen varias herramientas creadas para el desarrollo de la Ingeniería de Software, con el fin de desarrollar programas, utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automatizadas. Ejemplo de ello, son las herramientas CASE (Computer Aided Software Engineering), las cuales constituyen un conjunto de ayudas para el desarrollo de programas informáticos.

Uno de los objetivos más importante de las herramientas CASE (a largo plazo) es conseguir la generación automática de programas desde una especificación a nivel de diseño.

Entre las herramientas CASE orientadas a UML se encuentran:

- Rational Rose
- ArgoUML

- Poseidón
- Visual Paradigm
- MagicDraw UML
- Borland Together

En la investigación precedente se propuso como herramienta CASE para la modelación visual al Rational Rose Enterprise Edition de la Suite 2003, por ser una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Sin embargo en el marco del proyecto Ensayos Clínicos se determinó un cambio de la herramienta CASE, puesto que no sería conveniente el uso del Rational, al no ser un software libre, y requerir de una licencia para su empleo, lo cual resulta poco favorable. Además se necesita una herramienta multiplataforma que permita su uso en cualquier sistema operativo. Por ello se decidió emplear el Visual Paradigm (VP) en el proyecto, utilizándose por consiguiente en la investigación.



Figura 5 Visual Paradigm para UML

Esta potente herramienta multiplataforma, utiliza UML como lenguaje de modelado, permitiendo una rápida construcción de las aplicaciones con alta calidad. Además permite dibujar todos los tipos de diagramas de clases, código inverso y generar códigos desde diagramas.

Visual Paradigm ofrece una integración sin fisuras de la herramienta de modelado UML con todos los IDEs como se muestra en la figura 6 (Visual Studio, Eclipse, Borland JBuilder, NetBeans / Sun ONE, IntelliJ IDEA, Oracle JDeveloper, BEA WebLogic Workshop), logrando así un mando unificado de modelado y entorno de desarrollo. Independientemente del IDE utilizado, Visual Paradigm facilita la navegación intuitiva entre el código visual y el modelado, así como una potente sincronización en tiempo real.

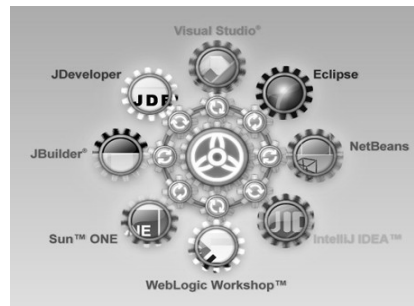


Figura 6 Integración del VP con los IDE

Visual Paradigm además permite una integración con Subversion (SVN), popular sistema de control de versiones que almacena centralmente los artefactos y realiza un seguimiento de los cambios realizados sobre un proyecto. Los desarrolladores lo utilizan para facilitar el modelado simultáneo, almacenar los archivos de proyectos y hacer un seguimiento de los cambios.



Figura 7 Integración del VP con Subversion

1.8.3.2 Komposer

La investigación precedente propuso como herramienta a usar para el diseño de interfaces el Dreamweaver por permitir a sus usuarios diseñar, desarrollar y mantener de forma eficaz sitios y aplicaciones web basadas en normas y estándares internacionales y ser una de las más completas y estandarizadas a nivel mundial. Sin embargo una desventaja significativa radica en que es propietario y esto es una traba para el caso del SIMDECC. De ahí que el arquitecto del proyecto Ensayos Clínicos determinara el uso del Komposer como editor web por ser una herramienta visual libre y multiplataforma, para aplicaciones web con soporte XHTML, HTML y CSS. Sus desarrolladores mantienen HandCoder : una extensión que añade soporte para ASP/JSP/PHP. Por tanto se selecciona el Komposer como la herramienta a emplear para el diseño de interfaz de usuario en la investigación.



Figura 8 Komposer

Entre las características principales de esta herramienta se encuentran:

- **Edición WYSIWYG** (Lo que ves es lo que obtienes).

Aún sin tener conocimientos de HTML es posible crear una web atractiva de forma sencilla a través de interfaz gráfica. Esta característica es similar a la ofrecida en Dreamweaver o FrontPage.

- **Manejo de archivos por FTP**

Subir los archivos a un servidor web es sumamente sencillo, incluso se puede navegar en la estructura de archivos del hosting para abrir y guardar archivos directamente de ahí.

- **Multiplataforma**

Hay versiones para Windows, Linux y Mac.

- **Código HTML confiable**

Incorpora código HTML eficiente y apegado al estándar, que lo hace capaz de trabajar con los más populares buscadores de hoy. Tiene un potente soporte para formularios, tablas y templates.

- **Manejo de múltiples archivos**

Facilita el manejo de múltiples archivos a través de fichas (tabs).

- **Es 100% OpenSource**

1.8.3.3 Framework

En el desarrollo de un software, un framework es una estructura de soporte definida con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura de sus proyectos con mayor rapidez. Es el esqueleto sobre el cual varios objetos son integrados para lograr solución determinada y puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a unir los diferentes componentes de un proyecto. Su uso proporciona la facilidad a los ingenieros de

emplear mayor tiempo identificando requerimientos de software, en lugar de tratar con los tediosos detalles de bajo nivel de proveer un sistema funcional. Existen varios framework para obtener aplicaciones web, algunos de ellos se abordan en el presente capítulo.

1.8.3.3.1 Django

Django es un marco de trabajo de alto nivel, para el desarrollo de aplicaciones Web en el lenguaje de programación Python. Con Django, los programadores Linux podrán crear rápidamente aplicaciones Web elegantes y de alto rendimiento, pues se enfoca en la mayor automatización posible y se adhiere al principio DRY.

Entre las principales características de Django se encuentran:

- *Mapeo Objeto-Relacional.*
- *Interfaces de administración automáticas listas para ambientes de producción.*
- *Diseño de URLs elegante.*
- *Sistema de plantillas.*
- *Sistema de Cache.*
- *Soporte de internacionalización. (6)*

1.8.3.3.2 Kumbia

Kumbia es un web framework libre escrito en PHP5. Basado en las mejores prácticas de desarrollo web, usado en software comercial y educativo, fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer. Utiliza la arquitectura MVC y otros patrones de programación como ActiveRecord y TemplateView.

Sus principales características son:

- *Sistema de Plantillas sencillo.*
- *Administración de Cache.*
- *Scaffolding Avanzado.*
- *Modelo de Objetos y Separación MVC.*
- *Soporte para AJAX y Efectos visuales.*
- *Generación de Formularios.*
- *Componentes Gráficos.*
- *Seguridad. (7)*

Entre sus principales ventajas, Kumbia es sencillo de aprender y usar, incluso obtener una aplicación funcional se consigue en breve tiempo, además de que cuenta con herramientas que garantizan mayor agilidad en el trabajo. Al usar el patrón Modelo-Vista-Controlador separa la presentación de la lógica y los datos, logrando aplicaciones fáciles de crecer y mantener.

Ofrece la posibilidad de emplear los Helpers, aportando un código con menor cantidad de errores y bastante claro, puesto que evita el uso del código HTML, haciendo más rápida la programación de las páginas. A esto se une la documentación en español y el uso sin mucha complicación de los patrones de diseño.

1.8.3.3 Symfony

*Symfony ha sido probado en numerosos proyectos reales y se utiliza actualmente para el desarrollo de sitios Web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows (8).*

Es libre, usado y recomendado por diseñadores de aplicaciones potentes y conocidas como Yahoo. Incluye un gran número de componentes, librerías, plugins, los llamados Helpers, que a partir del código PHP permiten generar todo el código cliente (CSS, JavaScript, HTML). Permite la división del sistema por módulos, y da buen soporte para pruebas unitarias y funcionales. Como patrones de diseño incluye los GRASP (Patrones de Asignación de Responsabilidades) y el Decorator.

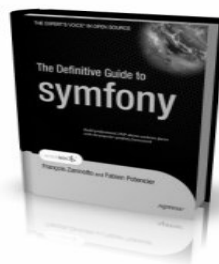


Figura 9 Framework Symfony

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- *Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).*
- *Independiente del sistema gestor de bases de datos utilizado.*
- *Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.*
- *Basado en la premisa de “convención por encima de configuración”, de modo que el desarrollador solo debe configurar aquello que no es convencional.*
- *Se adapta con la mayoría de las mejores técnicas y patrones de diseño para la Web*
- *Preparado para aplicaciones empresariales, y adaptable a sus políticas y arquitecturas, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.*
- *Código fácil de leer que incluye comentarios de phpDocumentor, permitiendo un sencillo mantenimiento.*
- *Extensible, lo que permite su integración con las librerías de otros fabricantes.*
- *Integración con AJAX. (8)*

Es de suma importancia para la investigación, la utilización del framework Symfony, desarrollado completamente con PHP 5 y diseñado para la optimización de aplicaciones web y sustitución de las tareas de codificación reiterativas. Symfony añade una nueva capa encima del lenguaje PHP proporcionando herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación compleja. Utiliza el patrón Modelo-Vista-Controlador separando la lógica del negocio de la lógica del servidor y de la capa de presentación, lo cual brinda independencia absoluta en el trabajo de los desarrolladores, pues un cambio en cualquiera de las tres capas no afectaría el código de las restantes.

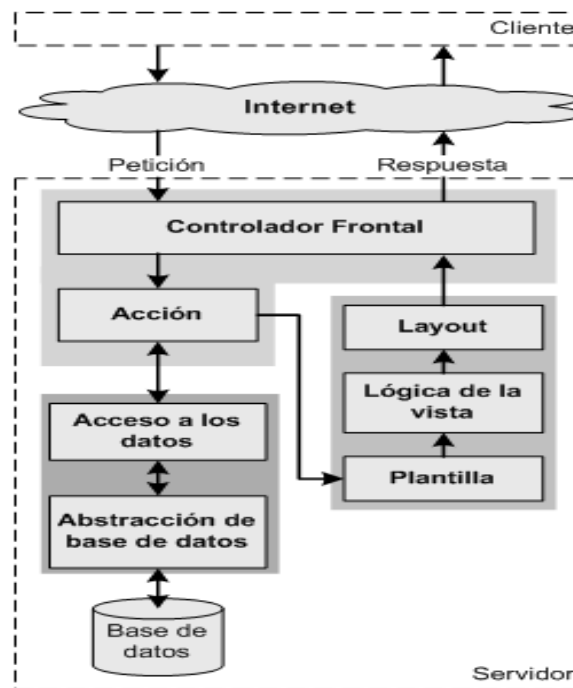


Figura 10 El flujo de trabajo de Symfony

Además de este patrón arquitectónico, Symfony implementa el patrón de diseño Decorator, usado fundamentalmente para agregar funcionalidad de adorno a un objeto individual de forma dinámica y transparente. A todo esto se adiciona que usa lo mejor de otros frameworks e incluso con mejor desarrollo y soporte con respecto a algunos muy buenos como Kumbia. Posee además mayor documentación y actualización, unido al hecho de que a diferencia de algunos, es un framework que va constantemente mejorando, integrando nuevas funcionalidades. Usa herramientas potentes para un mejor trabajo como es el caso de Propel, una de las mejores capas de abstracción de objetos/relacional disponibles en PHP 5 para el mapeo de objetos a bases de datos y que crea el esqueleto o estructura básica de las clases, generando automáticamente el código necesario.

Por todas las ventajas anteriormente expuestas se decide utilizar el framework Symfony 1.0.10 para realizar el diseño del submódulo “Derivación de las variables del CRD”.

1.8.3.4 Herramienta para el control de versiones

Se llama control de versiones a los métodos y herramientas disponibles para controlar todo lo referente a los cambios en el tiempo de un archivo.

Se propone mantener el uso de Subversion 1.4.2 por las siguientes razones:

➤ **Fuerte integración con Apache**

Permite definir controles de acceso avanzados y navegación vía web para consultar el depósito de archivos.

➤ **Transparencia al eliminar y cambiar nombres de archivos**

Contempla la deficiencia de intervención manual en el depósito y la corrige con éxito.

➤ **Copias ligeras sobre ramificaciones**

Subversion independientemente del número de ramificaciones creadas mantiene un árbol diferencial de cambios, que minimiza así el espacio consumido en el repositorio.

➤ **Copias diferenciales de archivos binarios**

Basado en el mismo principio de copias ligeras, Subversion es capaz de mantener un control diferencial sobre cualquier archivo binario del depósito reduciendo el consumo de espacio.

Conclusiones

En este capítulo se presenta una panorámica sobre la situación actual de los sistemas que tratan la validación en el mundo, así como la importancia de obtener de un submódulo que contribuya a establecer reglas de derivación para las variables de un CRD. Se define el objetivo de la investigación, así como los roles a desempeñar, artefactos a realizar y la metodología, tecnologías y herramientas a utilizar en el diseño de la aplicación.

Al finalizar el capítulo se arriba a las siguientes conclusiones :

- Diseñar un submódulo que formará parte de la aplicación web SIMDECC para solucionar el problema científico de la investigación, dada la carencia de sistemas que cumplan con las funcionalidades requeridas por el CIM.
- Utilizar la metodología RUP para el diseño del submódulo Derivación de las variables del CRD para el SIMDECC.
- Utilizar PHP 5.2.0 como lenguaje de programación y Apache 2.2.3 como servidor web.
- Emplear Visual Paradigm 3.1 como herramienta CASE para el desarrollo de la Ingeniería de Software y UML 2.0 como lenguaje de modelado.
- Usar el framework Symfony 1.0.10 con los patrones que implementa.
- Utilizar Subversion 1.4.2 como herramienta para el control de versiones.

Capítulo 2: Diseño del Sistema

En el presente capítulo se realiza el diseño del submódulo “Derivación de las variables del CRD”, con el objetivo de crear un modelo físico que sirva como entrada al modelo de implementación. Con el diseño se logra un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, o sea, cómo cumple el sistema sus metas. La realización de este capítulo es de suma importancia puesto que de él depende que la futura aplicación se implemente sin ambigüedades.

2.1 Modelo de Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema.

El modelo de diseño está muy cercano al de implementación, lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. (11)

El diseño del presente trabajo de diploma se realizará a partir de los artefactos arrojados por la investigación precedente referida en el capítulo anterior. De ellos, se cuenta con 13 requisitos funcionales y 6 casos de uso del sistema, los cuales se muestran a continuación:

Requisitos funcionales:

1. Mostrar modelos de un ensayo.
2. Mostrar submodelos de un modelo.
3. Mostrar variables de un submodelo.
4. Mostrar subvariables de una variable.
5. Adicionar variables de dependencia a la validación de una subvariable.
6. Eliminar variables de dependencia de la validación de una subvariable.
7. Establecer casos de validación de una subvariable.

8. Adicionar un nuevo caso a la validación de una subvariable.
9. Eliminar casos de validación de una subvariable.
10. Mostrar las condiciones que puede tener una variable de dependencia. (Mostrar las condiciones según el tipo de la variable)
11. Adicionar las condiciones definidas para una variable de dependencia.
12. Mostrar las reglas de derivación que puede tener una subvariable. (Mostrar las reglas de derivación según el tipo de la subvariable y las variables de las que dependa)
13. Adicionar las reglas de derivación definidas para una subvariable.

Casos de uso asociados:

- Iniciar validación.
- Adicionar variables de dependencia.
- Eliminar variables de dependencia.
- Gestionar casos.
- Establecer condiciones.
- Derivar.

2.2 Patrones

Los patrones surgen de la experiencia de seres humanos al tratar de lograr ciertos objetivos y capturan la experiencia existente para promover buenas prácticas. Proponen una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones. Cada patrón describe un problema que ocurre una y otra vez, y luego describe el núcleo de la solución a ese problema, de tal manera que se puede usar esa solución un millón de veces más, sin hacer la mismo dos veces.

2.2.3 Patrones de diseño

Los patrones de diseño hacen más fácil reutilizar con éxito los diseños y arquitecturas, ayudan a elegir entre diseños alternativos, hacen a un sistema reutilizable y evitan alternativas que comprometen la reutilización.

Los patrones de diseño se clasifican de acuerdo a su propósito en:

- **Patrones de Creación:** Tratan la creación de instancias.

- **Patrones Estructurales:** Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad.
- **Patrones de Comportamiento:** Tratan la interacción y cooperación entre clases.

Según su ámbito se clasifican en:

- **De clase:** Basados en la herencia de clases.
- **De objeto:** Basados en la utilización dinámica de objetos.

2.2.3.1 Patrón Singleton

Singleton (instancia única) es un patrón de creación de objetos, *diseñado para restringir la creación de objetos pertenecientes a una clase. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.*

El patrón singleton se implementa creando en la clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula la construcción de los objetos. En muchos lenguajes esto se logra restringiendo el alcance del constructor a través de atributos como protegido o privado.

Las situaciones más habituales de aplicación de este patrón son aquellas en las que dicha clase controla el acceso a un recurso físico único (como puede ser el ratón o un archivo abierto en modo exclusivo) o cuando cierto tipo de dato debe estar disponible para todos los demás objetos de la aplicación.

El patrón singleton provee una única instancia global gracias a que:

- *La propia clase es responsable de crear la única instancia.*
- *Permite el acceso global a dicha instancia mediante un método de clase.*
- *Declara el constructor de clase como privado para que no sea instanciable directamente. (12)*

Las clases que requieran acceder a la instancia de la clase Singleton lo consiguen mediante el método de recuperación de la instancia Singleton: getInstance. Evidentemente el método getInstance debe ser estático (para acceder sin una instancia concreta) o método de clase; en lenguajes que no dispongan de esta facilidad se puede usar una función global, que hace accesible al objeto pero no previene el crear múltiples instancias de objetos.

2.2.3.2 Patrón Decorator

Decorator es un patrón estructural de objeto, usado fundamentalmente para agregar funcionalidad de adorno a un objeto individual de forma dinámica y transparente, es decir, sin afectar a otros objetos. Es una alternativa a crear demasiadas subclases por herencia. Se emplea también para adicionar responsabilidades que pueden ser retiradas o añadidas, así como en situaciones en las que no resulta práctico usar la herencia para añadir dichas responsabilidades. Es muy utilizado también para proporcionar opciones de embellecimiento en las interfaces al usuario.

Ofrece las siguientes ventajas:

- *Es más flexible que la herencia estática.*
- *Evita que las clases altas de la jerarquía estén demasiado cargadas de funcionalidad.*(13)
- Se reduce el número de clases y el árbol de herencia de clases.

2.2.3.3 Patrón Fachada

Fachada es un patrón estructural que simplifica el acceso a un conjunto de clases o interfaces. Tiene como objetivo reducir la dependencia entre clases, ofreciendo un punto de acceso al resto de las clases; si estas cambian o se sustituyen por otras solo hay que actualizar la clase Facade sin que el cambio afecte a las aplicaciones cliente. Facade no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas, en los casos en que se requiere se puede acceder directamente a ellas. Oculta a los clientes parte de la complejidad de los subsistemas y disminuye el acoplamiento entre subsistemas y cliente.

2.2.3.4 Patrones GRASP

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender.

Los patrones GRASP son patrones de diseño que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades).

Los patrones GRASP son:

- **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 debemos conectar con el objeto producido en cualquier evento.
- **Experto:** Asignar una responsabilidad a un experto en información (la clase que cuenta con la información necesaria para cumplir la responsabilidad).
- **Controlador:** Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.
- **Alta cohesión:** Asignar una responsabilidad de modo que la cohesión siga siendo alta de manera que tenga responsabilidades moderadas en un área funcional y colabore con las otras para llevar a cabo las tareas.
- **Bajo acoplamiento:** Asignar una responsabilidad para mantener bajo acoplamiento, o sea permite asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

2.2.3.5 Más patrones para asignar responsabilidades

- **Polimorfismo:** *Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán mediante operaciones polimórficas a los tipos en el que el comportamiento presenta variantes.*
 - **Indirección:** *Se asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, y estos no terminen directamente acoplados. El intermediario crea una Indirección entre el resto de los componentes o servicios.*
 - **Fabricación Pura:** *Asignar un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema: una cosa inventada para dar soporte a una alta cohesión, un bajo acoplamiento y reutilización.*
 - **No hables con Extraños:** *Se asigna la responsabilidad a un objeto directo del cliente para que colabore con un objeto indirecto, de modo que el cliente no necesite saber nada del objeto indirecto.*
- (14)

Al hacer uso de un framework, como Symfony, que utiliza implícitamente una serie de patrones de diseño, la investigación se basa de igual modo en su empleo, ayudando a mejorar la calidad del diseño y la futura implementación. Patrones como los GRASP y el Decorator son ejemplos típicos, el

segundo de ellos usado fundamentalmente en la capa de la Vista que propone Symfony. Se cuenta también con el empleo de otros patrones como Fachada y Singleton, utilizados en el Modelo.

2.2.4 Patrón de arquitectura

2.2.4.1 Patrón Modelo-Vista- Controlador

Los patrones expresan el esquema fundamental de la organización para sistemas de software, de ahí la importancia de utilizarlos para lograr un mejor entendimiento de la estructura de la aplicación.

Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. MVC se ve frecuentemente en aplicaciones web, donde la Vista es la página HTML y el código que provee de datos dinámicos a la página.

MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- *Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.*
- *Vista (View): Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.*
- *Controlador (Controller): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("service requests") para el modelo o la vista. (15)*

En la investigación se hará uso del MVC por las facilidades que brinda y además por ser el patrón de arquitectura que utiliza o trae definido el framework Symfony, herramienta que se utilizará para el desarrollo de la aplicación, lo cual ha sido expresado con anterioridad.



