Universidad de las Ciencias Informáticas

Facultad 6

SIMDECC

Título: Sistema de Manejo de Datos de Ensayos Clínicos Cubano. Módulo Validación: Diseño del Submódulo “Validación de las variables del Cuaderno de Recogida de Datos y el control de errores“.

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

Autor(es):
Yailin Peña Cisnero.
Ana María Gómez Avila.

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

Ciudad de la Habana, Junio 2008
“Año 50 de la Revolución”

“Después de escalar una montaña muy alta, descubrimos que hay muchas otras montañas por escalar."

Nelson Mándela.

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 __2008____.

Yailin Peña Cisnero.

_____

Firma del Autor

Ana María Gómez Avila.

_____

Firma del Autor

Ing. Aislein González Blanco.

_____

Firma del Tutor

Ing. Lucía Rodríguez García.

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Aislein Blanco González

La compañera Aislein Blanco González, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Módulo Validación: Diseño del Submódulo “Validación de las variables del Cuaderno de Recogida de Datos y el control de errores”, fue graduada de la Universidad de las Ciencias Informáticas con el título de Ingeniera en Ciencias Informáticas en el año 2007, la misma se graduó con un promedio de 4.58 puntos, y en la exposición de su trabajo de diploma con bonificación de 5 puntos, actualmente se desempeña como profesora de BD en la universidad donde se graduó y como diseñadora de BD en el proyecto Ensayos Clínicos al cual pertenece.

Ing. Lucía Rodríguez García

La compañera Lucía Rodríguez García, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Módulo Validación Diseño del Submódulo “Validación de las variables del Cuaderno de Recogida de Datos y el control de errores”, fue graduada de la Universidad de las Ciencias Informáticas con el título de Ingeniera en Ciencias Informáticas en el año 2007, la misma se graduó con honores de título de oro, más de 4.75 puntos en su carrera y por ende obteniendo como calificativo la bonificación de 5 puntos en la exposición de su trabajo de diploma. Actualmente se desempeña como profesora de Matemática en la universidad donde se graduó, como Analista Principal y Jefa del módulo Validación en el proyecto Ensayos Clínicos al cual pertenece.

DEDICATORIA

A mis padres Margarita y Rodolfo por ser la luz que me guía por el sendero de la mi vida. Por apoyarme y ayudarme tanto en mis estudios como en mi vida personal, por estar siempre presente cuando más los necesité. Por haberme entregado el tesoro más valioso que puede dársele a un hijo: "amor". Los que sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y educarme. Los cuales siempre han tenido la ilusión de convertirme en una persona de provecho. A quienes nunca podré pagar todos sus desvelos ni aún con las riquezas más grandes del mundo. A ustedes les debo cuanto soy y seré.

A Yuleidis por ser más que mi hermana, por apoyarme tanto y estar tan pendiente de mi., por darme fuerzas para levantarme y seguir adelante enfrentándome a la vida.

A todos mis familiares y amigos por su cariño y apoyo incondicional.

Yailin Peña Cisnero.

A mis maravillosos padres Tanis y Jorge por ser las personas más importantes de mi vida, mi fuente de inspiración, mi guía y ejemplo, por permitirme llegar hasta aquí y convertirme en lo que soy. Gracias por confiar en mí, por brindarme su amor, apoyo y comprensión, por estar siempre presente en los momentos más difíciles, por darme aliento para salir adelante, por llenar mi vida de felicidad.

A mi querida abuela por ser tan especial para mí, por su amor y apoyo incondicional, por cuidarme, comprenderme y hacerme tan feliz.

A mi hermanito del alma por su apoyo y preocupación, he tratado de ser más que una hermana para ti.

A mi tía Yami por sus consejos, su cariño y por estar siempre tan pendiente de mí.

A Duniel por ser tan especial en mi vida, por su amor, apoyo y comprensión, por confiar en mí y estar siempre presente cuando más lo necesito, por hacerme tan feliz.

A todos mis familiares por su apoyo y preocupación.

A mis amigas Yinet, Marta, Analia, Leidy y Nelvis por brindarme su amistad, por sus consejos y su apoyo en los momentos difíciles, gracias por dejarme entrar en sus corazones.

Ana María Gómez Avila.

AGRADECIMIENTOS

Al terminar esta etapa de la vida, queremos expresar un profundo agradecimiento a todas las personas que con su ayuda, apoyo y comprensión nos alentaron para lograr este hermoso sueño que hoy se convierte en realidad.

A nuestros padres de forma muy especial, pues son el motivo de inspiración de nuestras vidas con sus consejos, su apoyo, dedicación, por estar siempre presente cuando más lo necesitamos, gracias por su confianza.

A nuestros buenos amigos, los nuevos y los viejos, los de aquí y los de allá, gracias por existir.

A todos aquellos que nos escucharon pacientemente cuando lo necesitamos, y a los que no, también le agradecemos por mostrarnos que el camino está lleno de espinas.

A la revolución cubana y a nuestro eterno Comandante en jefe Fidel Castro Ruz por darnos la oportunidad de estudiar en esta universidad, y formarnos como jóvenes revolucionarios y de buenas convicciones.

A nuestras tutoras por su paciencia y apoyo.

A todos los estudiantes y profesores del proyecto por estar siempre dispuestos a ayudarnos.

A nuestros compañeros de grupo, de apartamento, de fiestas y de estudios gracias por los momentos tan lindos que pasamos juntos.

Ani y Yaily

RESUMEN

La presente investigación surge con el desarrollo del proyecto Ensayos Clínicos perteneciente al Centro de Inmunología Molecular a raíz de la colaboración entre el Polo Científico y la Universidad de las Ciencias Informáticas.

El Centro de Inmunología Molecular ha desarrollado una serie de biomoléculas para el tratamiento de diferentes enfermedades, principalmente el cáncer. En la medida que se avanza en el desarrollo de estos productos, se avanza en las fases de los Ensayos Clínicos, los mismos son estudios donde se evalúan nuevos tratamientos médicos, que a través de su aplicación a seres humanos se pretende valorar su eficacia y seguridad, los Ensayos Clínicos tienen asociado un Cuaderno de Recogida de Datos en el que se recoge toda la información relacionada con el paciente.

El Cuaderno de Recogida de Datos tiene un gran volumen de información, lo que trae consigo que a la hora de entrar los datos ocurran múltiples errores. Para dar solución a esta problemática la presente investigación realizó el diseño del submódulo “Validación de las variables del Cuaderno de Recogida de Datos y el control de errores“ del módulo Validación para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano, tomando como punto de partida los resultados obtenidos en la investigación precedente.

PALABRAS CLAVE: Cáncer, Ensayos Clínicos, Cuaderno de Recogida de Datos.

ÍNDICE

DEDICATORIA ---------- I

AGRADECIMIENTOS ---------- 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 ¿QUÉ ES LA VALIDACIÓN DE LAS VARIABLES DEL CRD Y EL CONTROL DE ERRORES? ---------- 5
1.4 APLICACIONES INFORMÁTICAS RELACIONADAS CON EL CAMPO DE ACCIÓN ---------- 5
1.5 METODOLOGÍAS, HERRAMIENTAS Y TECNOLOGÍAS ---------- 7
1.6 CONCLUSIONES ---------- 18

CAPÍTULO 2: DISEÑO DEL SISTEMA ---------- 19

2.1 PATRONES DE DISEÑO ---------- 20
2.2 PATRONES DE ARQUITECTURA ---------- 24
2.3 MODELO DE DISEÑO ---------- 25
2.4 MAPA DE NAVEGACIÓN ---------- 55
2.5 DIAGRAMA DE CLASES PERSISTENTES ---------- 56
2.6 MODELO FÍSICO ---------- 57
2.7 DIAGRAMA DE DESPLIEGUE ---------- 58
2.8 VALIDACIÓN DEL DISEÑO ---------- 60
2.9 CONCLUSIONES ---------- 61

CONCLUSIONES GENERALES ---------- 62

RECOMENDACIONES ---------- 63

REFERENCIAS BIBLIOGRÁFICAS ---------- 64

BIBLIOGRAFÍA CONSULTADA ---------- **66**

ANEXOS ---------- **68**

GLOSARIO DE TÉRMINOS ---------- **78**

ÍNDICE DE FIGURAS

FIGURA 1 SOFTWARE OPENCLINICA. ---------- 6
FIGURA 2 RATIONAL UNIFIED PROCESS (RUP). ---------- 8
FIGURA 3 VISUAL PARADIGM PARA UML. ----------10
FIGURA 4 UNIFIED MODELING LANGUAGES (UML).----------11
FIGURA 5 SUBVERSION (SVN).----------12
FIGURA 6 KOMPOZER.----------13
FIGURA 7 FRAMEWORK SYMFONY. ----------15
FIGURA 8 PERSONAL HOME PAGE (PHP5).----------16
FIGURA 9 SERVIDOR WEB APACHE. ----------17
FIGURA 10 PATRÓN MODELO VISTA CONTROLADOR. ----------24
FIGURA 11 PAQUETES DEL DISEÑO. ----------26
FIGURA 12 DIAGRAMA DE CLASES DEL DISEÑO DEL CU VALIDAR.----------34
FIGURA 13 DIAGRAMA DE CLASES DEL DISEÑO DEL CU FINALIZAR VALIDACIÓN DE UNA SUBVARIABLE.----------35
FIGURA 14 DIAGRAMA DE CLASES DEL DISEÑO DEL CU FINALIZAR VALIDACIÓN DEL CRD. --------36
FIGURA 15 DIAGRAMA DE CLASES DEL DISEÑO DEL CU MOSTRAR VALIDACIÓN DE UNA SUBVARIABLE.----------37
FIGURA 16 DIAGRAMA DE CLASES DEL DISEÑO DEL CU ANALIZAR CONSISTENCIA DE LAS REGLAS ESPECIFICADAS. ----------38
FIGURA 17 DIAGRAMA DE CLASES DEL DISEÑO DEL CU EMITIR INFORME DE LA VALIDACIÓN DE UN SUBMODELO. ----------39
FIGURA 18 DIAGRAMA DE SECUENCIA PARA EL DISEÑO DEL CU FINALIZAR VALIDACIÓN DE UNA SUBVARIABLE.----------40
FIGURA 19 DIAGRAMA DE SECUENCIA PARA EL DISEÑO DEL CU VALIDAR. ----------41
FIGURA 20 DIAGRAMA DE SECUENCIA PARA EL DISEÑO DEL CU FINALIZAR VALIDACIÓN DEL CRD. ----------42
FIGURA 21 DIAGRAMA DE SECUENCIA PARA EL DISEÑO DEL CU MOSTRAR VALIDACIÓN DE UNA SUBVARIABLE.----------43
FIGURA 22 DIAGRAMA DE SECUENCIA PARA EL DISEÑO DEL CU ANALIZAR CONSISTENCIA DE LAS REGLAS ESPECIFICADAS.----------44