Figura 11 Patrón Modelo-Vista-Controlador

2.2.5 Uso de los patrones en la investigación

Los patrones de diseño son una guía para resolver problemas comunes en programación. Al aprenderlos, se aprende a identificar de mejor manera las dificultades que se pueden presentar antes de que aparezcan, y una vez identificadas, ayudan a decidir cuál es la solución que mejor se aplica al problema dado.

En la presente investigación se utilizan algunos patrones de diseño con vista a lograr un mejor entendimiento del diseño y una adecuada asignación de responsabilidades en cada clase. Un papel importante en esta función lo juega el patrón Singleton en la capa del Modelo que propone el framework de desarrollo, el cual mediante una instancia provee un punto de acceso global a la clase donde se encuentra el método `getInstance()`. En el caso de la investigación se tienen 6 de estas clases, las cuales son: `FachadaIniciarValidacion`, `FachadaAddVarDep`, `FachadaElimVarDep`, `FachadaEstabCondiciones`, `FachadaGestionarCasos`, `FachadaDerivar`, cada una correspondiente a un caso de uso del sistema. De este modo se garantiza tener una única instancia de ellas, restringiendo la creación de objetos. A su vez estas clases representan un intermediario entre las clases del Controlador y las restantes del Modelo, poniéndose de manifiesto el uso del patrón Indirección y Facade, al permitir además que se reduzca el número de clases con las cuales las clases de tipo Action del controlador tendrían que interactuar, logrando un acoplamiento más débil entre ellas. En la capa de la Vista el Layout decora cada una de las plantillas, siendo este comportamiento una implementación del patrón Decorator.

Patrones como Bajo acoplamiento y Alta cohesión son empleados al distribuir y organizar los métodos de las clases del diseño, aunque estos no son los únicos, pues se utilizan además el Creador, Controlador, entre otros, empleados indistintamente para la correcta asignación de responsabilidades siempre que se desee obtener un buen diseño.

El patrón de arquitectura se pone de manifiesto en todo momento puesto que la arquitectura propuesta para el desarrollo del sistema requiere de su uso asociado al framework Symfony.

2.3 Realización de los casos de uso del diseño

2.3.1 Vista Lógica

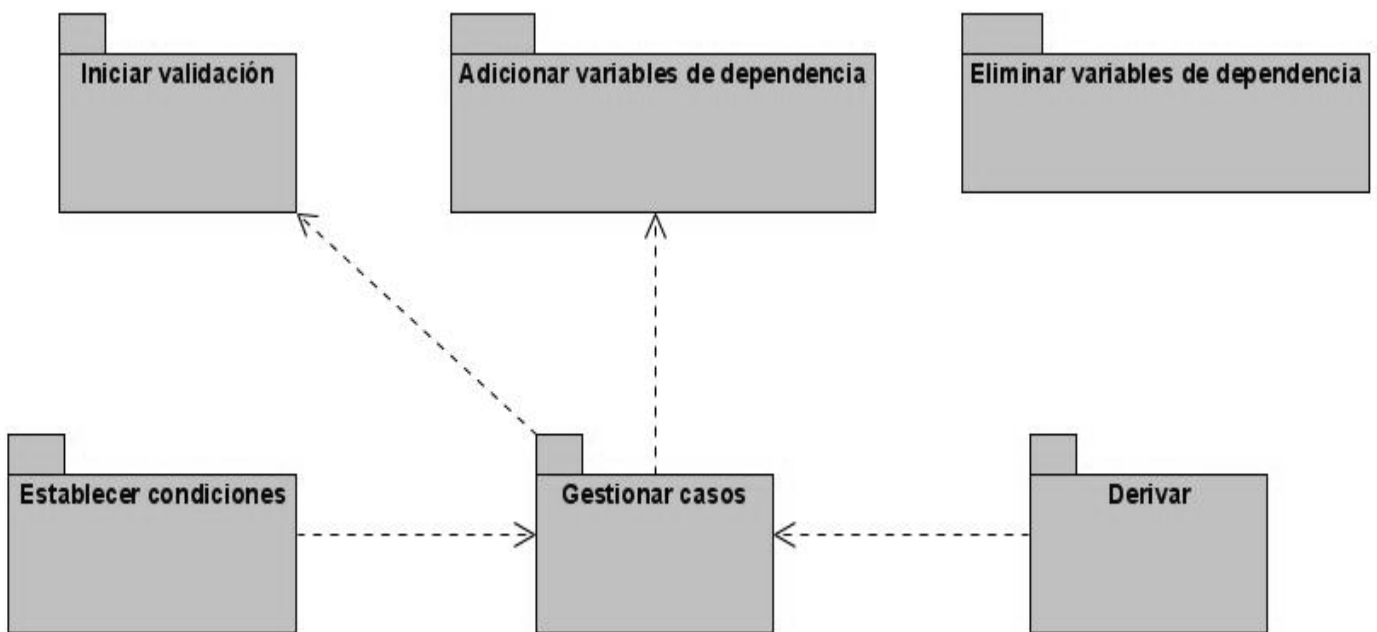


Figura 12 Diagrama de paquetes del diseño

2.3.2 Descripción de las clases del diseño

La descripción de las clases del diseño permite una mejor comprensión de cada una de ellas y detallan sus responsabilidades y atributos, logrando que el diagrama de clases del diseño muestre la interacción entre ellas y se logre un mayor entendimiento del mismo.

Tabla 2.1 Descripción de la clase del Diseño: `vallniciarValidacionActions`

Nombre: <code>vallniciarValidacionActions</code>
Tipo: Clase (Es la clase que contiene todas las funciones correspondientes al paquete del diseño Iniciar validación. Llama a la clase <code>FachadaIniciarVal</code> con el objetivo de utilizar las funciones relacionadas con el Modelo)

Atributo		Tipo
Responsabilidades		
Nombre:	Descripción:	
executeMostrarModelos()	Muestra todos los modelos correspondientes a un Ensayo Clínico. Llama a la función buscarModelos() de la clase FachadaIniciarVal.	
executeMostrarSubmodelos()	Muestra todos los submodelos correspondientes a un modelo seleccionado. Llama a la función buscarSubmodelos() de la clase FachadaIniciarVal.	
executeMostrarVariables()	Muestra todas las variables y subvariables correspondientes a un submodelo específico. Llama a la función mostrarVariables() de la clase FachadaIniciarVal.	

Tabla 2.2 Descripción de la clase del Diseño: valAddVarDepActions

Nombre: valAddVarDepActions		
Tipo: Clase (Es la clase que contiene todas las funciones correspondientes al paquete del diseño Adicionar variables de dependencia. Llama a la clase FachadaAddVarDep con el objetivo de utilizar las funciones relacionadas con el Modelo)		
Atributo		Tipo
Responsabilidades		
Nombre:	Descripción:	
executeVarDep()	Función que muestra la página que permite al usuario adicionar variables de dependencia (en este caso son subvariables, pero se utiliza el término de variables de dependencia para mejor entendimiento) a la subvariable seleccionada.	
executeAddCantVarDep()	Guarda el valor entrado por el usuario usando la función getUser(). Llama a la función executeListar Modelos ().	
hasErrorAddCantVarDep()	Muestra un mensaje de error si el valor entrado no es correcto.	
executeListarModelos()	Llama a la función buscarModelos() de la clase FachadaAddVarDep, la cual devuelve una lista de modelos. Redirecciona al método executeVarDep() para construir la página y mediante una función AJAX muestra un formulario para que el usuario seleccione las variables de las que depende la subvariable a derivar, mostrando todos esos modelos.	
executeSeleccionarModelo()	Función que guarda el modelo seleccionado. Llama a la función executeListarSubmodelo().	
executeListarSubmodelos()	Función que llama al método buscarSubmodelos() de la clase FachadaAddVarDep, la cual devuelve una lista de submodelos. Redirecciona al método executeVarDep() para construir la página y mediante una función AJAX muestra un formulario con todos los submodelos del modelo seleccionado.	
executeSeleccionarSubmodelo()	Función que guarda el submodelo seleccionado. Llama a la función executeListarVariables().	
executeListarVariables()	Función que llama al método buscarVariables() de la clase	

	FachadaAddVarDep, la cual devuelve una lista de variables. Redirecciona al método executeVarDep() para construir la página y mediante una función AJAX muestra un formulario con todas las variables del submodelo seleccionado.
executeSeleccionarVariable()	Función que guarda la variable seleccionada. Llama a la función ListarSubvariable().
executeListarSubvariable()	Función que llama al método buscarSubvariable() de la clase FachadaAddVarDep, la cual devuelve una lista de subvariables. Redirecciona al método executeVarDep() para construir la página y mediante una función AJAX muestra un formulario con todas las subvariables de la variable seleccionada.
executeSeleccionarSubvariable()	Función que guarda la subvariable de dependencia.
executeAdicionarVariable()	Guarda la variable (subvariable) de dependencia con el modelo, submodelo y variable a la que pertenece. Si la cantidad de variables de dependencia entrada con anterioridad es mayor que uno, da la posibilidad de volver a escoger cada uno de los parámetros para agregar la otra variable de dependencia.

Tabla 2.3 Descripción de la clase del Diseño: valEliminarVarDepActions

Nombre: valEliminarVarDepActions	
Tipo: Clase (Es la clase que contiene todas las funciones correspondientes al paquete del diseño Iniciar validación. Llama a la clase FachadaElimVarDep con el objetivo de utilizar las funciones relacionadas con el Modelo)	
Atributo	Tipo
Responsabilidades	
Nombre:	Descripción:
executeElimVarDep()	Llama a la función eliminarVar() de la clase FachadaElimVarDep que elimina la subvariable de dependencia seleccionada.

Tabla 2.4 Descripción de la clase del Diseño: valGestionarCasosActions

Nombre: valGestionarCasosActions	
Tipo: Clase (Es la clase que contiene todas las funciones correspondientes al paquete del diseño Gestionar casos. Llama a la clase FachadaGestionarCasos con el objetivo de utilizar las funciones relacionadas con el Modelo)	
Atributo	Tipo
Responsabilidades	
Nombre:	Descripción:
executeCantVarDep()	Verifica la cantidad de variables de dependencia que tiene la

	subvariable a derivar. Llama a la función buscarCantVarDep() de la clase FachadaGestionarCasos, la cual verifica si la cantidad de subvariables de dependencia es mayor o igual a cero. Llama al método executeCasos() para que se construya la página de los casos.
executeCasos()	Construye la página en dependencia del resultado arrojado por el método executeCantVarDep(), o sea, da la opción de adicionar la cantidad de casos que tiene la subvariable a derivar si la cantidad de subvariables de dependencia es mayor que cero, sino, muestra la tabla de estado con las opciones de adicionar condiciones, derivar y validar mediante una función AJAX, además de brindar las opciones de adicionar un nuevo caso y eliminar casos
executeCantCasos()	Guarda la cantidad de casos entradas por el usuario usando la función getUser() y mediante una función AJAX muestra una tabla de estado con la misma cantidad de filas, además de brindar las opciones de adicionar un nuevo caso y eliminar casos. Además, en las columnas de la tabla muestra las opciones para definir las condiciones de dependencia, derivar y validar. Llama al método executeCasos() para que se construya la página de los casos.
hasErrorCantCasos()	Muestra un mensaje de error si el valor entrado no es correcto.
executeNuevoCaso()	Adiciona una nueva fila en la tabla de estado y además llama a la función addnuevoCaso() de la clase FachadaGestionarCasos, la cual adiciona un nuevo caso.
executeEliminarCaso()	Llama a la función eliminarCaso() de la clase FachadaGestionarCasos, la cual elimina los casos seleccionados por el usuario y a su vez las filas correspondientes a ellos.
executeBuscarVarDep()	Llama a la función buscarVariablesDep() de la clase FachadaGestionarCasos, la cual devuelve todas las subvariables de dependencia asociadas a la variable que se está derivando. Llama a la función buscarModelos() de la clase FachadaIniciarValidacion y finalmente redirecciona a la función executeVarDep() del módulo AdicionarVariablesDependencia.

Tabla 2.5 Descripción de la clase del Diseño: valDerivarActions

Nombre: valDerivarActions	
Tipo: Clase (Es la clase que contiene todas las funciones correspondientes al paquete del diseño Derivar. Llama a la clase FachadaDerivar con el objetivo de utilizar las funciones relacionadas con el Modelo)	
Atributo	Tipo
Responsabilidades	
Nombre:	Descripción:
executeReglasDerivacion()	Muestra las reglas de derivación en correspondencia con el tipo de dato de la subvariable a derivar, y la cantidad de subvariables de dependencia asociadas a ella. Para ello, llama a la función

	<p>buscarVDep() de la clase FachadaDerivar devolviendo el listado de subvariables de dependencia. Luego verifica el tipo de dato de la subvariable a derivar llamando a la función verTipoDato() de la clase FachadaDerivar. Si el tipo de dato es nomenclador, llama a la función cargarNomencladores() de la clase FachadaDerivar, la cual carga los valores posibles a tomar por esa subvariable de tipo nomenclador.</p>
executeGuardarRegla()	<p>Llama a la función guardaRegla() de la clase FachadaDerivar para guarda la regla de derivación seleccionada por el usuario. Llama a la función executeCargaReglaDerivacion()</p>
executeCargaReglaDerivacion()	<p>Llama a la función cargarReglaDerivación() de la clase FachadaDerivar para cargar la regla de derivación seleccionada por el usuario para el caso en cuestión y mostrarla en la tabla de estado en la página de los casos, para lo cual redirecciona a la función executeCasos() del módulo GestionarCasos.</p>

Tabla 2.6 Descripción de la clase del Diseño: valEstabCondicionesActions

Nombre: valEstabCondicionesActions	
Tipo: Clase (Es la clase que contiene todas las funciones correspondientes al paquete del diseño Establecer condiciones. Llama a la clase FachadaEstabCondiciones con el objetivo de utilizar las funciones relacionadas con el Modelo)	
Atributo	Tipo
Responsabilidades	
Nombre:	Descripción:
executeCondiciones()	<p>Llama a la función buscarVarDepTipo() de la clase FachadaEstabCondiciones para buscar las variables de dependencia de la subvariable a derivar y el tipo de dato de cada una de ellas. Muestra las condiciones de dependencia posibles para las subvariables de acuerdo al tipo de dato de las mismas.</p>
executeGuardarCondiciones()	<p>Llama a la función guardarCondiciones() de la clase FachadaEstabCondiciones para guarda las condiciones de dependencia seleccionadas por el usuario. Llama además a la función executeCargarCondiciones().</p>
executeCargarCondiciones()	<p>Llama a la función cargarCondiciones() de la clase FachadaEstabCondiciones para cargar las condiciones de dependencia seleccionadas por el usuario para el caso en cuestión y mostrarlas en la tabla de estado en la página de los casos, para lo cual redirecciona a la función executeCasos() del módulo GestionarCasos. Si la regla seleccionada es "asignada" se habilita la opción de "Validar" en la tabla de estado.</p>

Tabla 2.7 Descripción de la clase del Diseño: NomencladorComponents

Nombre: NomencladorComponents	
Tipo: Clase perteneciente al paquete de diseño Establecer condiciones. Llama a la clase FachadaEstabCondiciones con el objetivo de utilizar las funciones relacionadas con el Modelo) Representa la lógica del partial _Nomenclador, representado en el diagrama de clases correspondiente a este paquete.	
Atributo	Tipo
Responsabilidades	
Nombre:	Descripción:
executeListarNomencladores()	Función que busca los posibles valores que pueden tomar las subvariables de tipo Nomenclador. Llama a la función listarNomencladores() de la clase FachadaEstabCondiciones, la cual carga los valores posibles a tomar por la subvariable de tipo nomenclador.

Como se aprecia en algunas de las descripciones de las clases del diseño, se emplea el término AJAX al detallar la funcionalidad de varios métodos de clase. Por tanto, vale destacar que AJAX, agrupa varias tecnologías usadas continuamente en la actualidad para el desarrollo de aplicaciones web.

Se utiliza para *cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar la página solo re-renderizando la página y mostrando u ocultando porciones de la misma.*(16)

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano. (17)

Un ejemplo de la necesidad de utilizar funciones AJAX en la investigación, se muestra en el caso de uso Adicionar variables de dependencia, donde se necesita para ello seleccionar un modelo, y a partir de este deben mostrarse en la misma página los submodelos, luego se selecciona un submodelo, y deben aparecer las variables correspondientes al mismo, y por último al seleccionar una variable, deben mostrarse las subvariables para que finalmente se pueda seleccionar la variable de dependencia sin haber cambiado en ninguno de los casos antes mencionado la dirección URL.

Otro ejemplo es para mostrar la tabla de estado del caso de uso Gestionar caso, donde se deben mostrar tantas columnas como número casos se haya adicionado anteriormente y que una vez establecidas las condiciones de dependencia y se haya derivado debe actualizarse esta tabla mostrándose dichos resultados.

2.3.3 Diagramas de Clases del Diseño

2.3.3.1 Estructura del diagrama de clases del diseño con la utilización de Symfony

Para la confección del diagrama de clases del diseño se realizó un estudio de la estructura y las características del framework Symfony, el cual está basado en el patrón arquitectónico MVC, por tanto, en las figuras que se mostrarán a continuación las clases están organizadas en las tres capas siguientes:

Modelo: En esta capa están representadas las clases que tienen la lógica del negocio y que se encargan del acceso a los datos, como son las clases objeto, las clases Bases y las clases Peer. Además se tiene una clase Fachada que sirve de intermediaria entre la capa Controladora y la del Modelo, además de controlar y simplificar de esta forma el acceso a los datos.

Vista: La vista en Symfony está compuesta por varias partes:

1. Plantilla: Constituyen la representación de los datos de la acción que se está ejecutando.
2. Layout: Contiene el código HTML que es común a todas las páginas. El mismo puede modificarse para definir zonas en las que se insertan componentes externos y elementos parciales.

Elementos parciales: Estos elementos no son más que trozos de código de plantilla que se pueden reutilizar en dependencia de lo que se necesite mostrar. Está dividido en dos partes: una de presentación y la otra parte es la lógica, para la cual se utilizan los llamados componentes.

Componentes: Son como acciones, pero mucho más rápidos y a diferencia de estas no pueden manejar la seguridad ni la validación. Los elementos parciales que se utilizan como presentación de un componente tienen los mismos nombres que estos. La lógica de los componentes se guarda en una clase que hereda de sfComponents.

Controlador: El controlador en Symfony es el encargado de vincular la lógica y la presentación, dividido en varios componentes utilizados para disímiles propósitos. Los dos principales, que se representan en los diagramas de clases del diseño son los siguientes:

1. Controlador frontal: Es el único punto de entrada de la aplicación, carga la configuración y decide la acción que se debe ejecutar. Todas las peticiones web son manejadas por un solo controlador frontal.
2. Acciones: Estas contienen la lógica de la aplicación y preparan los datos que son necesarios para la capa de presentación o sea utilizan el modelo, y definen las variables que son fundamentales para la vista.

Diseño del Sistema

Luego de esta explicación se presentan seis diagramas de clases del diseño correspondientes a cada uno de los casos de uso mencionados con anterioridad en el capítulo.