FIGURA 23 DIAGRAMA DE SECUENCIA PARA EL DISEÑO DEL CU EMITIR INFORME DE LA VALIDACIÓN DE UN SUBMODELO.----------45
FIGURA 24 PÁGINA CASOS. CU VALIDAR.----------46
FIGURA 25 CU VALIDAR. ----------47
FIGURA 26 CU ANALIZAR CONSISTENCIA DE LAS REGLAS ESPECIFICADAS Y CU FINALIZAR VALIDACIÓN DE UNA SUBVARIABLE.----------48
FIGURA 27 CU MOSTRAR VALIDACIÓN DE UNA SUBVARIABLE.----------49
FIGURA 28 CU MOSTRAR VALIDACIÓN DE UNA SUBVARIABLE.----------50
FIGURA 29 CU FINALIZAR VALIDACIÓN DEL CRD. ----------51
FIGURA 30 CU FINALIZAR VALIDACIÓN DEL CRD. ----------52
FIGURA 31 CU EMITIR INFORME DE LA VALIDACIÓN DE UN SUBMODELO.----------53
FIGURA 32 CU EMITIR INFORME DE LA VALIDACIÓN DE UN SUBMODELO.----------54
FIGURA 33 MAPA DE NAVEGACIÓN DEL MÓDULO VALIDACIÓN. ----------55
FIGURA 34 DIAGRAMA DE CLASES PERSISTENTES. ----------56
FIGURA 35 MODELO FÍSICO. ----------57
FIGURA 36 DIAGRAMA DE DESPLIEGUE.----------58

ÍNDICE DE TABLAS

TABLA 1 DESCRIPCIÓN DE LA CLASE DEL DISEÑO: FINALIZARVALSUBVARIABLEACTIONS. ---------27
TABLA 2 DESCRIPCIÓN DE LA CLASE DEL DISEÑO: VALIDARACTIONS. ----------28
TABLA 3 DESCRIPCIÓN DE LA CLASE DEL DISEÑO: FINALIZARVALIDACIONCRDACTIONS.----------29
TABLA 4 DESCRIPCIÓN DE LA CLASE DEL DISEÑO: MOSTRARVALSUBVARIABLEACTIONS. ---------30
TABLA 5 DESCRIPCIÓN DE LA CLASE DEL DISEÑO: EMITIRINFORMEVALIDACIONACTIONS. --------31
TABLA 6 DESCRIPCIÓN DE LA CLASE DEL DISEÑO: ANALIZARCONSISTENCIAREGLASACTIONS. -32

INTRODUCCIÓN

En Cuba se aúnan incalculables recursos materiales y humanos en aras de mantener los logros en la esfera de la salud y elevar la calidad de vida de su ciudadanía. Se trabaja incansablemente en la búsqueda e implementación de vías y alternativas que contribuyan al diagnóstico y tratamiento de forma eficaz y rápida de las enfermedades que tienen un denominador común: la transformación de la célula normal en otra que se comporta de forma muy peligrosa para el cuerpo humano "Cáncer". Quizás esta sea una de las palabras más utilizadas y que más asusta cuando se habla de salud y de su reverso.

Con el objetivo de lograr la aprobación de tratamientos para esta y otras enfermedades que ponen en riesgo la existencia del hombre se llevan a cabo una serie de investigaciones y métodos denominados Ensayos Clínicos (EC).

Los EC son estudios de investigación que prueban el funcionamiento de los nuevos enfoques clínicos en las personas. Cada estudio responde preguntas científicas e intenta encontrar mejores formas de prevenir, explorar, diagnosticar o tratar una enfermedad. Con los EC también se pueden comparar tratamientos nuevos con otros que ya se encuentran disponibles.

Las Agencias Regulatorias (AR) se encargan de aprobar los nuevos productos o medicamentos, asegurando que los productos para la salud tengan calidad, seguridad y eficacia, además exigen que los mismos sean producidos y distribuidos de forma tal que mantengan su calidad hasta llegar al paciente que los consume, es por eso que los EC tienen que ser lo más exactos posibles para lograr eficiencia en la obtención de los resultados.

Toda la documentación perteneciente a un EC se recoge a través de los Cuaderno de Recogida Datos (CRD), los cuales tienen un gran volumen de información asociada, como esta se obtiene de forma manual pueden ocurrir muchos errores y pérdida de la misma, por lo que se hace necesario validar toda la información que se encuentra en los CRD y controlar cada uno de los posibles errores, para obtener un mejor resultado en los EC.

Para realizar el proceso de obtención y almacenamiento de la información de forma eficiente, la Universidad de las Ciencias Informáticas conjuntamente con el Centro de Inmunología Molecular (CIM)

se ha propuesto realizar un sistema que permita manipular toda la información de forma automatizada, para ello un grupo de investigadores se dieron la tarea de realizar el análisis de la problemática, teniendo como resultado el modelamiento del negocio, el levantamiento de los requerimientos funcionales y no funcionales y el análisis. Por todo lo antes expuesto se plantea como ***Problema científico*** de la investigación ***¿Cómo traducir los requerimientos identificados en el submódulo "Validación de las variables del Cuaderno de Recogida Datos y el control de errores" del módulo Validación en elementos que puedan ser implementados?***

Con el objetivo de solucionar el problema anterior se enmarca el ***Objeto de estudio*** en: ***Los procesos de*** **gestión de la información de aplicaciones Web para Ensayos Clínicos**.

A partir del objeto de estudio se delimita el Campo de acción en: **Los procesos de gestión de la información de aplicaciones Web para la validación de las variables del Cuaderno de Recogida Datos y el control de errores de los Ensayos Clínicos**.

Para dar solución al problema planteado se define como Objetivo general de la investigación: **Diseñar el submódulo "Validación de las variables del Cuaderno de Recogida de Datos y el control de errores" del módulo Validación para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano en el Centro de Inmunología Molecular**.