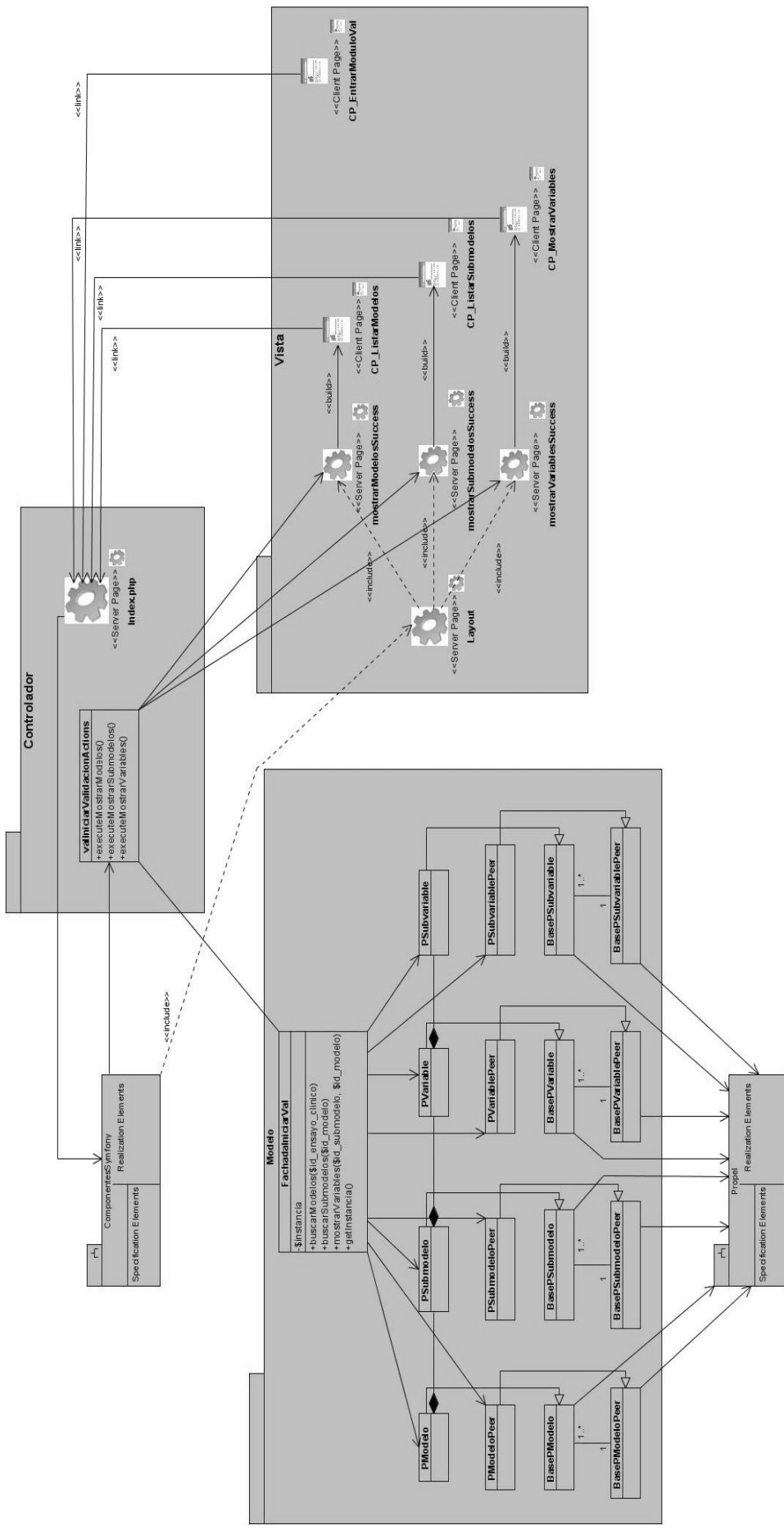


Figura 13 Iniciar validación

Diseño del Sistema

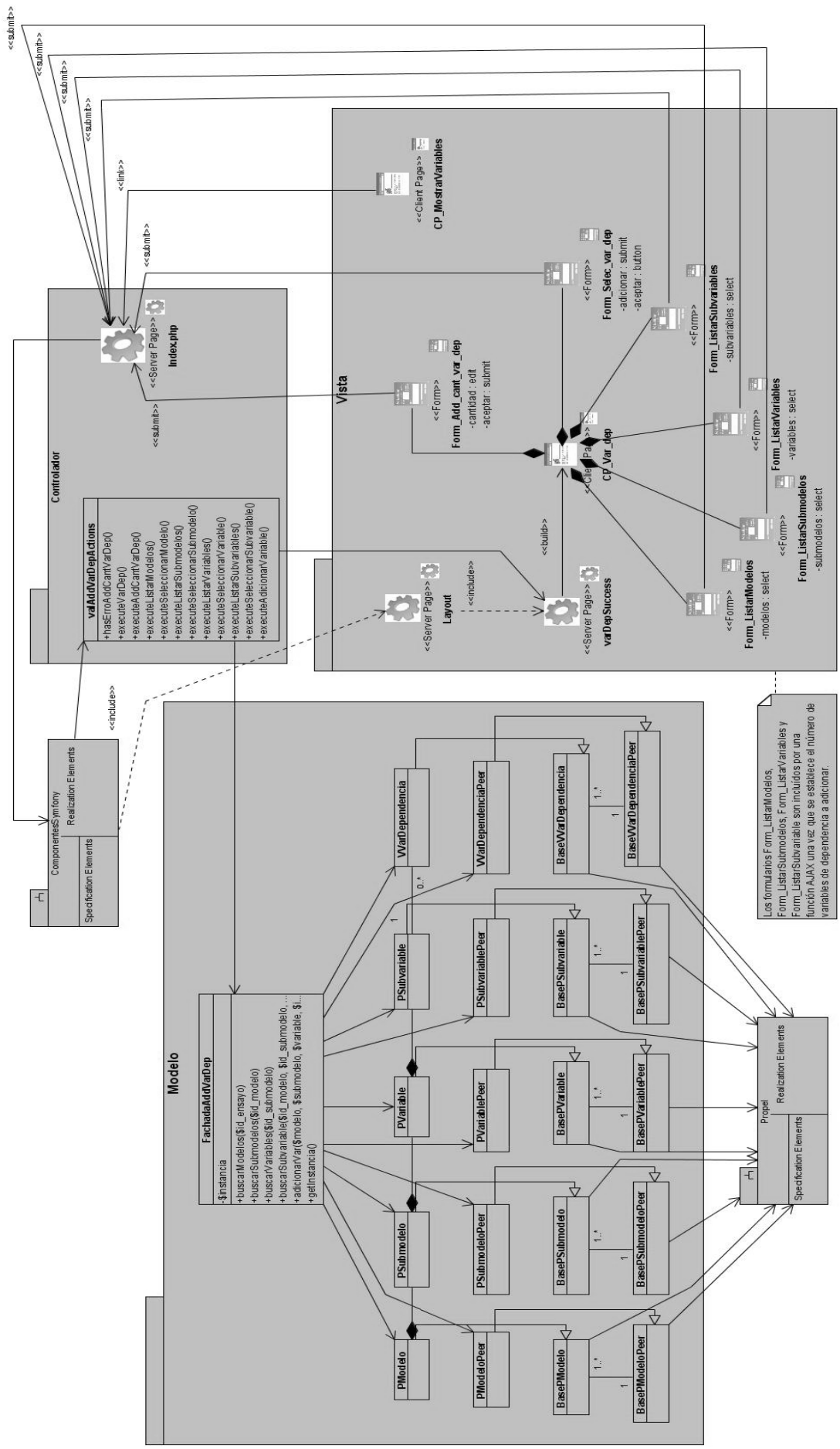


Figura 14 Adicional variables de dependencia

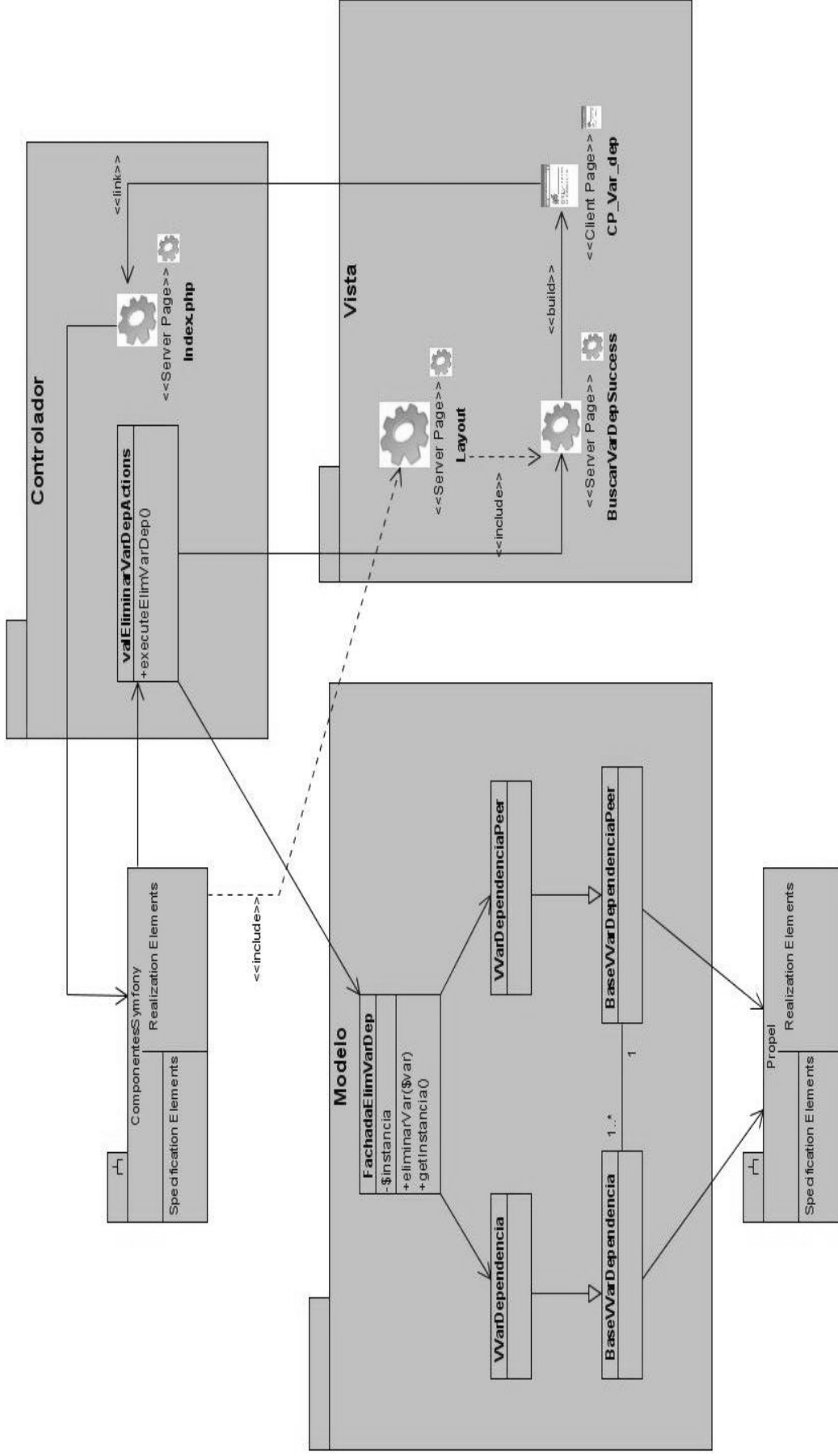


Figura 15 Eliminar variables de dependencia

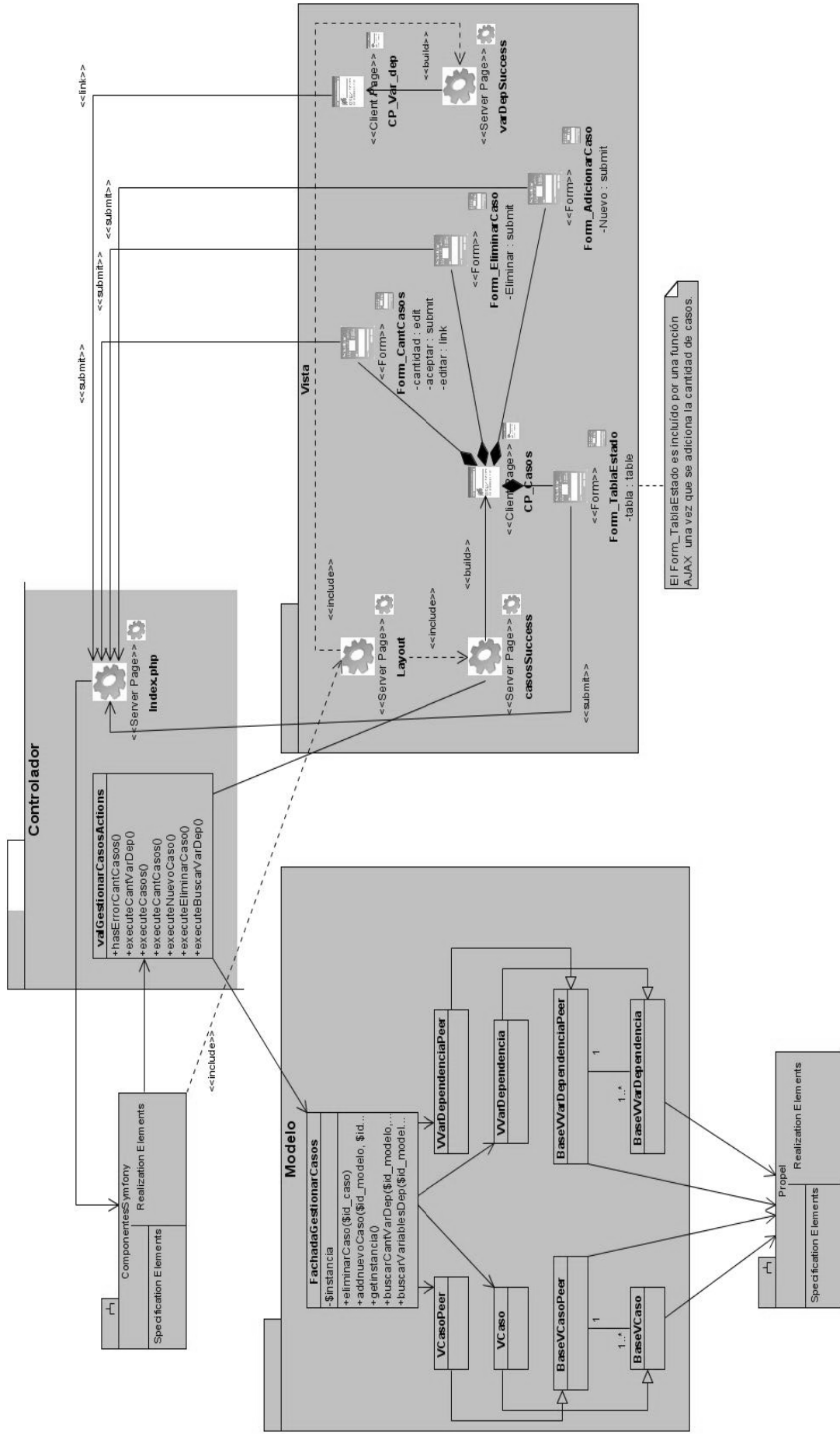


Figura 16 Gestionar casos

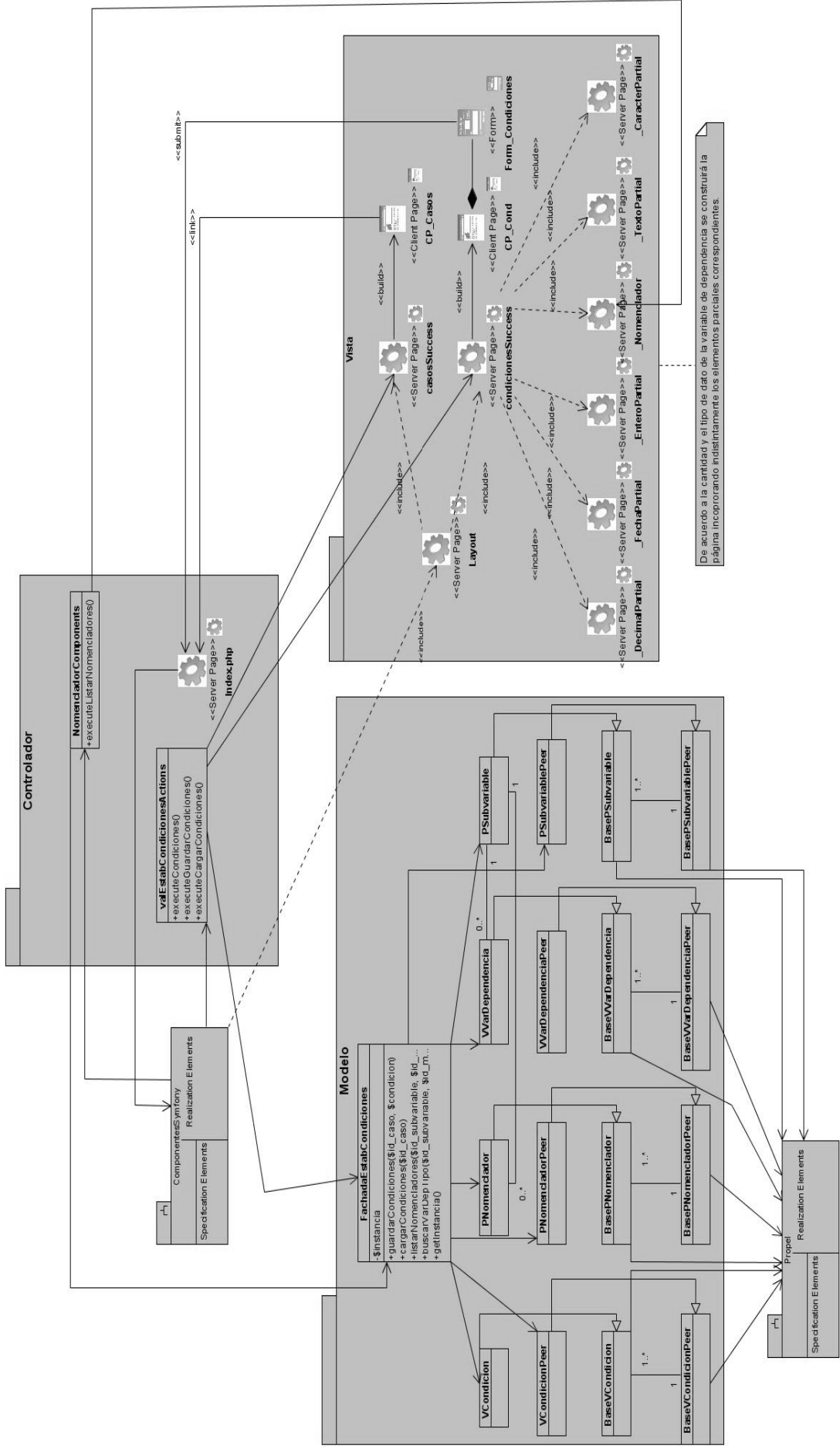


Figura 17 Establecer condiciones

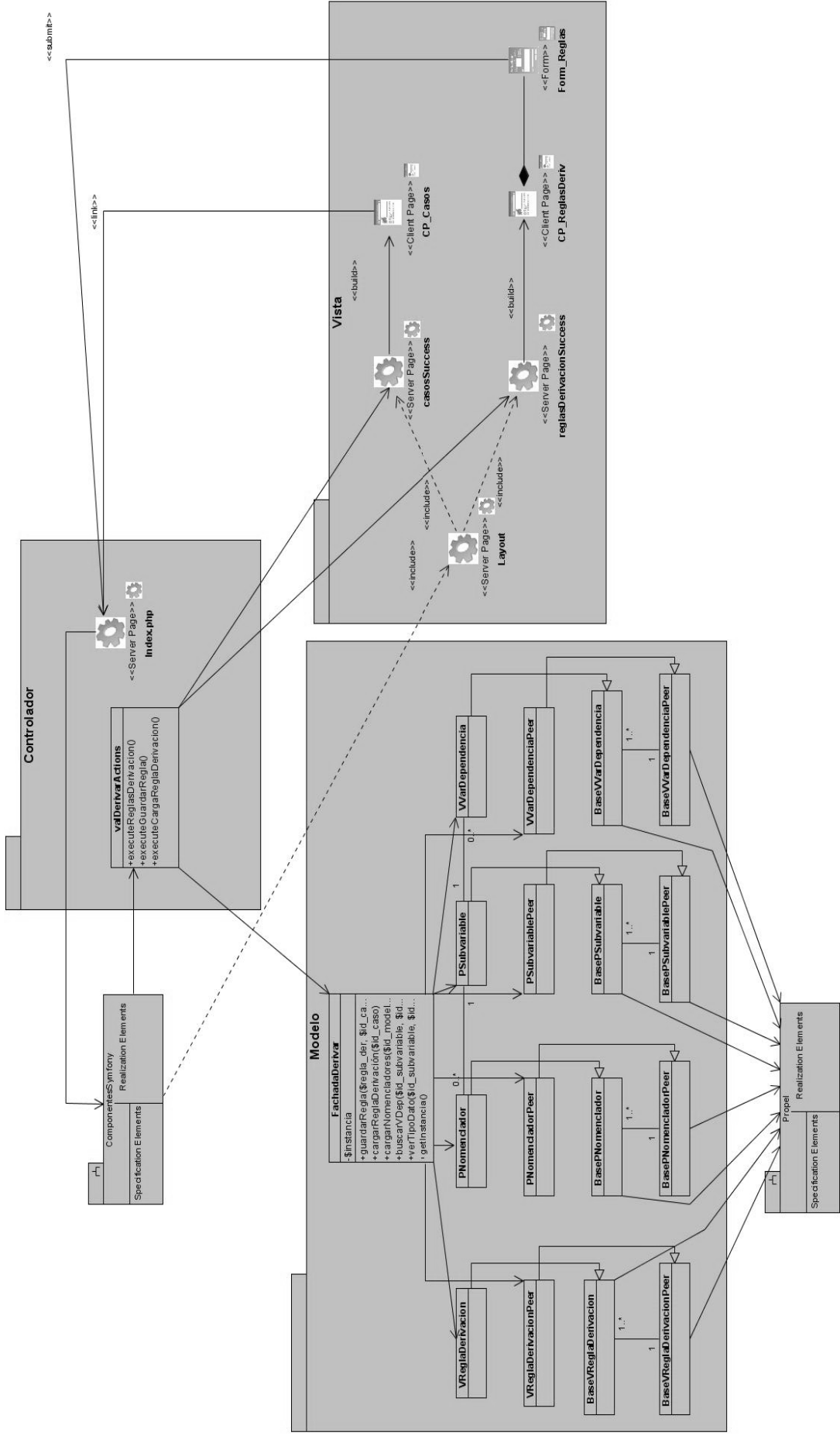


Figura 18 Derivar

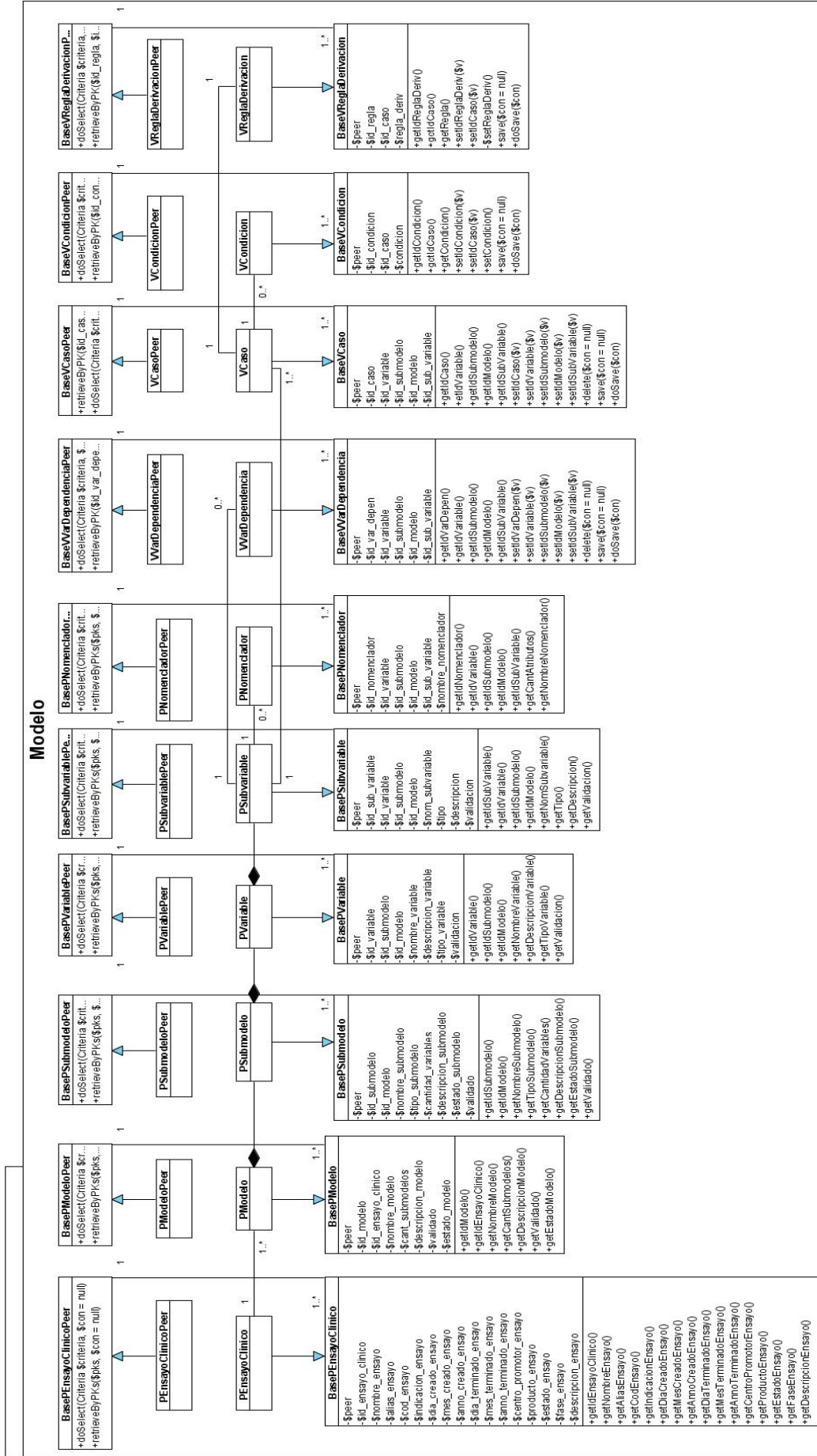


Figura 19 Capa del Modelo

2.3.4 Diagramas de Interacción

Los diagramas de secuencia muestran una interacción ordenada según la secuencia temporal de eventos. En particular, muestran los objetos participantes en la interacción y los mensajes (llamadas a métodos) que intercambian, ordenados según su secuencia en el tiempo.