Para dar cumplimiento a los objetivos planteados se anuncian las siguientes ***Tareas****:*

- **Estudio de los resultados de la investigación precedente relacionados con la validación de las variables del Cuaderno de Recogida Datos.**
- **Estudio y selección de las herramientas a utilizar en el diseño del módulo Validación.**
- **Estudio y selección del framework a emplear.**
- **Estudio de los patrones de diseño a utilizar.**
- **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.**

El documento está formado por resumen, introducción, dos capítulos que constituyen el cuerpo fundamental de este trabajo, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos. Los capítulos son:

Capítulo 1: Fundamentación teórica. En este capítulo se aborda el estado del arte de los procesos de diseño de aplicaciones Web para EC, siendo este el objeto de estudio de la investigación, se fundamentan las metodologías, tecnologías y herramientas a utilizar para dar solución a la problemática existente.

Capítulo 2: Diseño del sistema. En el presente capítulo se realiza el diseño del submódulo "Validación de las variables del CRD y el control de errores", con el objetivo de obtener un diseño que sirva como entrada al modelo de implementación. Además se fundamentan los patrones de diseño y arquitectura que se utilizarán para diseñar el submódulo. Con el diseño se logra un refinamiento del análisis teniendo en cuenta los requisitos no funcionales. Este capítulo es muy importante, porque de él depende que la aplicación se implemente sin imprecisiones.

Capítulo 1: Fundamentación Teórica

En este capítulo se aborda el estado del arte de los procesos de diseño de aplicaciones Web para EC, siendo este el objeto de estudio de la investigación, se fundamentan las metodologías, tecnologías y herramientas a utilizar para dar solución a la problemática existente.

1.1 ¿Qué es un ensayo clínico?

Los ensayos clínicos son estudios de investigación que prueban el funcionamiento de los nuevos tratamientos médicos aplicándolos a seres humanos. Cada estudio intenta encontrar mejores formas de prevenir y diagnosticar una enfermedad.

Cada ensayo clínico tiene un protocolo o plan de acción para llevarlo a cabo. El plan describe lo que se hará en el estudio, cómo se hará y por qué cada parte del estudio es necesaria. Cada estudio tiene sus propias reglas acerca de quién puede participar. Algunos necesitan voluntarios con una determinada enfermedad, otros necesitan personas sanas y otros solamente solicitan hombres o mujeres. [1]

1.2 ¿Qué es un CRD?

Cuando se realiza un ensayo clínico se genera un gran volumen de información referente al paciente, la enfermedad que padece y los exámenes médicos que se le aplican. Todos estos datos son registrados en un documento que está compuesto por modelos, submodelos, variables y subvariables. A este documento se le denomina Cuaderno de Recogida de Datos.

El cuaderno de recogida de datos (CRD) es un formulario que será incluido en la documentación sometida a aprobación por los comités éticos institucionales y las autoridades sanitarias. Debe ser reflejo de la documentación clínica original (historia clínica, informes de laboratorio y exploraciones complementarias), individual para cada paciente, debe facilitar la reproducción de los datos generados, su tratamiento estadístico, la preparación y corrección del informe final de un ensayo clínico y debe ser la sustentación científica de un nuevo medicamento o producto sanitario. [2]

1.3 ¿Qué es la validación de las variables del CRD y el control de errores?

Cada variable del CRD está compuesta por subvariables, cuando la variable no tiene subvariables ella asume el papel de subvariable.

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 posibles.

Al entrar los datos de cada una de las variables de un CRD, se corre el riesgo de cometer serios errores que puedan afectar a gran escala el resultado de la investigación médica, dados fundamentalmente por la falta de validación de las variables. Por tanto el control de errores consiste en controlar los errores después de determinar para cada una de las variables los rangos dentro de los que puede oscilar o los valores que puede o no tomar de acuerdo a su tipo de dato, e incluso las dependencias que existan entre ellas. Para ello se crea una serie de reglas de validación que estandarizarán la entrada de los datos de forma correcta.

1.4 Aplicaciones informáticas relacionadas con el campo de acción

Actualmente no existe una herramienta que les facilite a los usuarios validar las variables de los CRD utilizando reglas de validación, pues los sistemas que trabajan los Ensayos Clínicos realizan una validación sencilla, fundamentalmente utilizando JavaScript, además no cuentan con reglas de validación que permitan tratar los datos del CRD de forma más segura para almacenarlos correctamente, algunos de los software que cumplen con estas características son OpenClinica, MACRO y PhOSCo.

OpenClinica

OpenClinica es un software de código abierto para la recogida de datos electrónicos que se utiliza en investigaciones clínicas. Brinda una interfaz fácil de usar para la presentación y validación de los datos facilitando su uso a los médicos y los investigadores que participan en todo el proceso de atención al paciente. Además brinda la facilidad de crear el diseño de cuadernos para la captura de datos sin necesitar conocimientos de programación.

El proyecto esta basado en Java, recientemente liberado como software libre. La versión actual de OpenClinica funciona en Linux 3.0, Apache Jakarta Tomcat 5.0 y PostgreSQL v8.0. La arquitectura transparente del producto, y el modelo de desarrollo colaborativo ofrecen grandes flexibilidades. [3]

OpenClinica no cuenta con reglas específicas de validación para cada una de las variables asociadas a los Cuadernos de Recogida de Datos, lo cual no satisface a la necesidad del cliente de desarrollar una herramienta para obtener un conjunto de reglas previamente definidas con vista a registrar los datos en el CRD sin errores.

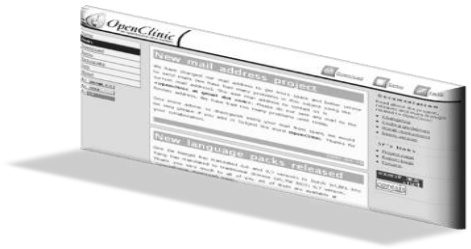

Figura 1 Software OpenClinica.

MACRO

Macro, es una de las soluciones y servicios que ofrece InferMed para el manejo y colección de datos que se diseñan para mejorar los resultados y sacar fármacos utilizados en los ensayos clínicos.

Esta herramienta permite el diseño del Cuaderno de Recogida de Datos (CRD) de acuerdo al estudio clínico que vaya a realizarse y la validación de los datos que estarán incluidos en dicho estudio. Es uno de los sistemas más completos para los procesos de diseño, pero no permite la estandarización de los CRD, cualquier usuario puede diseñar un CRD según criterios propios, nombrando las variables que recogerá como desee sin tener una plantilla como guía.

Para su adquisición es necesario un contrato con InferMed, pues es muy costoso poseer las licencias de configuración del servidor y el uso de cada uno de los módulos de Macro. [4]

PhOSCo

Es un proyecto para dar apoyo a las funciones clínicas de los ensayos de una de las más grandes industrias dedicadas a la conducción de Ensayos Clínicos, consumiendo aproximadamente el 10 por ciento del producto interno bruto de la mayoría de las naciones desarrolladas. [5]

Este software permite el diseño de los CRD mediante la creación de las páginas para cada ensayo y establece el orden en que deben aparecer cada una de ellas, así como la confección de todas las preguntas que deben estar incluidas de acuerdo a cada ensayo.

Permite establecer chequeos de validación a los datos entrados una vez creado el cuaderno de recogida de datos, que serán hechos en pedazos individuales de los datos que el usuario del registrador de ensayo recogerá e incorporará en el uso, lo cual crea el inconveniente de que cada diseñador cree su propio ensayo a su estilo, pudiendo estar creando el mismo ensayo en cuanto a términos de modelos, submodelos y variables pero con denominaciones diferentes cosa que rompe con el propósito de estandarización del Polo. Brinda la posibilidad de estudiar, señalar y supervisar el estado de los datos clínicos recogidos en el cuaderno. [5]

No permite la estandarización de los CRD y es un software propietario que tiene un alto precio y dada la situación económica que presenta Cuba actualmente, no puede sufragar estos gastos.

Después de haber analizado estos software, se arribó a que no constituyen la solución óptima para el proyecto, pues no cumplen con los requisitos que exige el polo científico, ya que presentan grandes limitaciones en el manejo de datos, no definen reglas de validación que eviten la entrada incorrecta de los datos en los CRD, la validación que se realiza es muy sencilla. Por todo esto se ratifica la necesidad de desarrollar un sistema para el país que sea libre, el cual garantice que la información de los CRD sea la más exacta y fiable posible, para ello se proponen metodologías, herramientas y tecnologías que permitan obtener el producto deseado.

1.5 Metodologías, Herramientas y Tecnologías

En este epígrafe se argumentarán las metodologías, herramientas y tecnologías que se utilizarán para el diseño del submódulo. Las mismas fueron definidas por el arquitecto y los miembros del proyecto, teniendo en cuenta la propuesta realizada por la investigación precedente y la necesidad de migrar a software libre, ya que el cliente desea un producto multiplataforma y libre.

Metodología RUP

RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Define las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos de apoyo. El proceso unificado de desarrollo se divide en cuatro fases:

- 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 proyecto realizado. [6]

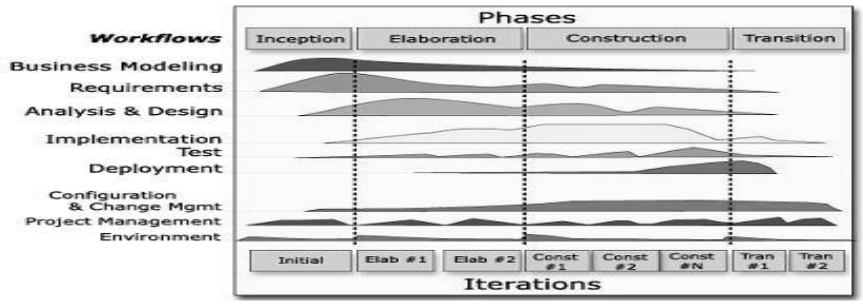

Figura 2 Rational Unified Process (RUP).

Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, entre otros.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la 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. Es importante mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas: Disciplina de Desarrollo y Disciplina de Soporte.

Disciplina de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado por el cliente está presente en el producto final.

Disciplina de Soporte

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto.
- Los elementos del RUP son:
- Actividades: Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: Son las personas o entes involucrados en cada proceso.
- Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo. [6]

Flujos de trabajo definidos:

RUP propone nueve flujos de trabajo y para cada uno de ellos se definen roles y artefactos. La investigación se centrará en los siguientes:

Análisis y Diseño:

Roles:

- Diseñador.
- Diseñador de interfaz de usuario.
- Diseñador de Base de Datos.
- Arquitecto.

Artefactos:

- Modelo del diseño.

- Clases del diseño.
- Realización de los CU.
- Paquetes y Subsistemas.
- Diagramas de despliegue.
- Prototipos de interfaz de usuario.
- Mapa de navegación.
- Modelo físico.

Herramientas CASE Visual Paradigm 3.1

Es una potente herramienta para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permita diseñar un producto con calidad de forma rápida y permita su uso en cualquier sistema operativo. Está orientada a la creación de diseños usando el paradigma de programación orientada a objetos (POO).