Caso de uso: Iniciar validación

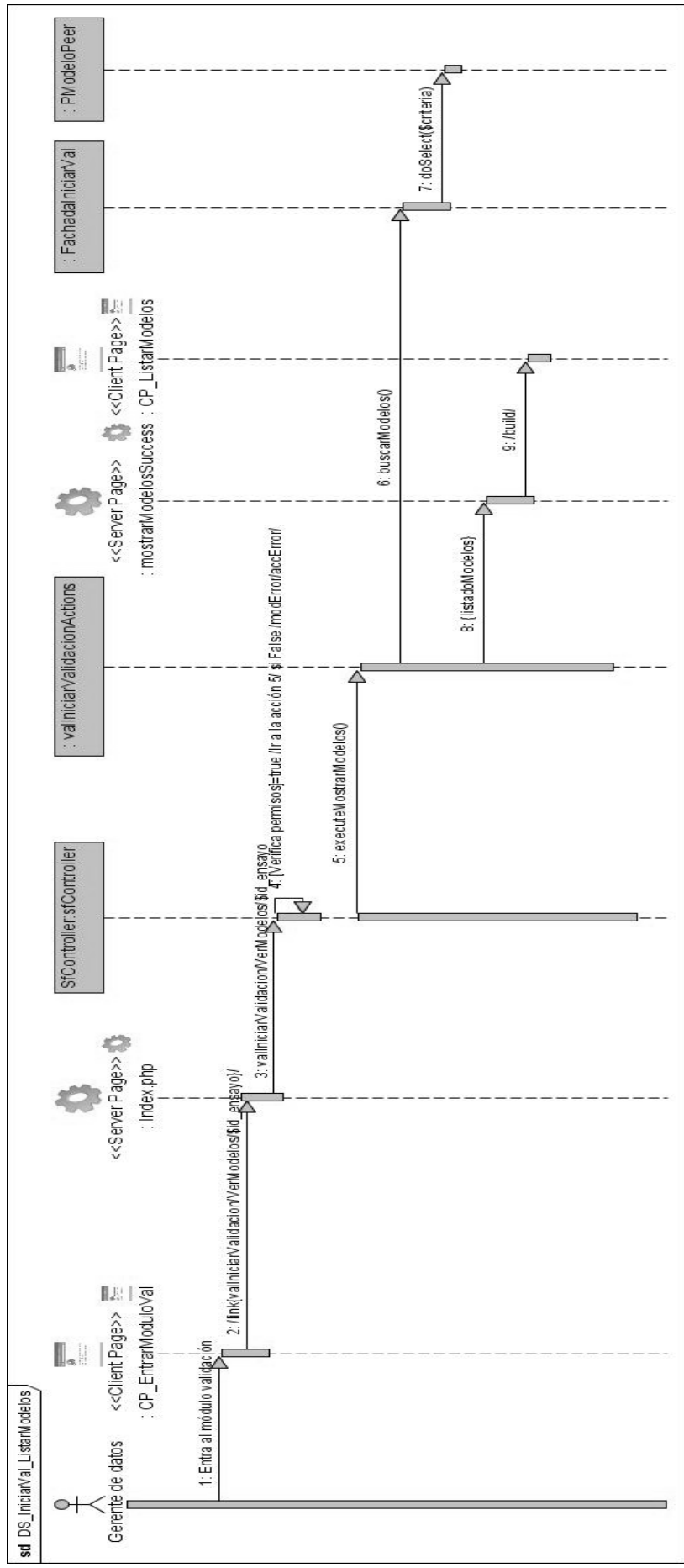


Figura 20 Listar modelos

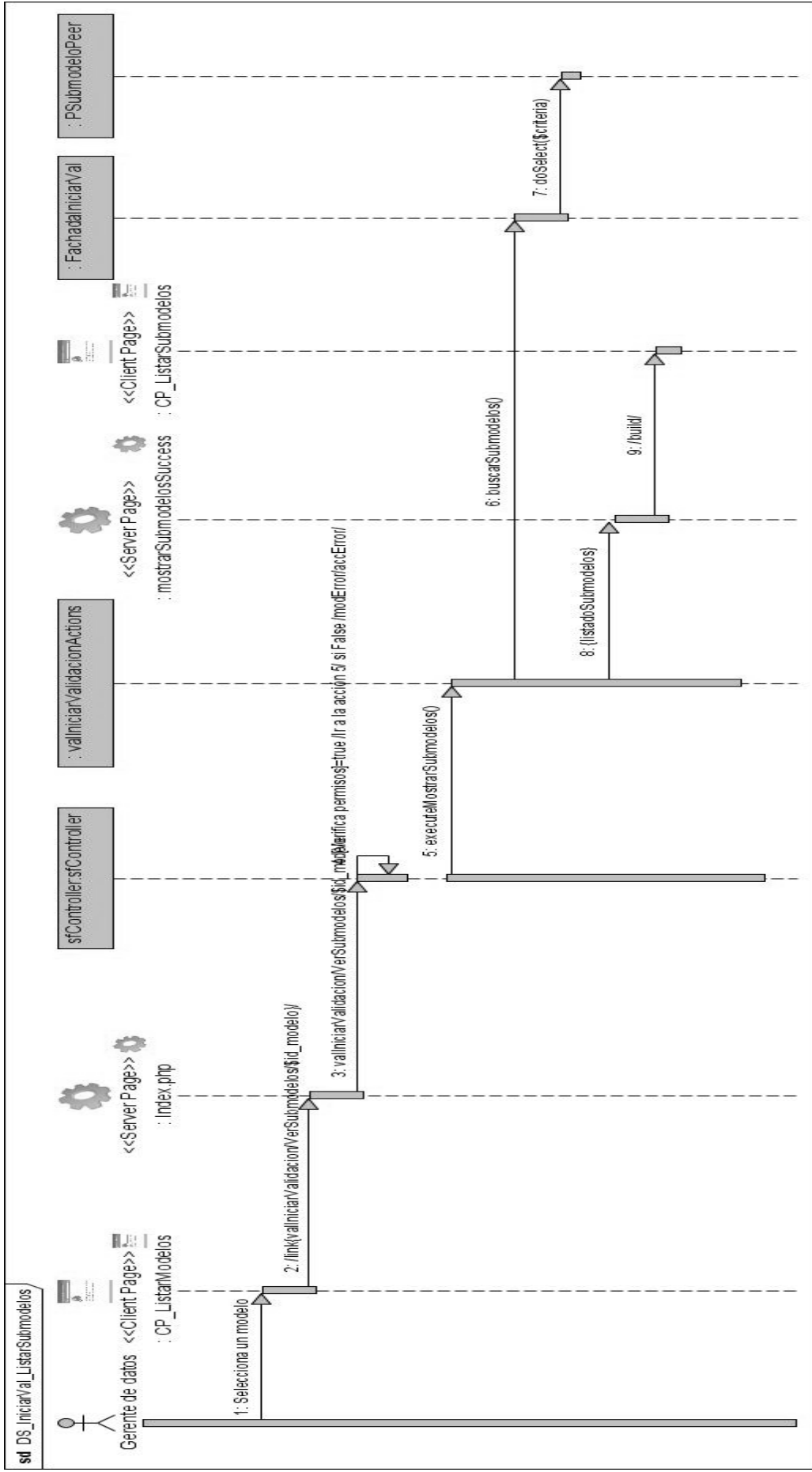


Figura 21 Listar submodelos

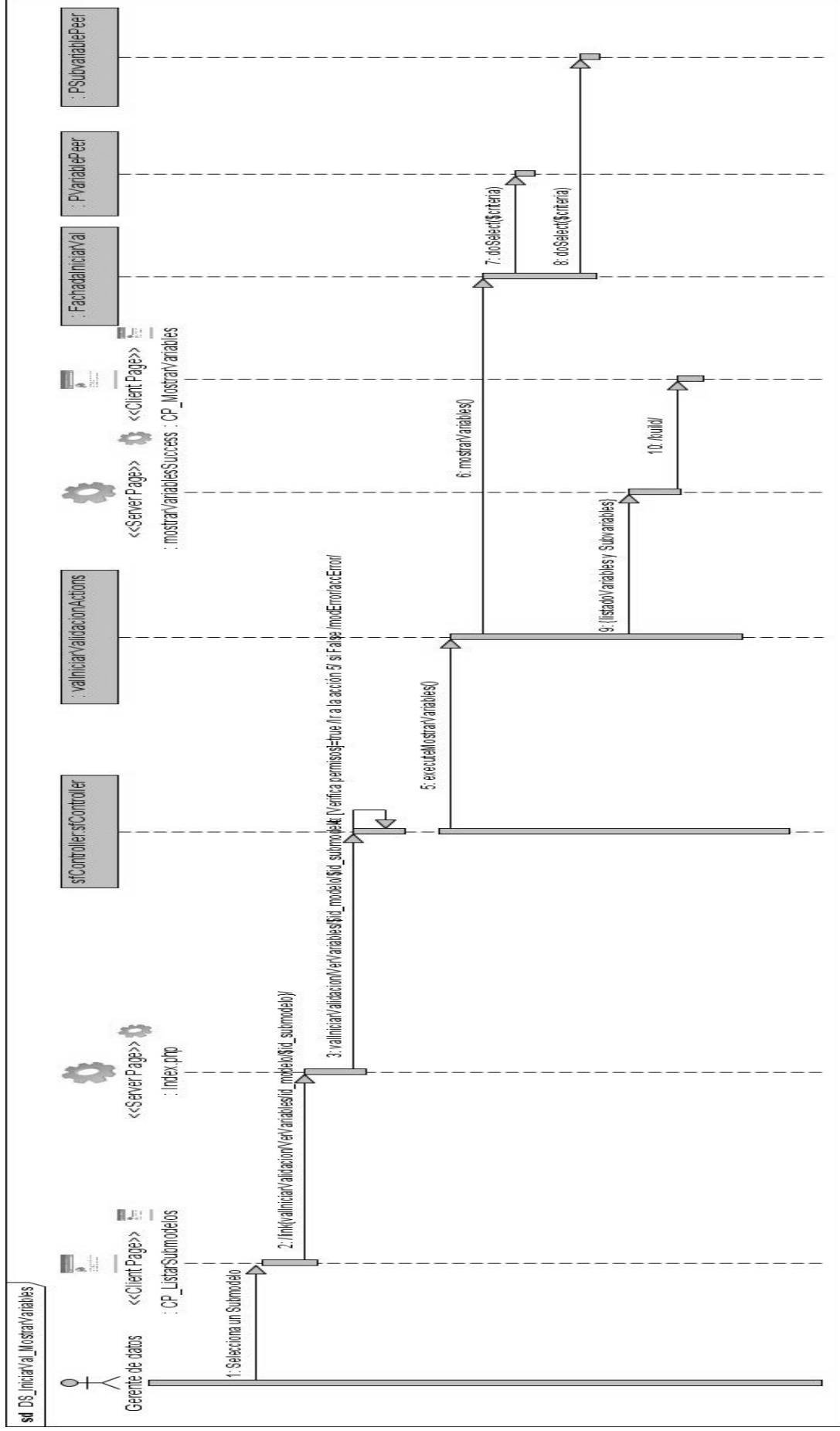


Figura 22 Mostrar variables

Caso de uso: Adicionar variables de dependencia

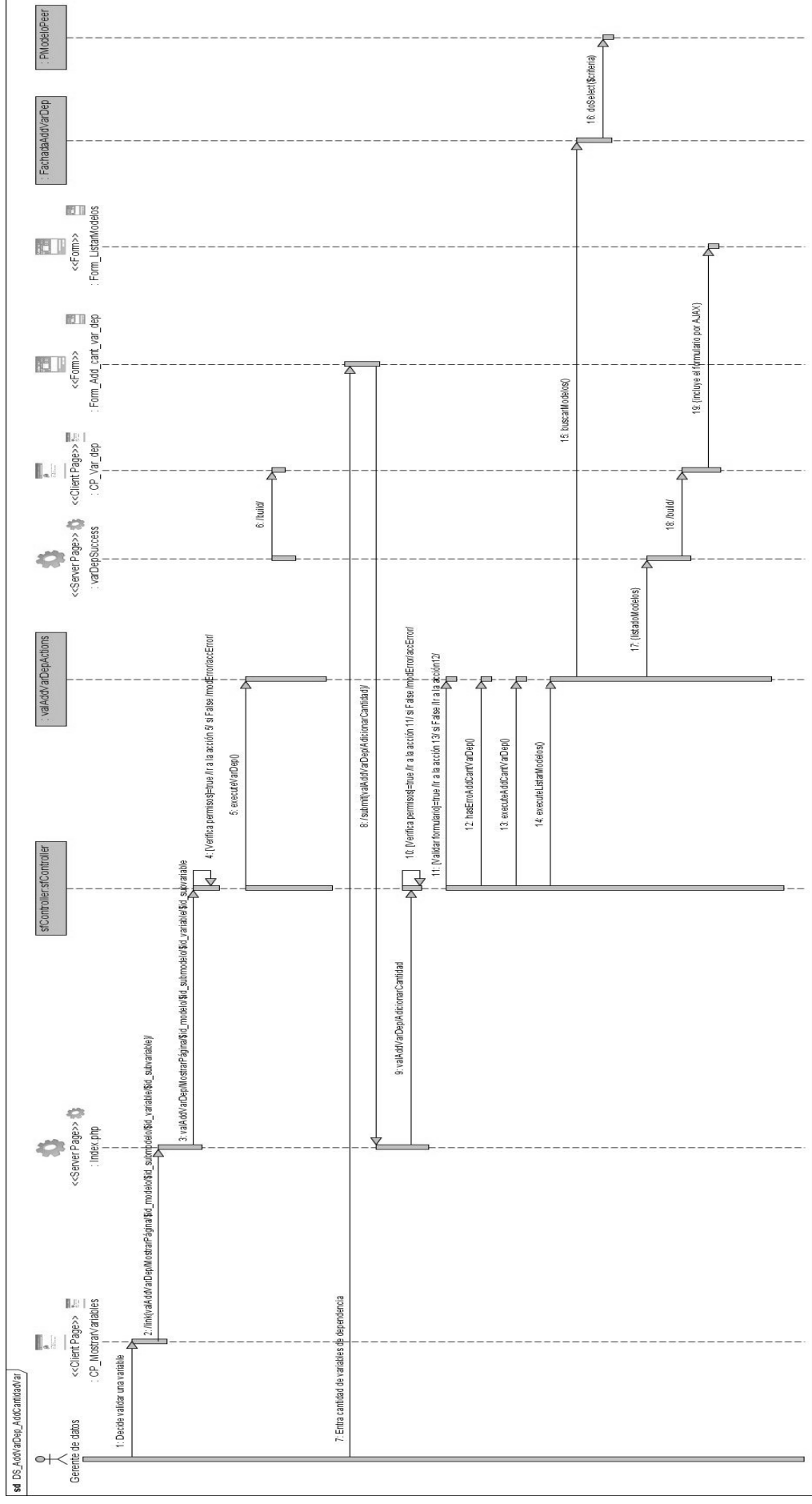


Figura 23 Adicionar cantidad de variables de dependencia

Caso de uso: Eliminar variables de dependencia

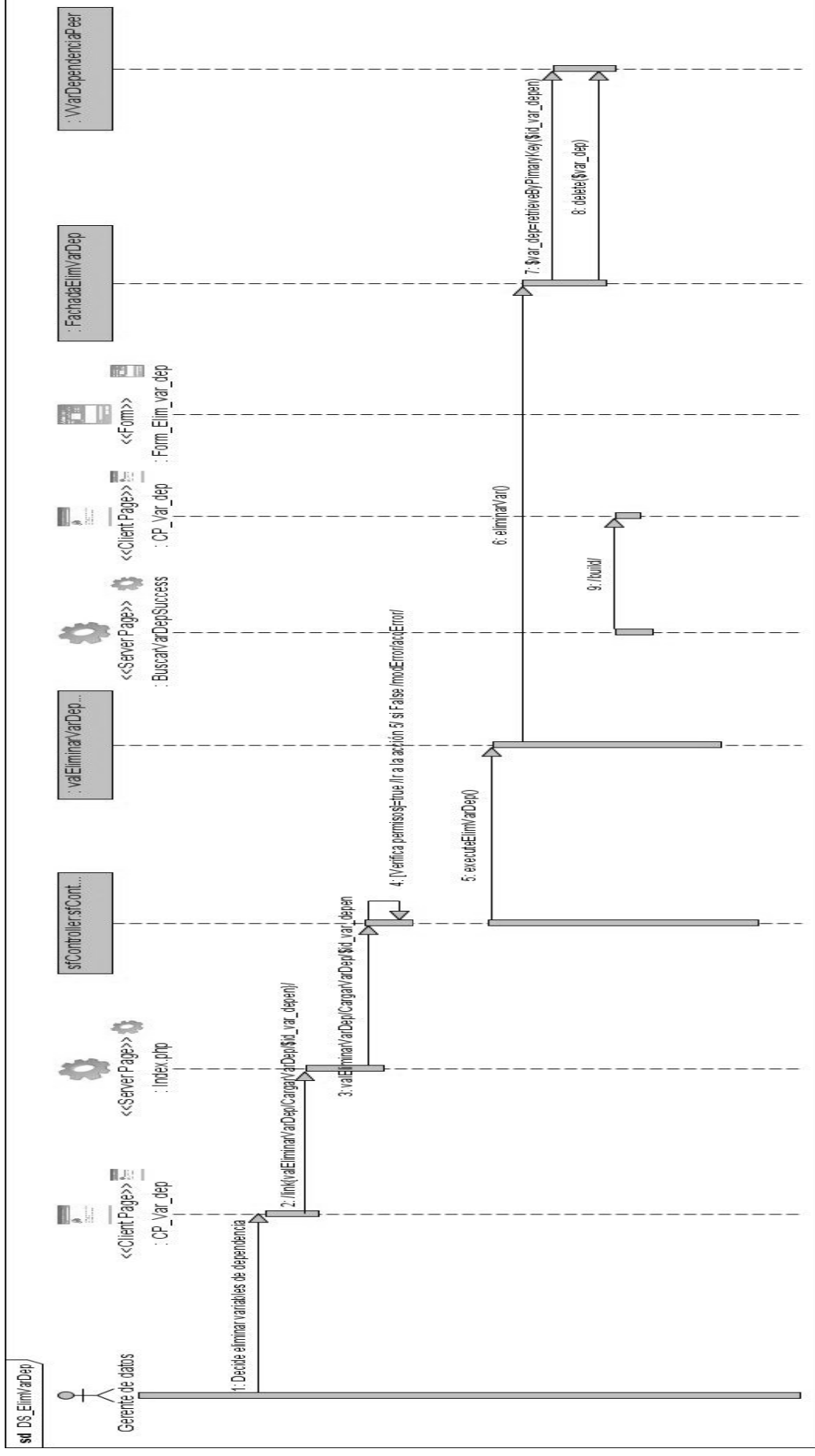
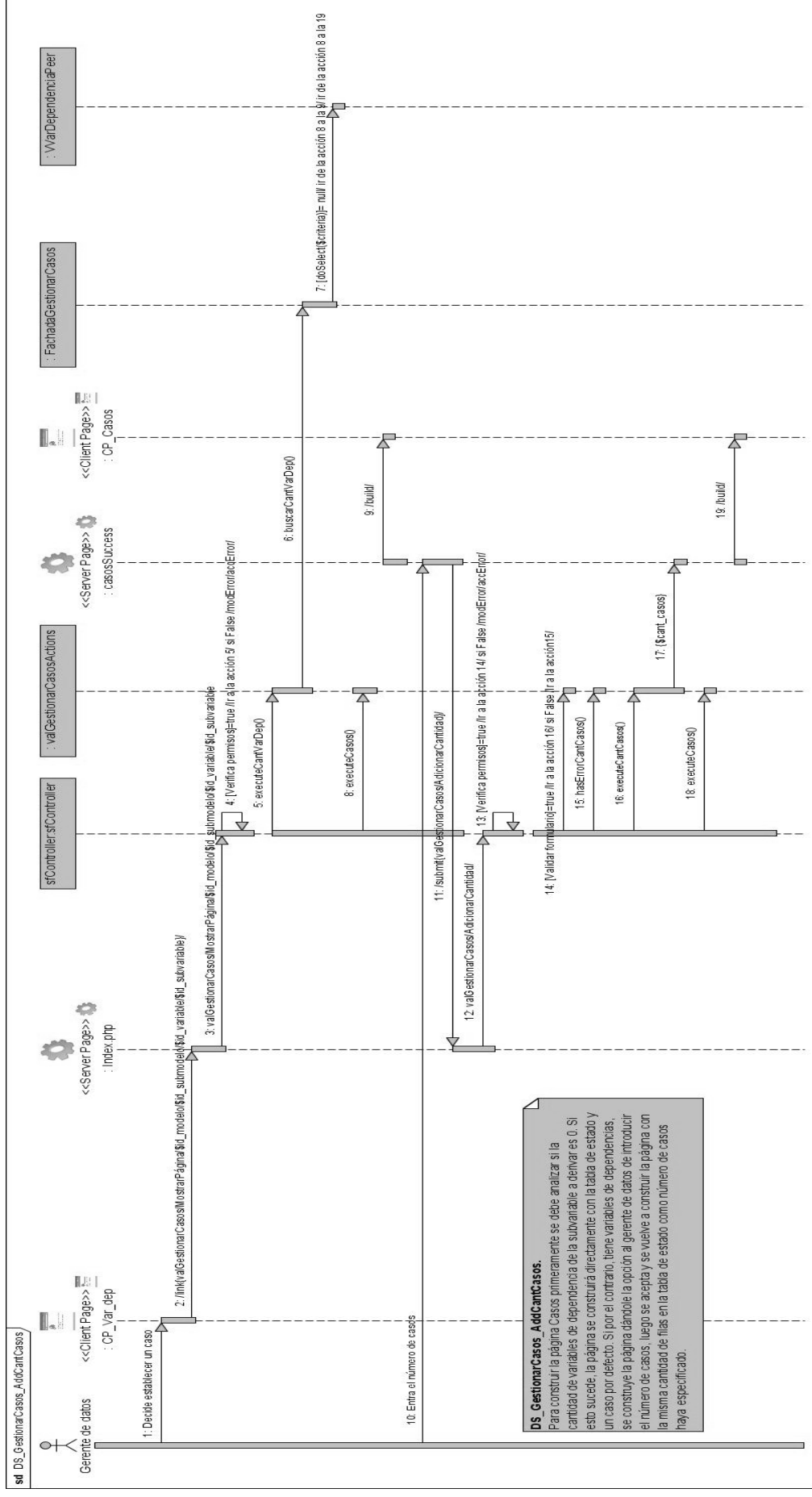


Figura 25 Eliminar variables de dependencia

Caso de uso: Gestionar casos



DS_GestionalCasos_AddCantCasos.
 Para construir la página Casos primeramente se debe analizar si la cantidad de variables de dependencia de la subvariable a derivar es 0. Si esto sucede, la página se construirá directamente con la tabla de estado y un caso por defecto. Si por el contrario tiene variables de dependencias, se constituye la página dándole la opción al gerente de datos de introducir el número de casos, luego se acepta y se vuelve a construir la página con la misma cantidad de filas en la tabla de estado como número de casos haya especificado.