Facilita la interoperabilidad con otras herramientas CASE y se integra con las siguientes herramientas: Eclipse, WebSphere, JBuilder, NetBeans, Oracle JDeveloper, BEA Weblogic. Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. [7]

Figura 3 Visual Paradigm para UML.

Es muy importante para la investigación el uso de la herramienta CASE Visual Paradigm por las ventajas que proporciona, así como la integración que posee con otras herramientas, lo que facilita el trabajo de los diseñadores y programadores, pues integrar la herramienta de modelado con el IDE de desarrollo conduce a un aumento de la productividad y agiliza todo el desarrollo del proyecto.

Lenguaje Unificado de Modelado 2.0 (UML)

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un 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. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Rational Unified Process), pero no especifica en sí mismo qué metodología o proceso usar. UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. [8]

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

Figura 4 Unified Modeling Languages (UML).

Herramientas para Control de Versiones Subversion 4.2.0

Control de versiones no es más que los métodos y herramientas disponibles para controlar todo lo referente a los cambios de un archivo en el tiempo. Es muy importante utilizar un sistema de control de versiones, ya que el volumen de documentación que genera el modelado del presente sistema informático es muy amplio.

Subversion es un software de control de versiones diseñado específicamente para reemplazar al popular Sistema de control de versiones (CVS). Este software presenta una fuerte integración con Apache, lo cual permite definir controles de acceso avanzados y navegación vía Web para consultar el depósito de archivos.

Posee una gran transparencia al eliminar y cambiar nombres de archivos. Es independientemente del número de ramificaciones creadas mantiene un árbol diferencial de cambios, que minimiza el espacio consumido en el repositorio.

Realiza 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 posibilitando reducir el consumo de espacio.

Una característica importante de Subversion es que los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Ventajas:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; tiene costo de complejidad constante (O (1)) y no lineal (O (n)).
- Se envían sólo las diferencias en ambas direcciones.
- Puede ser servido mediante Apache. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios.
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.). [9]

Figura 5 Subversion (SVN).

Herramienta para el diseño de interfaces Kompozer

Kompozer es una herramienta visual, libre y multiplataforma, basada en Nvu, para el diseño de Web con soporte XHTML, HTML y CSS. Incorpora código HTML eficiente, que lo hace capaz de trabajar con los más populares buscadores de hoy. Tiene un potente soporte para formularios, tablas y templates. La interfaz es distinta pero los comandos son muy similares a Dreamweaver. [10]

Permite crear páginas Web de forma sencilla a través de una interfaz gráfica. Posee una extensión que añade soporte para ASP/JSP/PHP. Debido a su fácil uso, no es necesario tener conocimientos especiales de HTML para crear un sitio atractivo.

Es un software gratuito y tiene muchas características parecidas a FrontPage e incluso al Dreamweaver. Permite tanto la edición gráfica como de código, viendo en todo momento el resultado en la misma interfaz del programa, ideal para los usuarios novatos.

Figura 6 Kompozer.

Framework Symfony 1.0.10

Los framework son diseñados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura a sus proyectos, permitiéndoles pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Además ayudan en el desarrollo de software, proporcionan una estructura definida la cual ayuda a crear aplicaciones con mayor rapidez. Ayuda a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la aplicación. [11]

Es muy importante para la investigación el uso del framework Symfony 1.0.10 pues es diseñado con el objetivo de optimizar la creación de las aplicaciones Web, con el uso de sus características. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Posee una librería de

clases que permiten reducir el tiempo de desarrollo. Se ha probado en numerosos e importanes proyectos reales por ejemplo Yahoo.

Creado con PHP 5, se puede utilizar en plataformas *nix (Unix, Linux) y Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón MVC, soporta AJAX, plantillas y un gran número de bases de datos. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Añade una nueva capa por encima de PHP y proporciona herramientas que simplifican el desarrollo de las aplicaciones Web complejas.

El mismo está basado en el patrón de arquitectura Modelo-Vista-Controlador, que está formado por tres niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir su lógica de negocio.
- La vista transforma el modelo en una página Web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Además utiliza el patrón de diseño Decorator usado fundamentalmente para agregar funcionalidad de adorno a un objeto individual de forma dinámica y transparente.

Este framework organiza el código fuente en una estructura de tipo proyecto y almacena los archivos del proyecto en una estructura estandarizada de tipo árbol. Considera un proyecto como “un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos”. [11]

Los archivos de configuración pueden modificar por completo el funcionamiento del framework. Como Symfony utiliza la configuración incluso para sus características internas y para la carga de los archivos, se puede adaptar fácilmente a muchos entornos.

Esta gran “configurabilidad” es uno de los puntos fuertes de Symfony. Aunque a veces echa para atrás a los programadores que están empezando a trabajar con el framework, porque son muchos archivos de configuración y hay que aprender muchas convenciones, lo cierto es que permite que las aplicaciones Symfony sean compatibles con un gran número de sistemas y entornos diferentes. Una

vez que se dominan los archivos de configuración de Symfony, se pueden ejecutar las aplicaciones en cualquier servidor del mundo. [11]

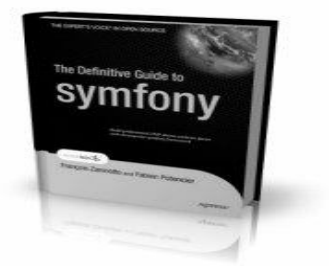

Figura 7 Framework Symfony.

Además incluye constantemente mejoras de forma consistente combinando la flexibilidad con la rapidez de ejecución. Tiene un libro que explica detalladamente todos sus conceptos y posee gran documentación. Permite reutilización de componentes, librerías externas buenas y muy probadas, en vez de intentar reinventar la rueda constantemente. Es un framework que usa herramientas potentes como 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. La comunidad de usuarios crece cada día y ofrece un gran soporte de forma gratuita. Su gran flexibilidad tanto en el diseño global del framework como en su sistema de configuración y en los plugins es muy buena. Además Symfony usa lo mejor de otros framework e incluso con mejor desarrollo y soporte.

Tecnología de desarrollo PHP 5.2.0 (Personal Home Page)

PHP es un lenguaje de programación usado normalmente para la creación de páginas Web dinámicas, para programar scripts del lado del servidor, que se incrustan dentro del código HTML.

Es un lenguaje gratuito e independiente de plataforma, rápido, con una gran librería de funciones y una amplia gama de documentación.

Puede ser ejecutado y compilado en diferentes versiones de Unix, Windows, y Macs. Debido a esto los scripts pueden ejecutarse de forma independiente al Sistema Operativo (SO).

Permite la implementación Orientada a Objetos, no necesita tipos de variables, tiene soporte para conectarse a muchos gestores de base de datos, además posibilita el tratamiento de errores de forma sencilla, es fácil de aprender y es libre.

Es posible encontrar documentación en cualquier lugar, así como respuestas a determinados problemas que se puedan presentar relacionados con el lenguaje, como foros de discusión online, ejemplos de códigos, tutoriales, videos con descarga gratis para el autoaprendizaje, entre otros.

Para realizar la implementación del submódulo se reafirma la utilización de PHP 5.2.0 como lenguaje de programación potente y eficaz.

Figura 8 Personal Home Page (PHP5).

Servidor Web Apache 2.2.3

El servidor Web es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

El servidor Web va a ser fundamental en el desarrollo de las aplicaciones del lado del servidor, que vayamos a construir, ya que se ejecutarán en él. [12]

Apache corre sobre múltiples sistemas operativos, es una tecnología gratuita de código abierto, altamente configurable, trabaja con PHP, Java, Perl y otros lenguajes. Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, no tiene una interfaz gráfica para su configuración.

Apache tiene amplia aceptación en la red, es el servidor HTTP más usado, la gran mayoría de los sitios Web en el mundo lo utilizan.

Además 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.

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 módulo para realizar una función determinada. [13]

Figura 9 Servidor Web Apache.

1.6 Conclusiones

Este capítulo abarca una panorámica general de los sistemas automatizados asociados a la validación en los EC, así como el estudio de las tendencias actuales de las metodologías, herramientas, y tecnologías de desarrollo de software. Luego de esta investigación se arribaron a las siguientes conclusiones:

Para realizar el diseño del submódulo perteneciente a la aplicación informática SIMDECC, con el propósito de solucionar el problema científico de la investigación se utilizará:

- RUP como metodología de desarrollo.
- Visual Paradigm 3.1 como herramienta CASE.
- Utilizando UML 2.0 como lenguaje de modelado.
- Symfony 1.0.10 como framework.
- Subversion 4.2.0 como herramienta para el control de versiones.
- Kompozer como herramienta para el diseño de interfaces.
- PHP 5.2.0 como lenguaje de programación.
- Apache 2.2.3 como servidor Web.

Capítulo 2: Diseño del Sistema

En el presente capítulo se realiza el diseño del submódulo “Validación de las variables del CRD y el control de errores”, con el objetivo de obtener un diseño que sirva como entrada al modelo de implementación. Además se fundamentan los patrones de diseño y arquitectura que se utilizarán para diseñar el submódulo. Con el diseño se logra un refinamiento del análisis teniendo en cuenta los requisitos no funcionales. Este capítulo es muy importante, porque de él depende que la aplicación se implemente sin imprecisiones.

La presente investigación dará solución a la problemática planteada anteriormente tomando como punto de partida los resultados arrojados por la investigación precedente, los cuales fueron diez requisitos funcionales y seis casos de uso del sistema.

Requisitos funcionales

1. Mostrar las reglas de validación que puede tener una subvariable (Mostrar las reglas de validación según el tipo de la subvariable). (CU 1)
2. Adicionar las reglas de validación definidas para una subvariable. (CU 1)
3. Mostrar las reglas de validación de una subvariable. (CU 1)
4. Mostrar la validación de una subvariable (CU 4)
    a) Mostrar casos.
    b) Mostrar variables de dependencia.
    c) Mostrar condiciones.
    d) Mostrar reglas de validación y de derivación.
5. Finalizar la validación de una subvariable. (CU 3)
6. Mostrar estado de la subvariable. (CU 3)
7. Mostrar estado de un modelo. (CU 5)
8. Finalizar la validación del CRD. (CU 5)
9. Verificar las consistencias de las reglas especificadas para una subvariable. (CU 2)
10. Mostrar la validación de todas las subvariables de un submodelo.(CU 6)

Casos de uso del sistema

1. Validar.
2. Analizar consistencia de las reglas especificadas.

3. Finalizar validación de una subvariable.
4. Mostrar la validación de una subvariable.
5. Finalizar validación de un CRD.
6. Emitir informe de la validación de un submodelo.

2.1 Patrones de Diseño

Los patrones de diseño son una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Están basados en la recopilación del conocimiento de los expertos en desarrollo de software. Es una experiencia real, probada y que funciona. Nos ayuda a no cometer los mismos errores.

Para lograr un buen diseño es recomendable el uso de patrones que conducen a arquitecturas más pequeñas, más simples y más comprensibles.

Patrones de diseño GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa *General Responsibility Asignment Software Patterns* (patrones generales de software para asignar responsabilidades). Los patrones básicos se refieren a cuestiones y aspectos fundamentales del diseño, algunos de estos patrones utilizados en este trabajo son:

- Experto
- Creador
- Alta Cohesión
- Bajo Acoplamiento
- Controlador

Experto:
Consiste en asignar responsabilidad a individuos que disponen de la información necesaria para llevar a cabo una tarea.