Figura 26 Adicionar cantidad de casos

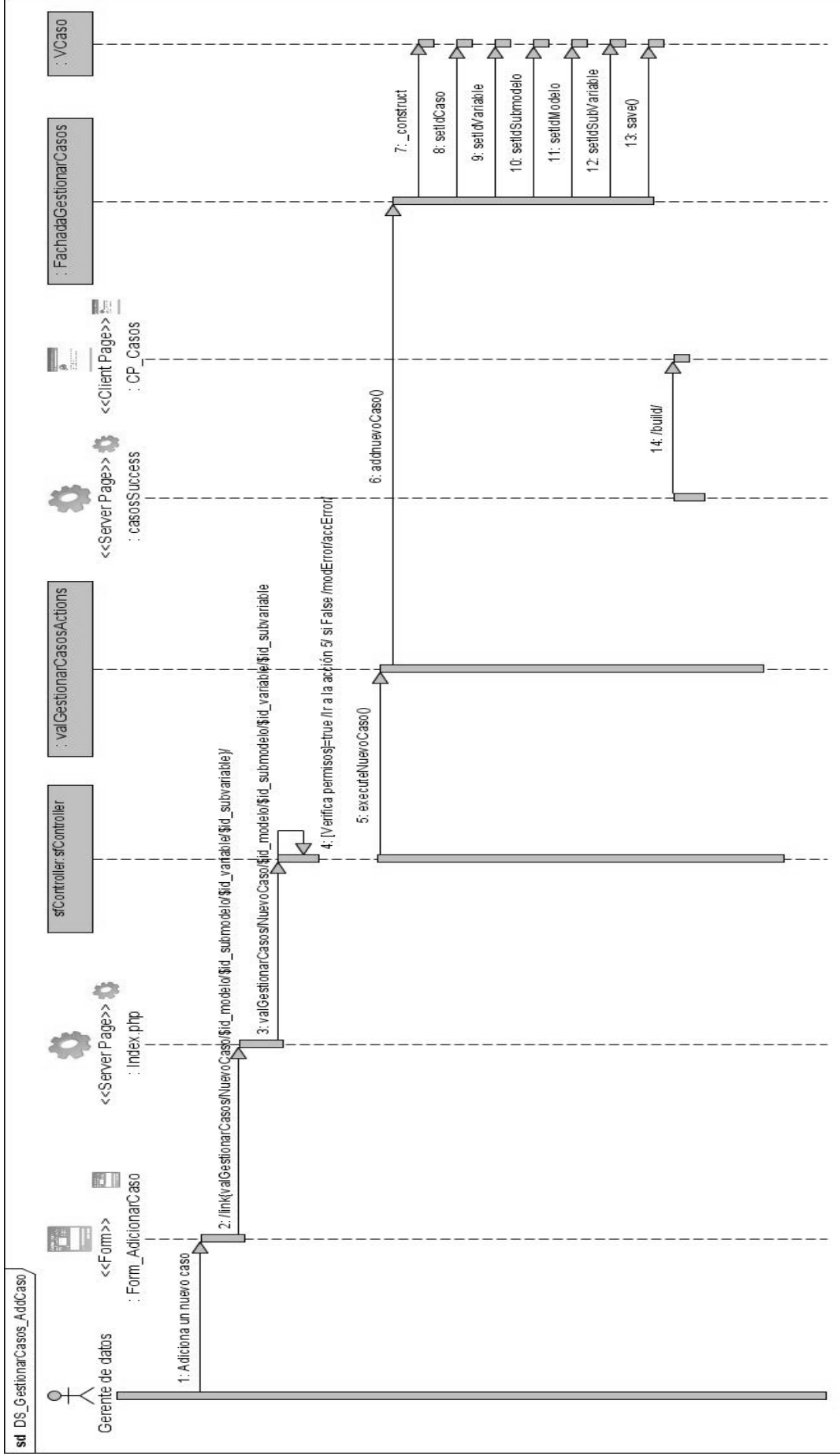


Figura 27 Adicionar casos

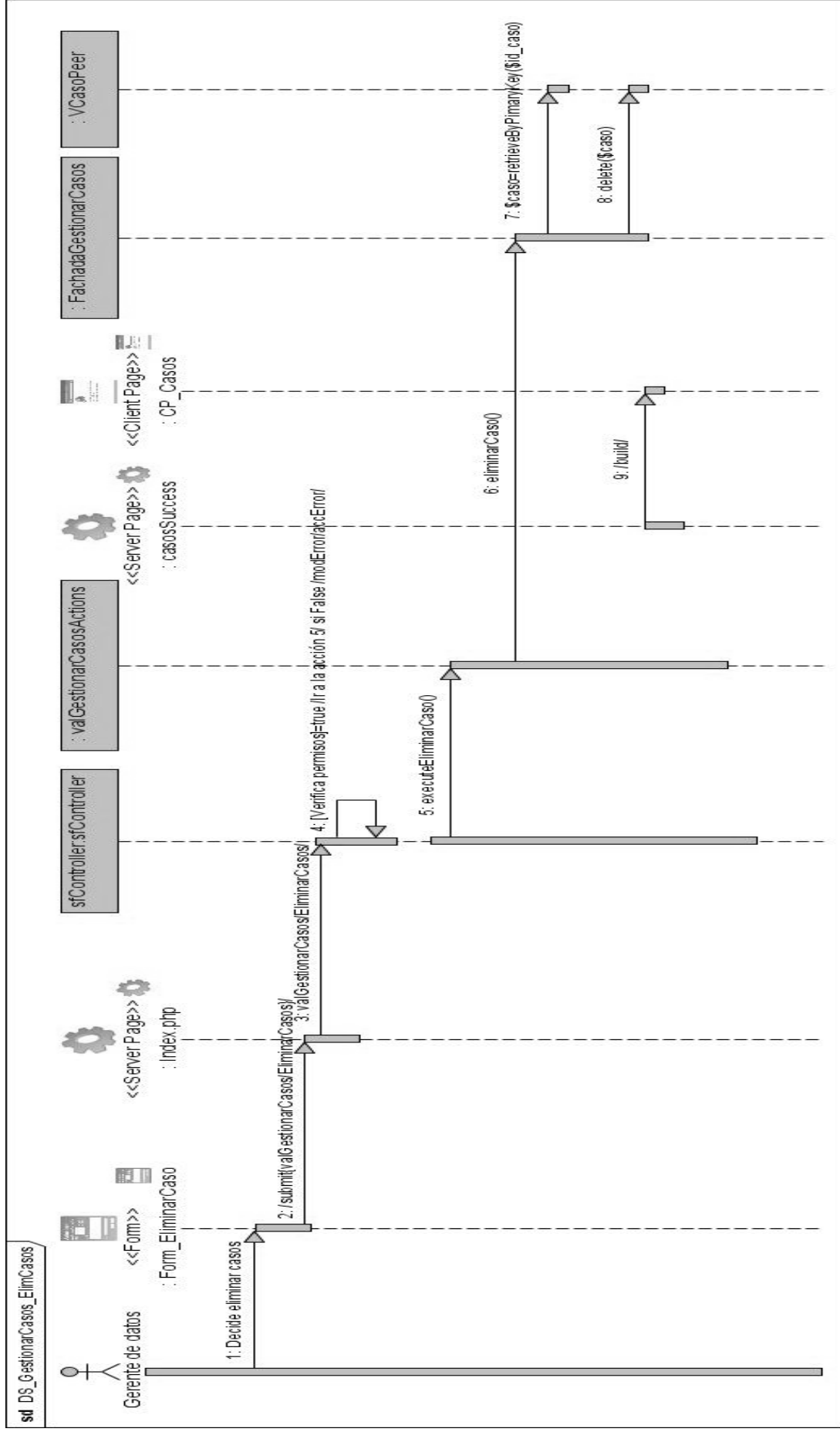


Figura 28 Eliminar casos

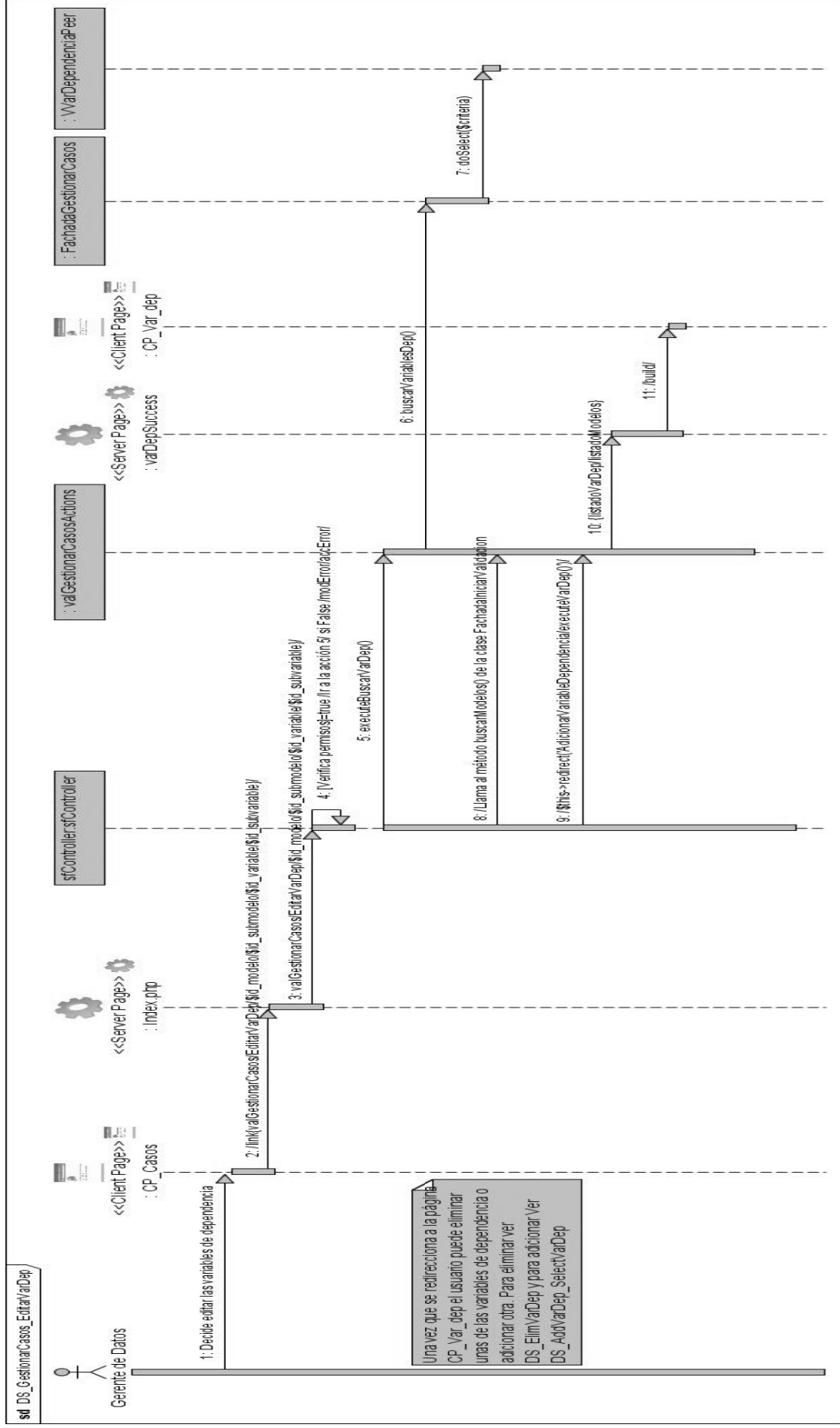


Figura 29 Editor variables de dependencia

Caso de uso: Derivar

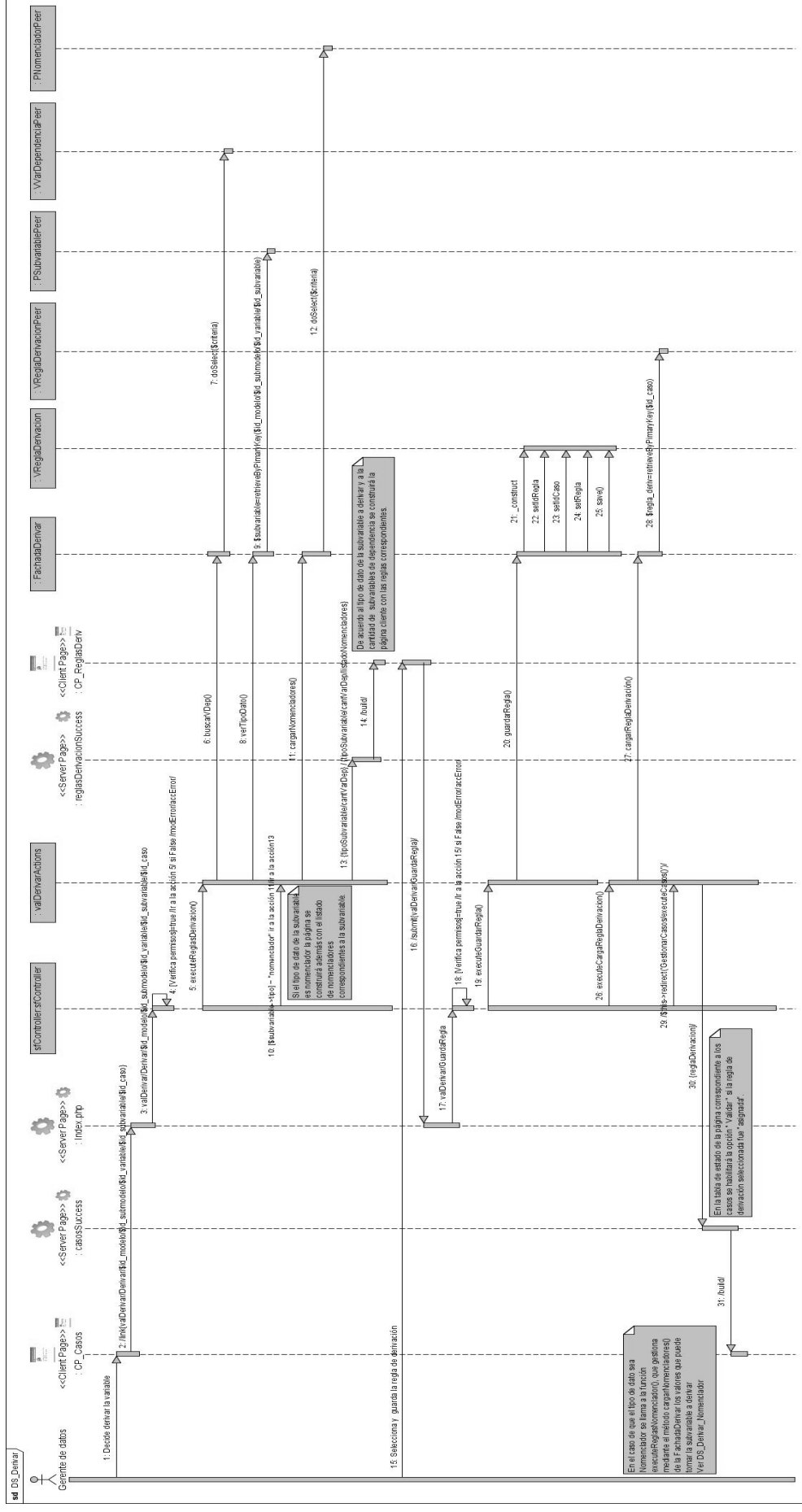


Figura 32 Derivar

2.4 Prototipos no funcionales

La confección de los prototipos de interfaz no funcionales para el módulo Validación: submódulo “Derivación de las variables del CRD”, facilita la comunicación con el cliente al visualizar si realmente el futuro sistema cumple con los requerimientos funcionales capturados en el flujo de trabajo Requerimientos; permiten verificar si está acorde a las necesidades planteadas y constituyen un punto de partida para los programadores, los cuales se centrarán en las funcionalidades del sistema haciéndolas coincidir con el diseño propuesto en estos prototipos.

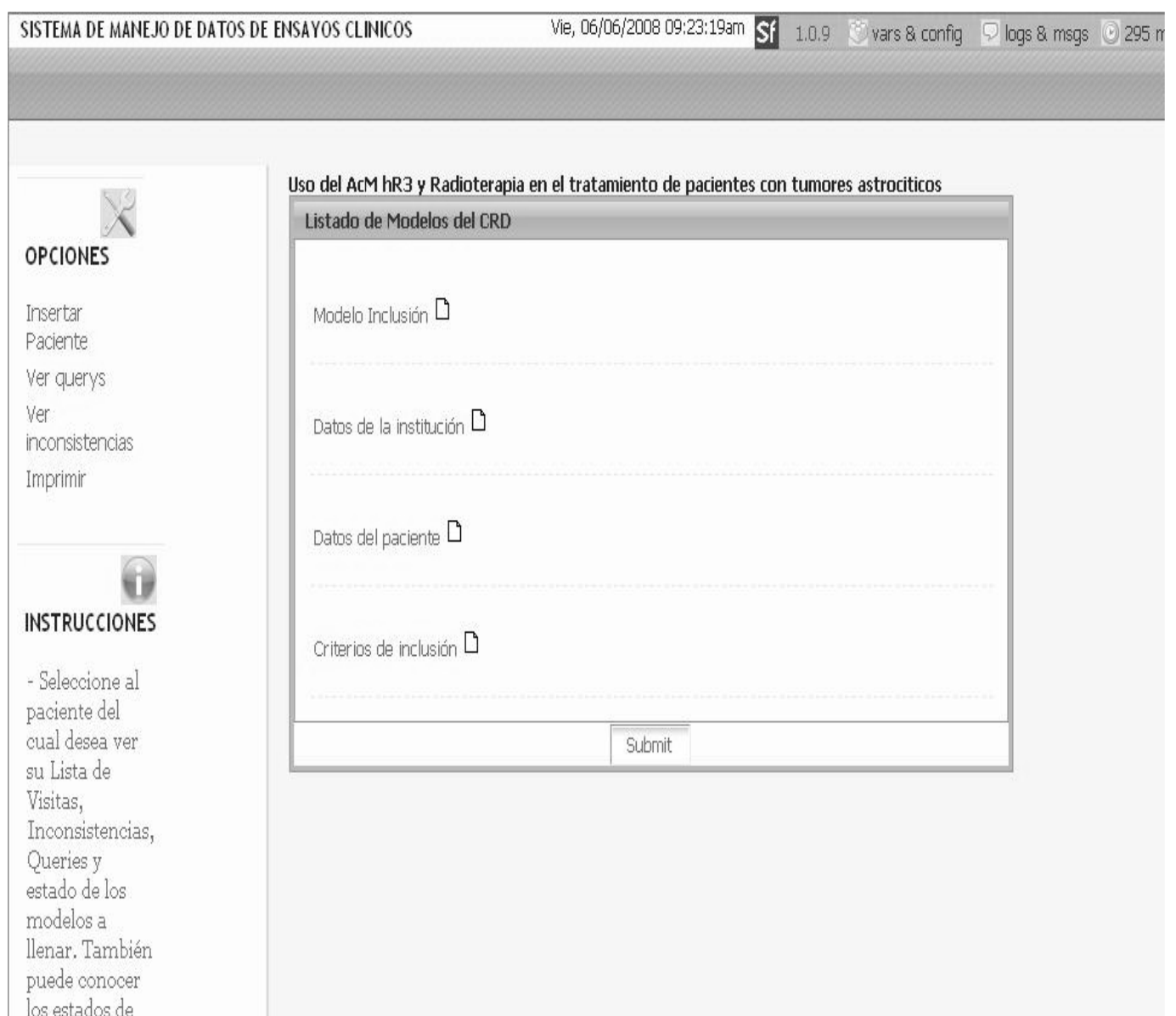


Figura 33 Caso de uso Iniciar validación: Listar modelos

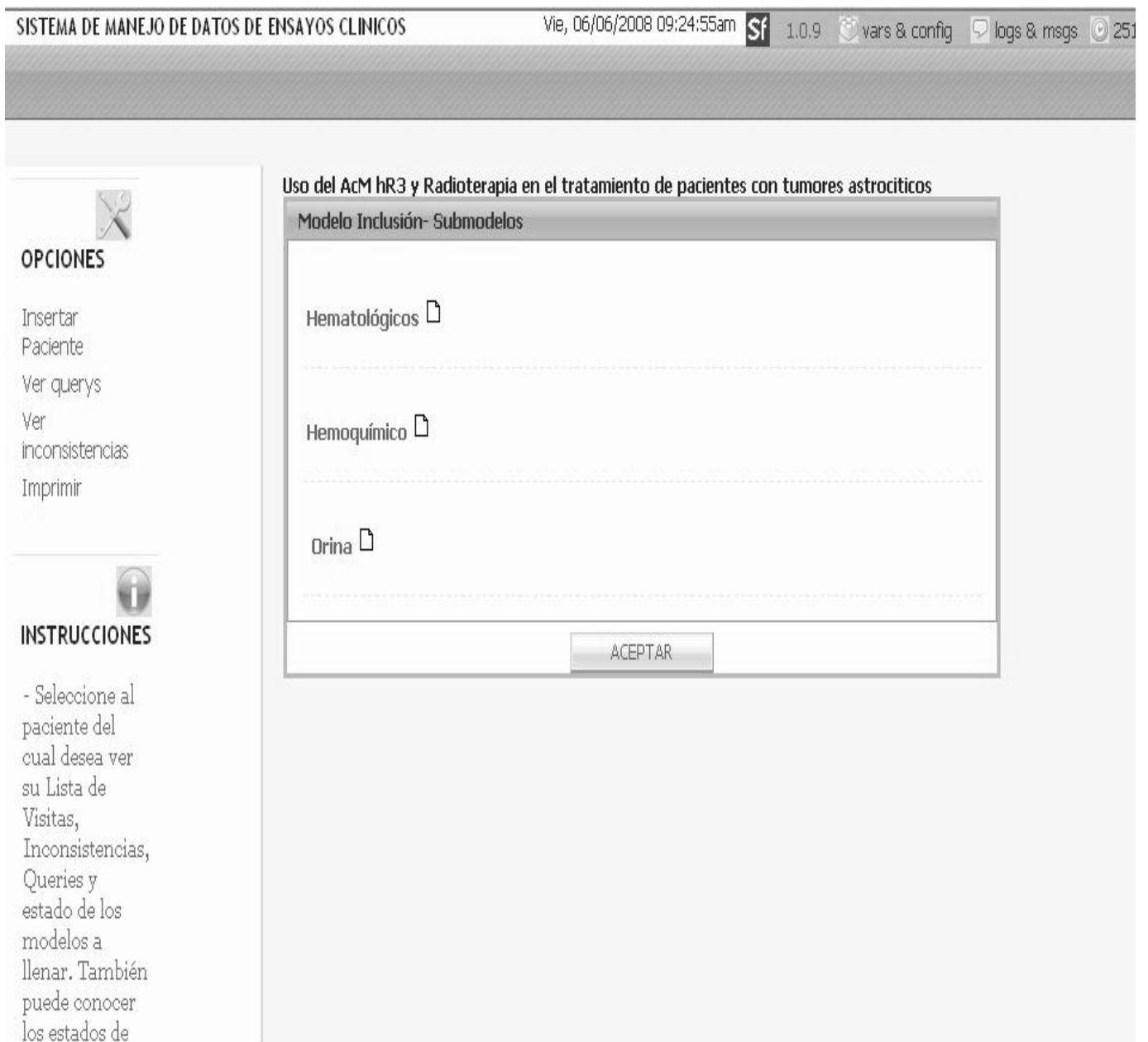


Figura 34 Caso de uso Iniciar validación: Listar submodelos

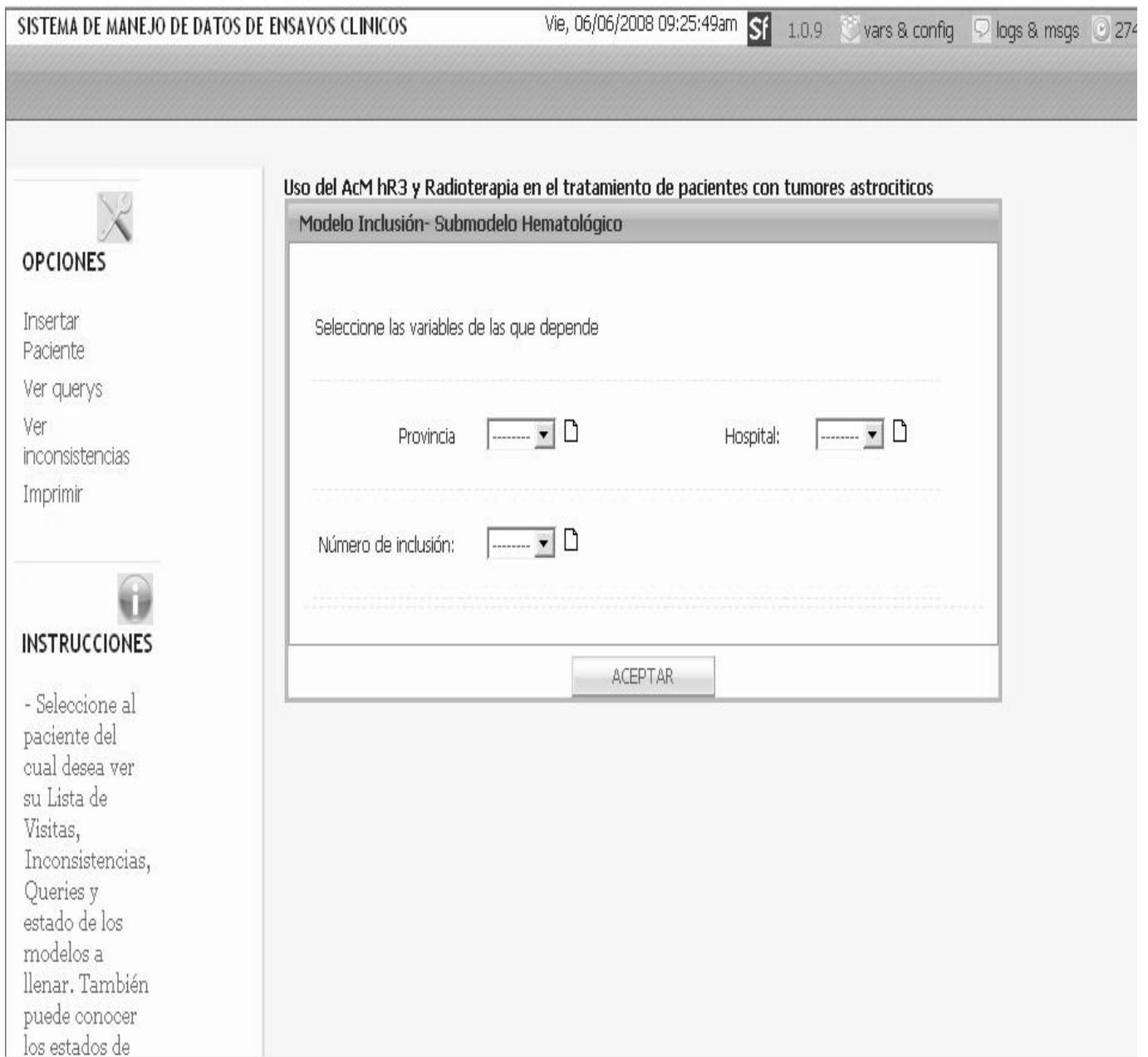



Figura 35 Caso de uso Iniciar validación: Variables de un submodelo

SISTEMA DE MANEJO DE DATOS DE ENSAYOS CLINICOS Vie, 06/06/2008 09:57:25am Sf 1.0.9 vars & config logs & msgs


 **OPCIONES**

Insertar Paciente

Ver querys

Ver inconsistencias

Imprimir

 **INSTRUCCIONES**

- Seleccione al paciente del cual desea ver su Lista de Visitas, Inconsistencias, Queries y estado de los modelos a llenar. También puede conocer los estados de

Uso del AcM hr3 y Radioterapia en el tratamiento de pacientes con tumores astrocíticos

Modelo Laboratorio clínico-Submodelo Hematológicos













Exámen	Resultado	Fecha de realizacion	Significado
Hemoglobina	<input type="text"/> 	Dia <input type="text"/> Mes <input type="text"/> Año <input type="text"/>  <input type="text"/> 
Leucograma con diferencial	<input type="text"/> 	Dia <input type="text"/> Mes <input type="text"/> Año <input type="text"/>  <input type="text"/> 
Hematocrito	<input type="text"/> 	Dia <input type="text"/> Mes <input type="text"/> Año <input type="text"/>  <input type="text"/> 
Plaquetas	<input type="text"/> 	Dia <input type="text"/> Mes <input type="text"/> Año <input type="text"/>  <input type="text"/> 

Figura 36 Caso de uso Iniciar validación: variables y subvariables de un submodelo

Modelo Inclusión-Submodelo Hematológico-Variable Hospital- Subvariable Hospital

Variable: Hospital

Submodelo: Hematológico

Cantidad de variables de las que depende:

Modelo: Submodelo:

Variable: Subvariable:


Inclusión **Datos Institución** **Provincia** **Provincia** 

Figura 37 Caso de uso Adicionar variables de dependencia

Modelo Inclusión-Submodelo Hematológico-Variable Hospital- Subvariable Hospital

Depende de: V1= 1 - Inclusión - Datos Institución Provincia Editar

Cantidad de casos:

Casos	Condiciones	Derivar	Validar
<input type="checkbox"/> Caso1	condición	derivar	validar
Por defecto		derivar	validar

Figura 38 Caso de uso Gestionar casos

Modelo Inclusión-Submodelo Datos del paciente-Variable Código de identificación

Depende de: V1= 1- Inclusión - Datos Institución - Nro inclusión

V2= 1- Inclusión - Datos Institución - Hospital

V3= 1- Inclusión - Datos del paciente- Iniciales del paciente

Condiciones de dependencia:

ASIGNADA NO ASIGNADA

> >= V1 V2 V3 1 2 3

V1: < <= Concat & 4 5 6

= <> + - * / ln raiz || () . 0 7 8 9

V2: ASIGNADA NO ASIGNADA

= <>

V3: ASIGNADA NO ASIGNADA

?

Figura 39 Caso de uso Establecer condiciones

Uso del AcM hR3 y Radioterapia en el tratamiento de pacientes con tumores astrocíticos

Modelo Inclusión-Submodelo Hematológico-Variable Hospital- Subvariable Hospital

Depende de: V1= 1- Inclusión - Datos Institución - Provincia

Condiciones: V1=Las Tunas

Vx: Campo obligatorio

Valores

<> =

=

Ernesto Guevara
Pitti Fajardo
Frank Pais

Cerrar

Figura 40 Caso de uso Derivar

Modelo Laboratorio clínico-Submodelo Datos del Paciente- Variable Código de Identificación

Depende de: V1= 1 - Inclusión - Datos Institución - No inclusión

V2= 1 - Inclusión - Datos Institución - Hospital

V3= 1 - Inclusión - Datos del paciente- Iniciales del paciente Editar

Casos	Condiciones	Derivar Vx	Validar Vx
<input type="checkbox"/> Caso1	V1 Asignada V2 Asignada V3 Asignada	$Vx=V2-V3-V1$	---
<input type="checkbox"/> Por defecto		derivar	validar

Figura 41 Resultado del caso de uso Derivar

2.5 Mapa de Navegación

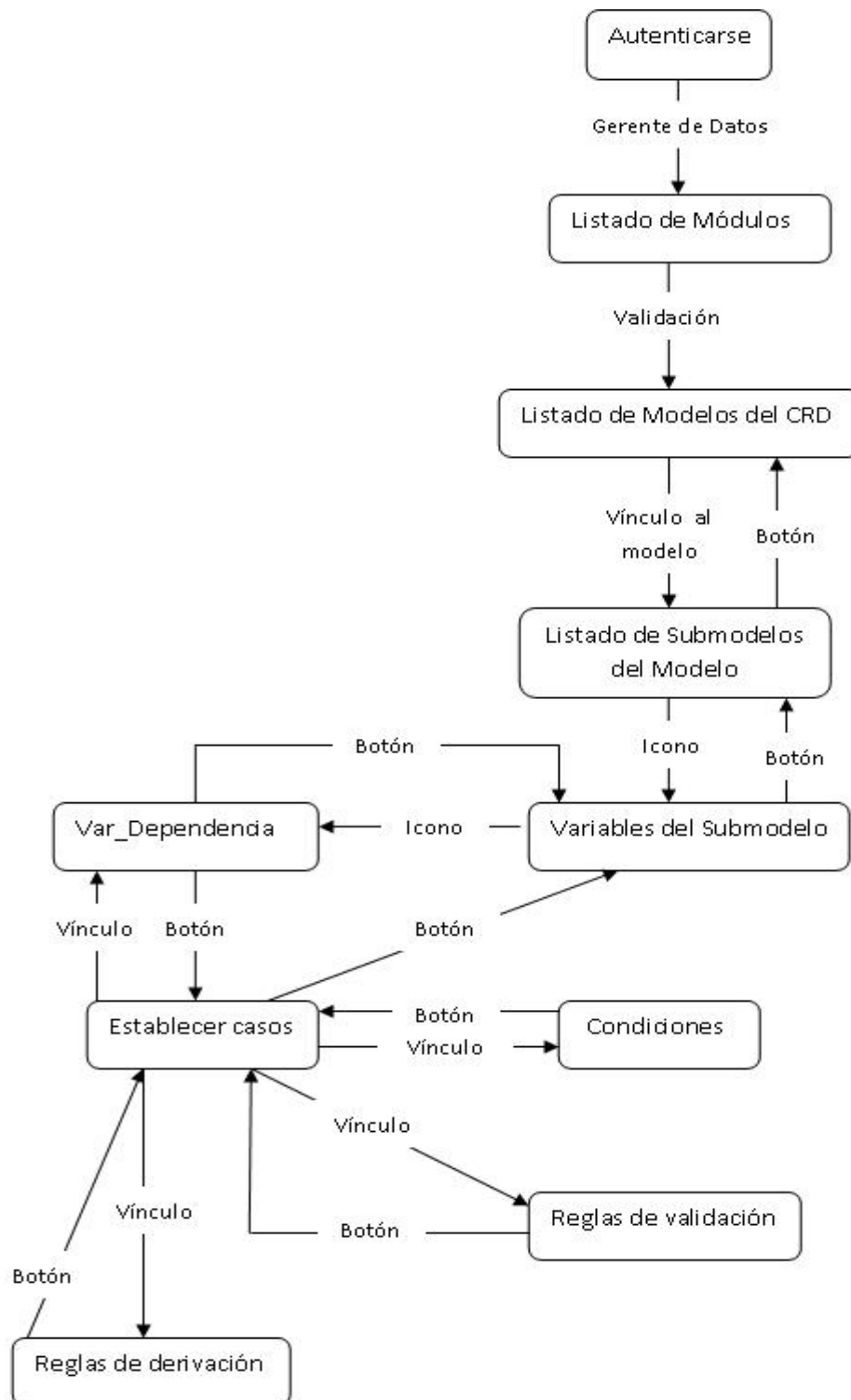


Figura 42 Mapa de Navegación

2.6 Clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo, por tanto todas las clases definidas e identificadas en el dominio del análisis no son persistentes. Lo serán solo aquellas cuya información sea necesaria guardar en la Base de Datos. El objetivo de definir las clases persistentes radica en modelar la información y el comportamiento asociado a algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso.

2.6.1 Diagrama de clases persistentes

El Diagrama de Clase se utiliza para modelar la estructura lógica de la BD, con clases representando tablas, y atributos de clase representando columnas. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos. Para analizar el diagrama de clases persistente del proyecto SIMDECC en su totalidad, remitirse al Expediente de Proyecto. A continuación se muestra el diagrama de clases persistentes correspondiente al submódulo.

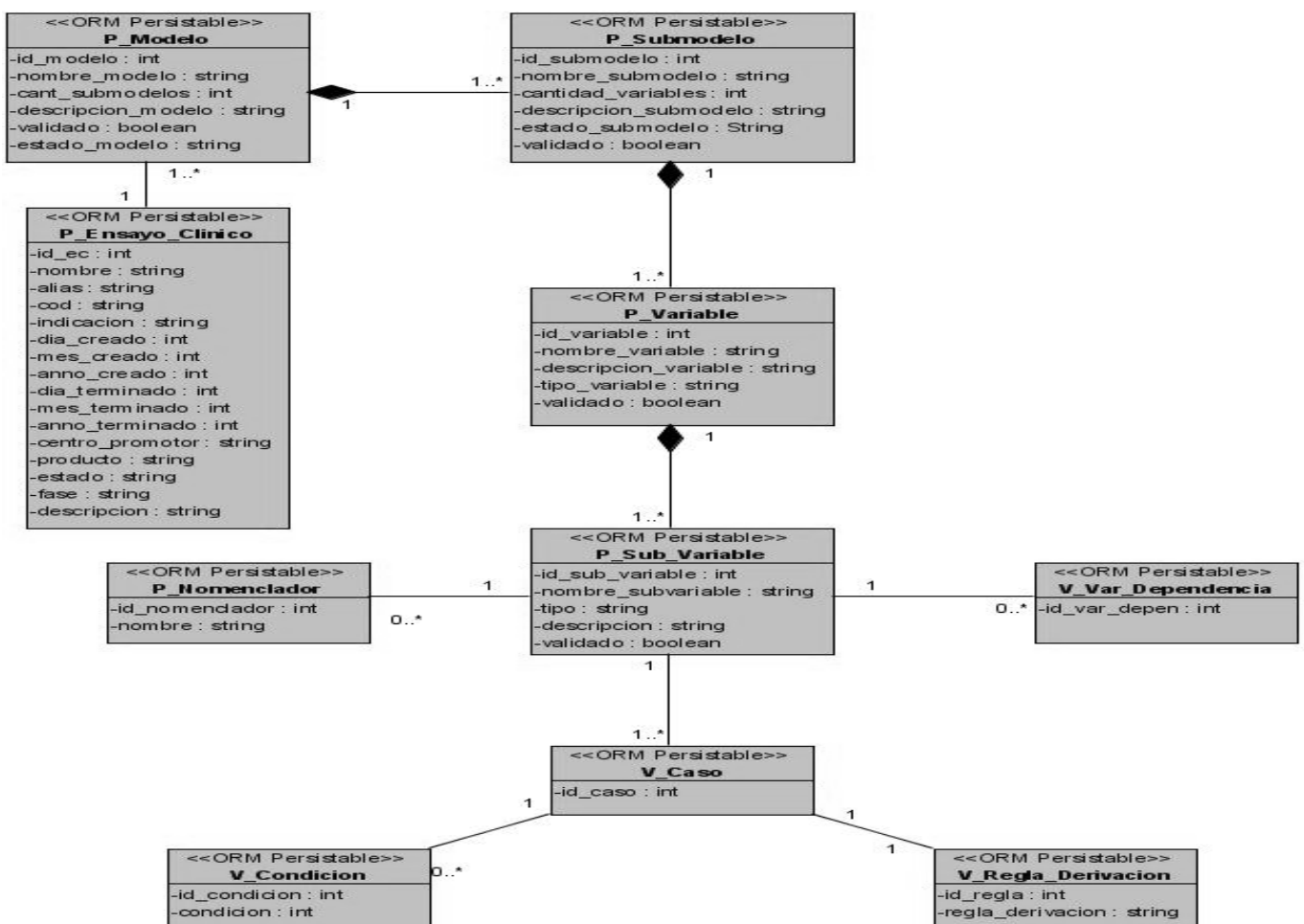


Figura 43 Diagrama de clases persistentes

2.7 Modelo de Datos

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Hay dos tipos de modelos de datos: los modelos conceptuales y los modelos lógicos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender. En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos. A continuación se muestra el modelo de datos del submódulo.

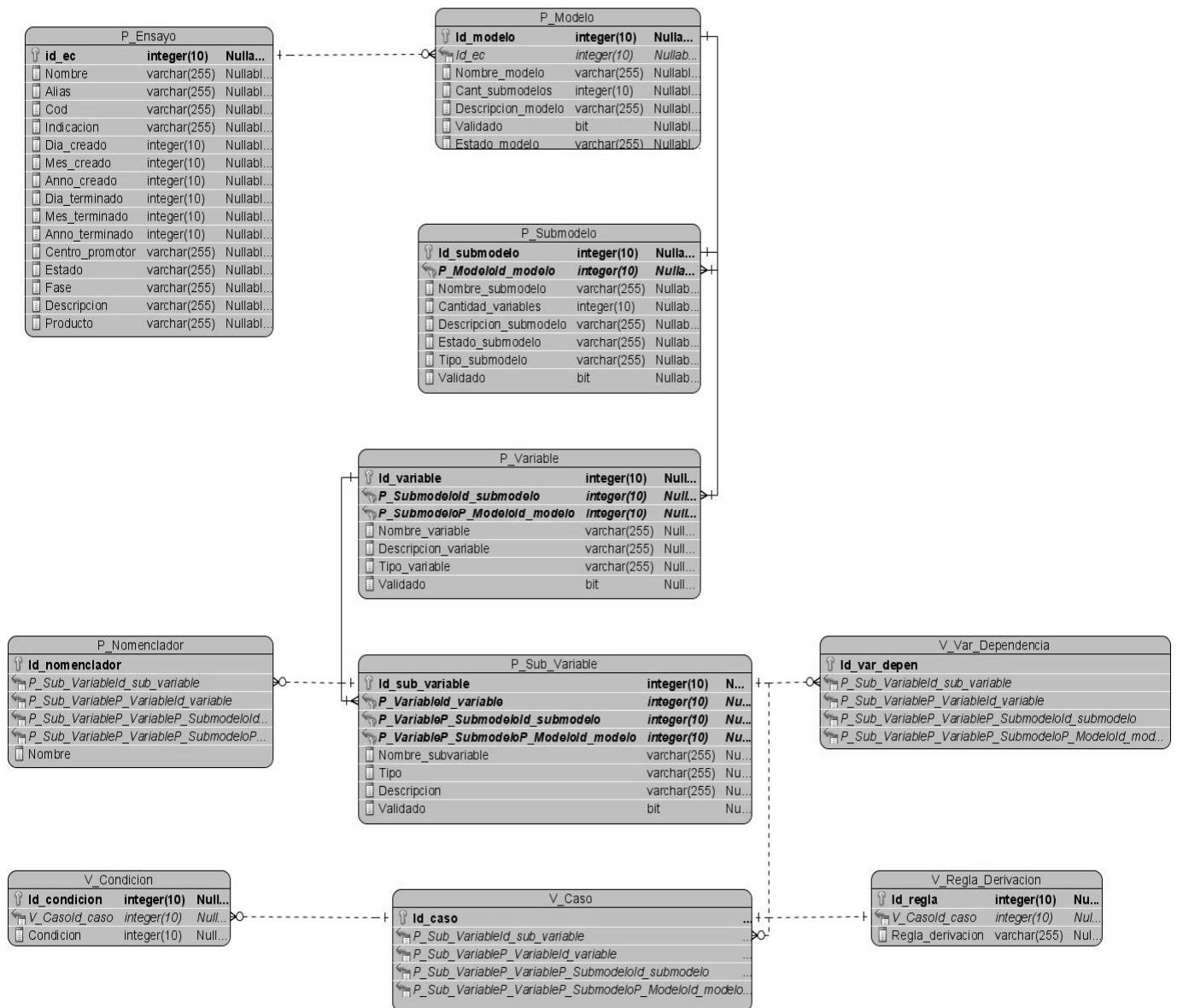


Figura 44 Modelo de datos

2.8 Diagrama de Despliegue

La aplicación web, resultado de la presente investigación estará distribuida de la siguiente forma:

En el Centro de Inmunología Molecular se encontrará la aplicación servidora, a la que estarán conectadas todas las PCs clientes del centro, y también se conectarán a esa aplicación otras PCs de otras partes del país con la debida autorización del centro y configurado en el servidor Firewall + Proxy. Existirán impresoras para la impresión de los modelos y otros datos de importancia, esto fue un requisito del cliente, pero en caso de faltar este dispositivo no afectará en nada el funcionamiento de la aplicación. Habrá un servidor de bases de datos, que será la base de datos primario, donde se encontrarán todos los datos del sistema, y se realizarán copias de resguardo de la información en otro servidor.

La aplicación principal estará en el CIM.

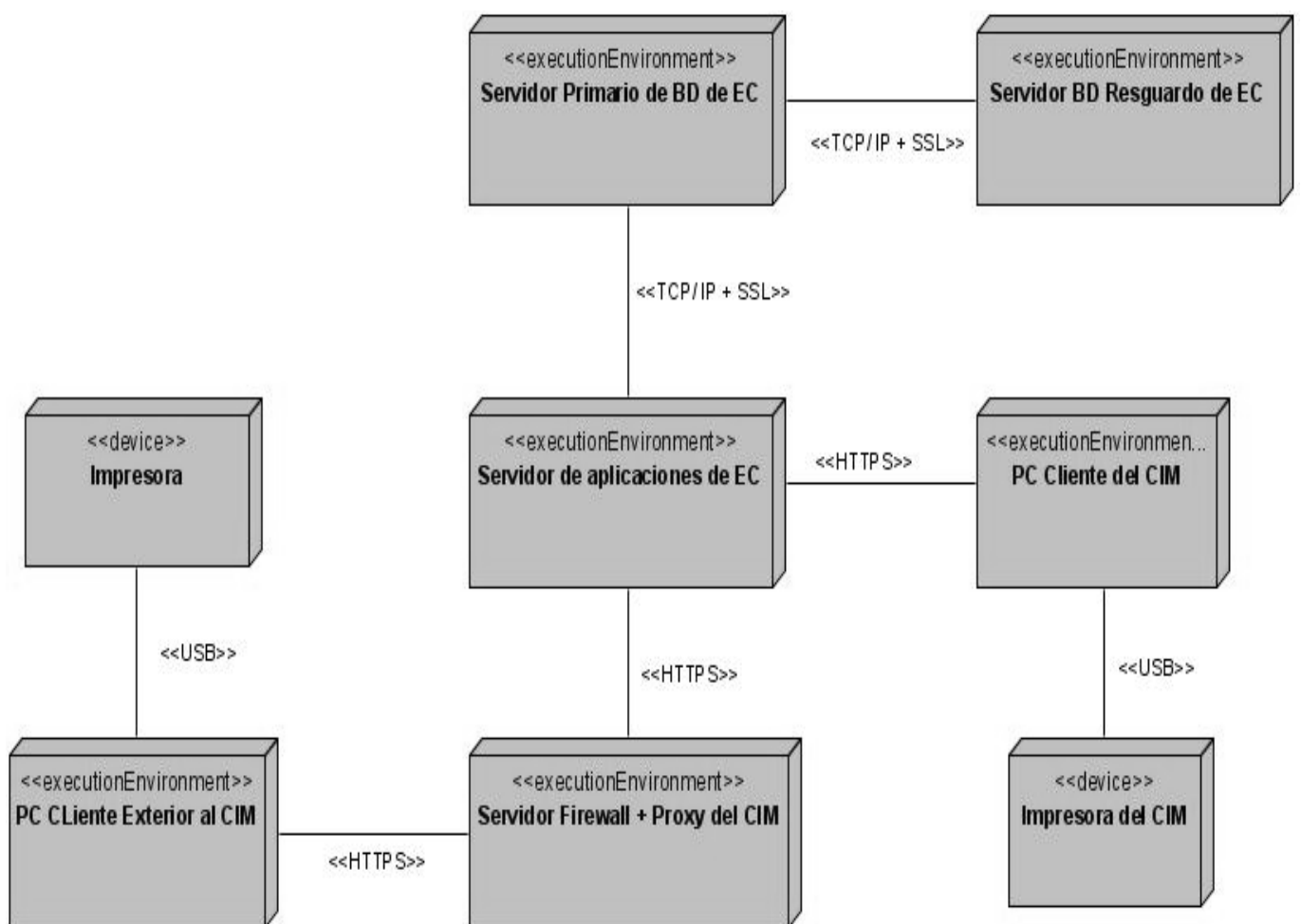


Figura 45 Modelo de despliegue de la aplicación

A partir del diagrama de despliegue anteriormente expuesto se define la distribución de las capas lógicas en los nodos de procesamiento de la siguiente forma:

- *En el nodo Servidor de Aplicaciones del EC son ubicadas las capas lógicas Vista, Controlador, Modelo y Servicios.*
- *En los nodos PC Cliente del CIM y PC Cliente Exterior al CIM se presentaría el resultado de la capa Vista en cada petición, producido mediante código HTML.*
- *Dentro de los nodos Servidor Primario de BD del EC y Servidor BD Resguardo de EC estaría la Base de datos y salvas, respectivamente (no forman parte de las capas lógicas, se presentan para una mayor claridad).(16)*

2.9 Validación del diseño

Para obtener una aplicación que cumpla con los requisitos exigidos por el cliente y consiga su total satisfacción es importante haber logrado una implementación lo más organizada y correcta posible. Pero a su vez, un buen diseño, capaz de hacer comprender a los programadores en detalle toda la lógica que requiere el sistema, garantiza que esto ocurra con mayor claridad.

La presente investigación propone un diseño validado desde la perspectiva de los futuros implementadores del módulo Validación del Sistema de Manejo de Datos de Ensayos Clínicos Cubano, quienes afirman comprender los diagramas de clases del diseño del submódulo “Derivación de las variables del CRD”, así como los diagramas de interacción que muestran los flujos de mensajes con cada acción y método que debe ejecutarse en un momento determinado. Opinaron favorablemente en cuanto a algunas ideas arrojadas por el diseño para su futuro trabajo, como por ejemplo el uso de los elementos parciales para el caso de uso “Derivar”. Garantizan poder desarrollar el prototipo funcional a partir de la investigación, dado por la comprensión de los diagramas expuestos, con ayuda de la descripción textual de las clases del diseño, brindando un entendimiento sobre el por qué de cada una de las funciones propuestas para las clases fundamentales. Aseguran contar con un diseño que brinda transparencia y sirve de guía para el desarrollo de la fase siguiente, orientándolos paso a paso sobre cómo proceder para hacerlo y de este modo cumplir con las expectativas que se esperan para el sistema.

A continuación se exponen las opiniones de los implementadores del Módulo.

Leonelbys Herrera Pérez:

“Pienso que el trabajo realizado por las diseñadoras del submódulo “Derivación de las variables del CRD” en el proyecto Ensayos Clínicos tiene una alta calidad. Dicho trabajo resolverá grandes dificultades que existían en torno al desarrollo del módulo Validación, aporta nuevas ideas de implementación, ideas que pueden ser captadas rápidamente por los programadores debido a la organización y claridad con que cuenta. Además, pienso que es un trabajo que hay que tomar bien en serio ya que ha sido desarrollado con mucho cuidado, indagando con los implementadores y así llegar a una decisión final que será muy importante con vistas a la culminación de la fase de implementación del software SIMDECC”.

Leandro Evangelio Hernández Cuello:

“Como futuro programador del módulo validación creo que el diseño realizado es de mucha utilidad para el momento de la implementación. Cuenta con un alto nivel de detalle en cuanto a los métodos a utilizar, las clases del modelo de datos que se usan por cada caso de uso así como el flujo de mensajes mostrados en los diagramas de secuencia. Las descripciones de los casos de uso del sistema y los diagramas de secuencia coinciden totalmente. Por tanto no dudo en que este diseño pueda servirme de mucha ayuda para mi trabajo como implementador”.

Para una lograr una mejor validación del diseño propuesto, se hizo además una revisión de la implementación realizada por los programadores del módulo Administración del SIMDECC perteneciente al proyecto Ensayos Clínicos, donde, por las características de la arquitectura que tributa al empleo de varios patrones comunes para todos los módulos, se logra que tanto el diseño como la implementación de los diferentes módulos coincidan en determinados aspectos de forma general. Por tanto, al consultar el código se puede asumir que el diseño de la investigación resulta factible para una futura implementación, pues son muy similares en cuanto al llamado de los métodos, clases, así como al uso de determinadas facilidades del framework. Por ejemplo, para el caso de uso Iniciar validación en el que se requiere listar los modelos (ver Figura 20) y submodelos del CRD se necesita llamar a la clase FachadaIniciarValidacion, que se encarga de gestionar las clases de la capa Modelo. De esta forma se captura de la base de datos la información requerida usando un objeto de la clase Criterias para a partir de él devolver el resultado que se espera y ser mostrado una vez que se pasen a la Vista las variables o parámetros.

A continuación se muestran fragmentos del código antes mencionado.

Método de la clase **adm_PrincipalActions**

```
public function executeIndex()
{
    $usuario = $this->getUser();
    $this->mens = 0;
    $id_user = $usuario->getUsuarioActivo()->idUsuario;
    $ip_user = $usuario->getUsuarioActivo()->ip;
    $arr_ensayos=FachadaAdministracion::ensayosPorUsuarioAutenticado($id_user,$ip_user);
    //Uso de la clase Fachada perteneciente a la capa Modelo//

    $this->todosensayos = $arr_ensayos;
    //paso de parámetros a la capa Vista.
    ...
}
```

Método de la clase **FachadaAdministracion**

```
public static function ensayosPorUsuarioAutenticado($id_user,$ip_user)
{
    $c = new Criteria();
    // uso de una instancia de la clase Criteria

    $c->add(ARolUselpPeer::ID_USUARIO, $id_user);
    $articulos = ARolUselpPeer::doSelect($c);
    // uso del método doSelect() de las clases Peer de la capa Modelo

    return $arr_ensayos;
}
```

El diseño que se propone en el trabajo de diploma sirve como punto de partida para la posterior implementación del submódulo. El nivel de detalle de los diagramas de interacción, la claridad en las descripciones de clases y la estructura de los diagramas de clases, conjuntamente con el conocimiento que tienen los programadores sobre el lenguaje de modelado y el framework de desarrollo, hacen posible que pueda obtenerse un código legible y claro, de acuerdo a las funcionalidades requeridas.

Conclusiones

En este capítulo se definen las clases del diseño, organizadas por capas (Modelo, Vista, Controlador), de acuerdo a la arquitectura seleccionada para el desarrollo de la solución. Estas clases están agrupadas en los paquetes del diseño, donde se tienen además los diagramas de interacción que muestran todo el flujo de mensajes entre las clases que intervienen en los diagramas de clases del diseño. Se exponen además los prototipos no funcionales y el mapa de navegación los cuales facilitarán el trabajo de los programadores. Por último se presenta el diagrama de despliegue que expresa la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

Conclusiones generales

El estudio de los procesos de gestión de la información de aplicaciones web para la derivación de las variables en el CIM, propició el desarrollo del presente trabajo, cuyo resultado fue el diseño del submódulo: “Derivación de las variables del Cuaderno de Recogida de Datos” perteneciente al módulo Validación, para el SIMDECC en el CIM, atendiendo al objetivo propuesto para la investigación. Para ello se logró específicamente:

- El diseño de los prototipos no funcionales del submódulo, permitiendo una mejor comunicación con el cliente y logrando un punto de partida para la implementación del sistema.
- La aplicación de los patrones de diseño orientado a objetos, con el objetivo de evitar o corregir problemas existentes mediante la reutilización de las soluciones propuestas por cada uno de ellos.
- La obtención de los artefactos correspondientes al flujo de trabajo Análisis y Diseño, constituyendo una guía de mucha utilidad para la futura implementación de la solución propuesta.

El trabajo de diploma marcará un punto de partida para los implementadores del futuro sistema en aras de disminuir los errores en los datos de los Ensayos Clínicos, propiciando de esta forma, que las pruebas realizadas para la evaluación de nuevos fármacos que combatan enfermedades tan graves como el cáncer, logren la calidad y veracidad requerida por los especialistas del CIM.

Recomendaciones

- Se recomienda realizar la futura implementación del SIMDECC, basándose en el diseño propuesto, para obtener un producto funcional acorde a las necesidades del CIM.
- El diseño propuesto puede ser de utilidad a otros sistemas que gestionen información y necesiten una validación compleja para lograr la integridad de los datos, de ahí que se recomiende su uso en aquellas aplicaciones que lo requieran.

Referencias bibliográficas

1. **López Díaz, Aidacelys y Rodríguez García, Lucía.** *Sistema de Manejo de Datos de Ensayos Clínicos. Tesis (Ingeniero en Ciencias Informáticas)* Ciudad Habana. Universidad de las Ciencias Informáticas, 2007.
2. Pivotal. [En línea] [Citado el: 26 de 3 de 2008.] <http://www.pivotal.es/extranet/servicios>.
3. Foro Hipócrates en Internet. [En línea] [Citado el: 26 de 3 de 2008.] <http://www.hipocrates.com/>
4. Entorno Virtual de Aprendizaje. *Material de Apoyo Conferencia 1 Introducción a la Ingeniería de Software.* [En línea] 16 de septiembre de 2007. [Citado el: 12 de junio de 2008.] <http://teleformacion.uci.cu/mod/resource/view.php?id=6655>
5. Appcelerator Developer Network. [En línea] [Citado el: 25 de 3 de 2008.] <http://www.appcelerator.org/index.html>
6. Comunidad Imperial de Desarrolladores Linux. [En línea] <http://www.linperial.com/communities/forums/developers/?q=node/69>
7. Kumbia PHP Framework. [En línea] [Citado el: 24 de 3 de 2008.] <http://kumbiaphp.com/>
8. Blog de Programación. [En línea] diciembre de 2007. [Citado el: 15 de marzo de 2008.] <http://blogdeprogramacion.blogspot.com/search/label/Symfony>
9. Osmosis Latina. [En línea] 2005. [Citado el: 15 de marzo de 2008.] http://www.osmosislatina.com/aplicaciones/servidor_web.htm
10. Ciberaula. [En línea] 2006. [Citado el: 15 de marzo de 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro/
11. Entorno Virtual de Aprendizaje. *Material de Apoyo Conferencia Diseño.* [En línea] 22 de febrero de 2008. [Citado el: 15 de marzo de 2008.] <http://teleformacion.uci.cu/mod/resource/view.php?id=21363>
12. **Rodríguez Luque, Daniel y Rodríguez Luque, David.** *Herramienta para la generación de código de aplicaciones Web.* Ciudad Habana : s.n., 2007.

13. **Dodero, Juan Manuel y Fernández Llamas, Camino.** *Patrones estructurales: Decorator.* Madrid : s.n., 2002.
14. **Larman, Craig.** *UML y Patrones: Introducción al análisis y diseño orientado a objetos.(2).* La Habana : Félix Varela, 2004.
15. **Ciudad Ricardo, Febe Ángel.** Monografías.com. *Utilización del Patrón Modelo – Vista – Controlador (MVC) en el diseño de software educativos.* [En línea] abril de 2006. [Citado el: 20 de marzo de 2008.]
<http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml>
16. AJAX, un nuevo acercamiento a aplicaciones web. [En línea] [Citado el: 12 de junio de 2008.]
<http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php>.
17. **Eguíluz Pérez, Javier.** Capítulo 1. Introducción a AJAX. *Introducción a AJAX.*
18. **Ballester Marsal, Andrés.** *Documento de Arquitectura de Software(Ensayos Clínicos).* Ciudad Habana : s.n., 2008.

Bibliografía Consultada

1. **Rodríguez García, Lic. Lucía y López Díaz, Lic. Aidacelys.** *Sistema de Manejo de Datos de Ensayos Clínicos: Módulo de diseño.* Ciudad Habana : s.n., 2007.
2. **Larman, Craig.** *UML y Patrones: Introducción al análisis y diseño orientado a objetos.(2).* La Habana : Félix Varela, 2004.
3. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* Cuarta Edición.
4. Ayuda Extendida del Rational.
5. Pivotal. [En línea] [Citado el: 26 de 3 de 2008.] <http://www.pivotal.es/extranet/servicios>
6. Foro Hipócrates en Internet. [En línea] [Citado el: 26 de 3 de 2008.] <http://www.hipocrates.com/>
7. **Mendoza Sánchez, María A.** Informatizate. *Metodologías De Desarrollo De Software.* [En línea] 7 de 6 de 2004. [Citado el: 27 de 1 de 2008.]
[http://www.informatizate.net/articulos/metodologias de desarrollo de software 07062004.html](http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html)
8. Appcelerator Developer Network. [En línea] [Citado el: 25 de 3 de 2008.]
<http://www.appcelerator.org/index.html>
9. Comunidad Imperial de Desarrolladores Linux. [En línea]
<http://www.linperial.com/communities/forums/developers/?q=node/69>
10. Kumbia PHP Framework. [En línea] [Citado el: 24 de 3 de 2008.] <http://kumbiaphp.com/>
11. Blog de Programación. [En línea] diciembre de 2007. [Citado el: 15 de marzo de 2008.]
<http://blogdeprogramacion.blogspot.com/search/label/Symfony>
12. Osmosis Latina. [En línea] 2005. [Citado el: 15 de marzo de 2008.]
[http://www.osmosislatina.com/aplicaciones/servidor web.htm](http://www.osmosislatina.com/aplicaciones/servidor_web.htm)

13. Ciberaula. [En línea] 2006. [Citado el: 15 de marzo de 2008.]
http://linux.ciberaula.com/articulo/linux_apache_intro/
14. Entorno Virtual de Aprendizaje. *Material de Apoyo Conferencia Diseño*. [En línea] 22 de febrero de 2008. [Citado el: 15 de marzo de 2008.] <http://teleformacion.uci.cu/mod/resource/view.php?id=21363>
15. **Rodríguez Luque, Daniel y Rodríguez Luque, David**. *Herramienta para la generación de código de aplicaciones Web*. Ciudad Habana : s.n., 2007.
16. **Dodero, Juan Manuel y Fernández Llamas, Camino**. *Patrones estructurales: Decorator*. Madrid : s.n., 2002.
17. **Larman, Craig**. *UML y Patrones: Introducción al análisis y diseño orientado a objetos.(2)*. La Habana : Félix Varela, 2004.
18. **Ciudad Ricardo, Febe Ángel**. Monografías.com. *Utilización del Patrón Modelo – Vista – Controlador (MVC) en el diseño de software educativos*. [En línea] abril de 2006. [Citado el: 20 de marzo de 2008.] <http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml>
19. **Ballester Marsal, Andrés**. *Documento de Arquitectura de Software(Ensayos Clínicos)*. Ciudad Habana : s.n., 2008.
20. **Franco Navarro, Ing. Jose Angel**. *UML en acción. Modelando Aplicaciones Web*. Ciudad Habana, : s.n.
21. **Potencier, Fabien y Zaninotto, François**. *Symfony, la guía definitiva*. 2007.
22. **María Mercedes Marqués Andrés**. Apuntes de ficheros y Bases de Datos. [En línea] <http://www3.uji.es/~mmarques/f47/apun/apun.html>
23. AJAX, un nuevo acercamiento a aplicaciones web. [En línea] [Citado el: 12 de junio de 2008.] <http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php>.
24. **Eguíluz Pérez, Javier**. Capítulo 1. Introducción a AJAX. *Introducción a AJAX*.

Anexos

Requisitos no funcionales del sistema

Usabilidad

Descripción del requisito

- Las personas que interactuarán con el software serán médicos, clínicos y especialistas de la salud ubicados en el CIM, CIGB, Instituto Finlay, CENCEC y todos los hospitales del país.
- La aplicación tendrá un ambiente sencillo y será fácil de manejar para los usuarios incluso aquellos que no han tenido mucha experiencia en el trabajo con computadoras o con sistemas informáticos.
- Se impartirá una preparación a los usuarios con la explicación de como se realizará el trabajo con el software.
- El sistema contendrá un manual de usuario, que será usado como ayuda para el trabajo con la aplicación.

Fiabilidad

Descripción del requisito

- El acceso a cualquier manipulación del sistema, tanto entrada como análisis de datos debe estar sometido a un proceso de autenticación del usuario donde será especificado el rol, usuario y contraseña.
- Las contraseñas deberán tener más de 7 caracteres de longitud y tener una fortaleza media.
- Los usuarios estarán obligados a cambiar la contraseña cada 60 días como máximo.
- Cada usuario tendrá asignado uno o varios roles en el sistema. Cada rol definido tendrá niveles de acceso al Software.
- Solo podrán acceder a la aplicación, clientes a través de direcciones IP específicas bien controladas.
- Todo cambio o modificación en el sistema debe ser atribuible a un usuario particular según su autenticación.
- Paralelo a la base de datos primaria se debe mantener una base de datos que registre todas las modificaciones hechas a la base de datos original, ordenadas cronológicamente y con la

especificación del usuario responsable de dicha modificación, de manera que siempre se realicen trazas a la información manejada.

- Se debe garantizar comunicaciones seguras entre los clientes y el servidor, encriptado todo el tráfico de información con la utilización de llaves negociadas, algoritmos y protocolos.

Eficiencia

Descripción del requisito

Se garantizará el funcionamiento de la aplicación durante las 24 horas del día y los siete días de la semana con el menor tiempo posible de recuperación de fallos.

El servidor de aplicación debe soportar un aumento de usuarios concurrentes por minuto de 1 a 400.

Tener una maquina con capacidad de almacenamiento suficiente ara soportar la información de 15 años de Ensayos Clínicos.

Soporte

Descripción de requisito

Requisitos de Software

Para la instalación de la aplicación se debe disponer del sistema operativo Windows o GNU Linux.

En las computadoras de los clientes también deberán existir las mismas restricciones de los Sistemas Operativos incluyendo un navegador asociado al sistema operativo finalmente escogido para la visualización de las interfaces Web.

Descripción de requisito

Requisitos de Hardware

Para el funcionamiento de la aplicación son imprescindibles un navegador y conectividad.

El servidor web debe tener alta disponibilidad y un rendimiento adecuado, garantizado por al menos un procesador Dual Intel Xeon 3 GHz o similar y RAM suficiente (4 GB a 8 GB).

Los servidores de almacenamiento de datos deben tener de 1 a 3 TB disponibles pues el volumen de información es bastante grande y perdura en el tiempo hasta 15 años.

Restricciones de diseño

El análisis y diseño de la aplicación será basado en la Metodología RUP con el uso del lenguaje de modelado UML.

Se usará como herramienta CASE Visual Paradigm para el modelado de los artefactos que se generan en cada uno de los flujos de trabajo definidos por RUP.

Para el diseño de las interfaces se utilizará Komposer.

Se usará como lenguaje de programación PHP.

Se usará como Gestor de Base de Datos Postgre-SQL.

Podrán ser utilizados varios estándares como HTTP, HTML, XML, SOAP, UDDI.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

Componentes Comprados

En el proyecto no se ha comprado ningún componente durante la duración del mismo.

Interfaz

Interfaces Hardware

El proyecto no cuenta con interfaces de hardware

Interfaces Software

No se utilizan interfaces de software

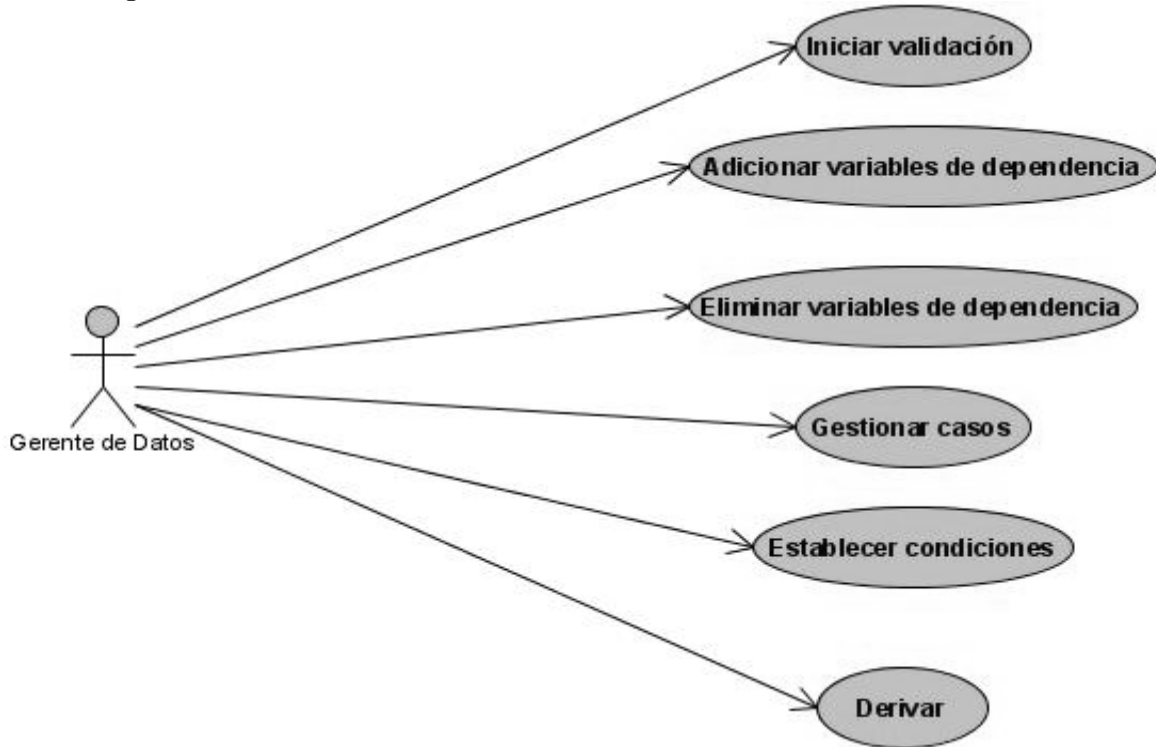
Interfaces de Comunicación

No se utilizan interfaces de software

Requisitos de Licencia

Durante el desarrollo del proyecto se han utilizado herramientas open source con licencias GPL

1. Diagrama de casos de uso del sistema



2. Descripción de los casos de uso del sistema

Caso de uso: Iniciar validación

Nombre del CU	Iniciar validación
Actores	Gerente de datos
Propósito	Escoger una variable para realizar una validación.
Resumen	El caso de uso inicia cuando el gerente de datos decide escoger una variable para realizar una validación. Busca dentro del modelo y del submodelo la subvariable que desea.
Referencias	
Precondiciones	Debe estar definido el cronograma de ejecución
Poscondiciones	Queda seleccionada la variable a la cual se le realizará la validación.
Curso Normal de los Eventos	
Acciones del Actor	Respuesta del Sistema
1. El gerente de datos entra al sistema.	2. El sistema muestra los modelos que contiene el ensayo en elaboración.
3. El gerente de datos selecciona un modelo	4. El sistema muestra los submodelos que contiene el modelo.

5. El gerente de datos selecciona un submodelo	6. El sistema muestra las variables del submodelo con sus respectivos componentes y al lado de cada variable, un ícono indicando si esta ha sido o no validada y finaliza el caso de uso.
Prioridad:	Crítico

Caso de uso: Eliminar variables de dependencia

Nombre del CU	Eliminar variables de dependencia	
Actores	Gerente de datos	
Propósito	Eliminar variables de dependencia	
Resumen	El caso de uso inicia cuando el gerente de datos decide eliminar variables de dependencia, selecciona las variables a eliminar y el sistema las elimina.	
Referencias		
Precondiciones		
Poscondiciones	Quedan eliminadas las variables de dependencia.	
Curso Normal de los Eventos		
Acciones del Actor	Respuesta del Sistema	
1. El gerente de datos decide eliminar variables de dependencia seleccionando la variable de dependencia que desea eliminar.	2. El sistema elimina de la validación la variable seleccionada y finaliza el caso de uso.	
Prioridad:	Crítico	

Caso de uso: Derivar

Nombre del CU	Derivar	
Actores	Gerente de datos	
Propósito	Permitir establecer las reglas de derivación para una variable en un caso	
Resumen	El caso de uso se inicia cuando el Gerente de datos decide establecer las reglas de derivación de la variable en un caso determinado. El sistema muestra las reglas de derivación de acuerdo al tipo de la variable y a la dependencia de esta, y el Gerente selecciona una regla.	
Referencias		
Precondiciones		
Poscondiciones	Quedan determinadas las reglas de derivación de una variable en un caso	
Curso Normal de los Eventos		
Acciones del Actor	Respuesta del Sistema	

<p>1. El Gerente de datos selecciona (en la tabla de estado) la opción de establecer las reglas de derivación de una variable en un caso determinado.</p> <p>3. El usuario selecciona una regla para la variable a derivar.</p>	<p>2 El sistema muestra todas las reglas de derivación según el tipo de la variable y su dependencia con otras variables.</p> <p>4. El sistema guarda la regla especificada.</p> <p>5. Actualiza la tabla de estado especificando en la fila del caso que se está analizando, en la casilla Derivar, la regla de derivación escogida.</p> <p>6. Si el usuario seleccionó la regla Asignada, el sistema habilita (en la tabla de estado) la opción de establecer las reglas de validación de la variable para el caso que se está analizando. Finaliza el caso de uso.</p>
Prioridad:	Crítico

Caso de uso: Adicionar variables de dependencia

Nombre del CU	Adicionar variables de dependencia.	
Actores	Gerente de datos	
Propósito	Escoger las variables (variables de dependencia) de las cuales depende la subvariable a la cual se establecerán las reglas de validación y de derivación.	
Resumen	El gerente de datos decide adicionar variables de dependencia. El sistema permite la búsqueda de las variables y el gerente escoge finalmente dichas variables.	
Referencias		
Pre condiciones		
Poscondiciones	Quedan definidas las variables de dependencia para una subvariable determinada.	
Curso Normal de los Eventos		
Acciones del Actor	Respuesta del Sistema	
1. El gerente de datos decide adicionar variables de dependencia.	2. Si es la primera vez que se van a adicionar variables de dependencia, el sistema pregunta la cantidad de variables de las cuales depende la subvariable a la cual se establecerán las reglas.	
2. El gerente entra la cantidad de variables	3. El sistema muestra la opción de buscar una variable de dependencia mostrando el listado de modelos.	

<p>4. El gerente selecciona el modelo al que pertenece una de las variables de dependencia</p> <p>6. El gerente escoge un submodelo</p> <p>8. El gerente escoge una variable de dependencia en el submodelo.</p> <p>9. El gerente escoge la subvariable y agrega la variable de dependencia.</p>	5. El sistema muestra el listado de submodelos de ese modelo.
	7. El sistema muestra el listado de variables de ese submodelo.
	9. El sistema muestra las subvariables de la variable seleccionada.
	10. El sistema vuelve a mostrar la opción de buscar otra variable (acción 2.1 – acción 6) hasta haber buscado la cantidad de variables de dependencia entradas por el gerente de datos y finaliza el caso de uso.
Prioridad:	Crítico

Caso de uso: Gestionar casos

Nombre del CU	Gestionar casos(Establecer y editar casos)
Actores	Gerente de datos
Propósito	Establecer el número de casos de validación para la subvariable
Resumen	El caso de uso inicia cuando el Gerente de datos decide establecer el número de casos de validación de la subvariable, el sistema permite la entrada del número de casos y muestra una tabla con las opciones de establecer las condiciones para cada caso, validar y derivar la subvariable en cada uno de ellos, adicionar o eliminar casos.
Referencias	
Precondiciones	
Poscondiciones	Queda establecido el número de casos de validación y eliminado o adicionado un caso.
Curso Normal de los Eventos	
Acciones del Actor	Respuesta del Sistema
<p>1. El Gerente de datos decide establecer el número de casos que va a tener la validación de una variable de acuerdo a las variables de dependencia establecidas.</p> <p>3. El gerente entra el número de casos.</p>	<p>2. Si la cantidad de variables de dependencia es mayor que cero, el sistema da la posibilidad al gerente de entrar el número de casos.</p> <p>4. El sistema muestra una tabla de estado dando la opción, en cada caso, de establecer las condiciones para que el caso se cumpla y las opciones de validar y derivar la variable según las condiciones de cada caso. muestra la opción de adicionar o eliminar casos.</p>

5. Una vez que se han establecido los casos para establecer las reglas de validación y de derivación de una variable, el Gerente de datos puede eliminar un caso establecido o adicionar uno nuevo.	6. El sistema ejecuta una de las siguientes acciones: a) Si el gerente decide eliminar un caso, ir a la sección “Eliminar caso”. b) Si el gerente decide adicionar un nuevo caso, ir a la sección “Adicionar caso”.
---	---

Sección “Eliminar caso”

Curso Normal de los Eventos

Acciones del Actor	Respuesta del Sistema
1. El Gerente de datos selecciona el o los casos que desea eliminar.	2. El sistema los elimina y finaliza el caso de uso.

Sección “Adicionar caso”

Curso Normal de los Eventos

Acciones del Actor	Respuesta del Sistema
1. El Gerente de datos decide adicionar un nuevo caso.	2. El sistema adiciona una nueva fila con un nuevo caso en la tabla de estado y las opciones de establecer condiciones, validar y derivar para este caso. Y finaliza el caso de uso.

Curso Alternativo de los Eventos

Acciones del Actor	Respuesta del Sistema
	2.1. Si la cantidad de variables de dependencia es cero, el sistema muestra la tabla de estado con un solo caso y las opciones de validar y derivar la variable. Y finaliza el caso de uso.

Prioridad: | Crítico

Caso de uso: Establecer condiciones

Nombre del CU	Establecer condiciones
Actores	Gerente de datos
Propósito	Establecer los valores que puede tomar cada variable de dependencia para que se cumpla un caso de validación determinado
Resumen	El caso de uso inicia cuando el gerente decide establecer los valores que debe tomar cada variable de dependencia en un caso determinado. El sistema da la posibilidad de establecer estos valores y una vez determinados por el gerente, los muestra en la tabla de estado en la casilla correspondiente a las condiciones del caso que se está validando.
Referencias	
Precondiciones	Debe haber al menos una variable de dependencia para la variable que se valida.

Poscondiciones	Quedan establecidas las condiciones para que se cumpla un caso determinado.
Curso Normal de los Eventos	
Acciones del Actor	Respuesta del Sistema
<p>1. El Gerente de datos decide establecer las reglas de validación y de derivación de un caso, para ello necesita establecer las condiciones para validar y derivar una variable de acuerdo a los valores tomados por las variables de dependencia en cada caso. Selecciona en la tabla de estado las condiciones para un caso determinado.</p> <p>5. El gerente especifica los valores que puede tomar la variable de dependencia o los valores que no puede tomar para que se cumpla el caso que se está validando.</p>	<p>2. El sistema muestra las variables de dependencia especificadas para la subvariable a la cual se le están estableciendo las reglas.</p> <p>3. El sistema muestra para cada variable de dependencia la opción de especificar si esta debe ser asignada o no y en caso de ser asignada, la opción de establecer los valores, o el rango de valores, que cada una debe tomar.</p> <p>4. Para las variables de tipo nomenclador, el sistema muestra el listado de valores que estas pueden o no tomar.</p> <p>6. El sistema muestra en la tabla de estado, en la casilla de las condiciones del caso que se está validando, los valores especificados que puede o no tomar cada variable de dependencia y finaliza el caso de uso.</p>
Curso Alternativo de los Eventos	
Acciones del Actor	<p>Respuesta del Sistema</p> <p>4.1. Para las variables de dependencia que no son de tipo nomenclador, el sistema da la posibilidad de establecer los rangos de valores en los que deben estar cada variable para que se cumpla el caso que está siendo validado (para establecer un rango se usarán los signos >, <, >=, <=, <>, = , acompañado de los signos , &, para establecer más de un rango de valores y se dará la opción de entrar los valores) (Ir a la acción 2 del flujo normal de eventos.</p>
Prioridad:	Crítico

Glosario de términos

C

CASE: Computer Aided Software Engineering

CIM: Centro de Inmunología Molecular

CRD: Cuadernos de Recogida de Datos

D

Derivación: Dependencia entre los valores de las variables del CRD, permitiendo derivar un dato a partir de otros.

E

EC: Ensayos Clínicos

I

IDE: Entorno de Desarrollo Integrado

M

Modelo: Conjuntos de variables que pertenecen a un determinado examen o evaluación.

MVC: Patrón arquitectónico Modelo Vista Controlador.

R

RUP: Rational Unified Process (Proceso Unificado de Desarrollo).

S

SIMDECC: Sistema de Manejo de Datos de Ensayos Clínicos de Cuba.

Submodelo: Es un subconjunto del modelo.

SVN: Subversion, herramienta para el control de versiones.

V

Validación: Se refiere a establecer reglas que definan el valor de los datos en los CRD cuando estos son entrados directamente por el usuario.

Variables: Todos los datos referentes a un pacientes y al estado de su enfermedad. Subconjunto de los

submodelos.

VP: Visual Paradigm

U

UML: Unified Modeling Language (Lenguaje Unificado de Modelado)

URL: Uniform Resource Locator (Localizador Uniforme de Recursos).