Creador:

El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Bajo Acoplamiento:

Permite asignar una responsabilidad de forma tal que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. El Bajo Acoplamiento es un principio que se debe recordar durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, dando la oportunidad de una mayor productividad.

Alta Cohesión:

Asignar una responsabilidad teniendo en cuenta que la cohesión siga siendo alta de modo que tenga responsabilidades moderadas en un área funcional y colabore con las otras para llevar a cabo las tareas. Como el patrón Bajo Acoplamiento, también Alta Cohesión es un principio que debemos tener presente en todas las decisiones de diseño: es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño.

Controlador:

No es más que asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona.

Otros patrones de diseño

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. [14]

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**.**

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. [14]

Patrón Singlenton:

El patrón de diseño Singlenton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase. Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

El patrón Singlenton se implementa creando en la clase un método que crea una instancia del objeto solo 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 datos debe estar disponible para todos los demás objetos de la aplicación.

El patrón Singlenton 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. [15]

Patrón Decorator:

Responde a la necesidad de añadir dinámicamente funcionalidad de adorno a un objeto individual de forma dinámica y transparente, o sea, sin afectar a otros objetos. 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 para proporcionar opciones de embellecimiento en las interfaces.

Ventajas:

- Es más flexible que la herencia estática.
- Evita que las clases altas de la jerarquía estén demasiado cargadas de funcionalidad.
- Se reduce el número de clases y el árbol de herencia de clases. [16]

Patrón Fachada

Fachada es un patrón que se utiliza para estructurar el diseño, que simplifica el acceso a determinadas clases. Su objetivo es disminuir la dependencia entre las clases, el cual permite acceder al resto de ellas, por lo que si estas cambian solo se actualizaría la clase fachada, en la cual estarán los métodos de acceso a datos. De esta forma no se afecta la aplicación cliente.

En la presente investigación se utilizan los patrones de diseño para un mejor entendimiento del diseño y una buena asignación de responsabilidades a las clases. Se manifiesta el uso del patrón Singlenton en la capa del Modelo, en la cual realiza un importante papel pues permite crear una única instancia de la clase Fachada. También se ponen de manifiesto los patrones de asignación de responsabilidades como Bajo Acoplamiento ya que las clases utilizadas tienen poca dependencia entre ellas y no presentan herencias profundas, Controlador con el objetivo de centralizar las actividades y mantener una alta cohesión en todo momento ya que mantiene una labor única dentro del submódulo.

Además se evidencia el uso del patrón Fachada en la clase fachada de cada caso de uso, pues estas son clases que gestionan todas las peticiones en la capa Modelo y contiene los métodos de acceso a datos, siendo un intermediario entre las capas Controlador y Modelo, evidenciando así el uso del patrón Indirección. También se utiliza el patrón Decorator pues es implementado por Symfony el cual causa efecto en la vista, agregando funciones de embellecimiento a las plantillas.

2.2 Patrones de Arquitectura

Existen muchos patrones de arquitectura, pero aquí sólo se listaran algunos de los más interesantes.

- Modelo-Vista-Controlador (MVC).
- Arquitecturas en Capas.
- Arquitecturas Orientadas a Objetos.
- Arquitecturas Basadas en Componentes.
- Arquitecturas Orientadas a Servicios.

Modelo Vista Controlador (MVC)

Modelo-Vista-Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de la aplicación en tres componentes distintos. Se ve frecuentemente en aplicaciones Web, donde la Vista es la página HTML y el código que provee de datos dinámicos a la página, el Modelo es el Sistema de Gestión de Base de Datos y el Controlador representa la lógica de negocio, está compuesto por 3 niveles:

- El modelo: representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista: transforma el modelo en una página Web que permite al usuario interactuar con ella.
- El controlador: se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

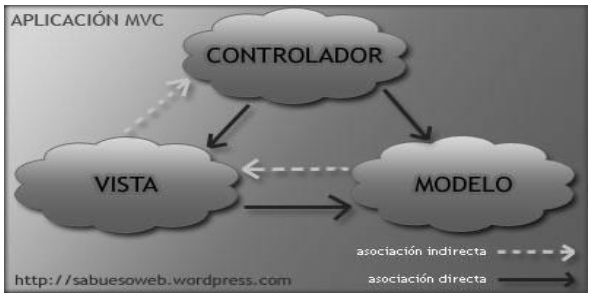

Figura 10 Patrón Modelo Vista Controlador.

La arquitectura MVC divide el modelo y la vista por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

El patrón Modelo-Vista-Controlador presenta múltiples ventajas por ejemplo, soporte de múltiples vistas dado que la vista se haya separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes. Agregar nuevas opciones de presentación no afecta al modelo, pues este no depende de la vista.

En la presente investigación se hará uso del patrón Modelo-Vista-Controlador (MVC), por todas las ventajas que brinda y por ser el patrón arquitectónico que implementa el framework Symfony.

2.3 Modelo de Diseño

El diseño es el refinamiento del análisis, para el cual se tiene en cuenta como el sistema cumplirá sus metas. Es la entrada al flujo de trabajo de implementación y debe ser suficiente para implementar el submódulo sin imprecisiones.

En el diseño se modela el sistema para que cumpla con 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 se le de 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. [17]

Paquetes del Diseño.

Es una colección de clases, relaciones, diagramas, realizaciones de casos de uso y otros paquetes que estén de alguna forma relacionados. Se usa para estructurar el modelo de diseño dividiéndolo en partes más pequeñas.

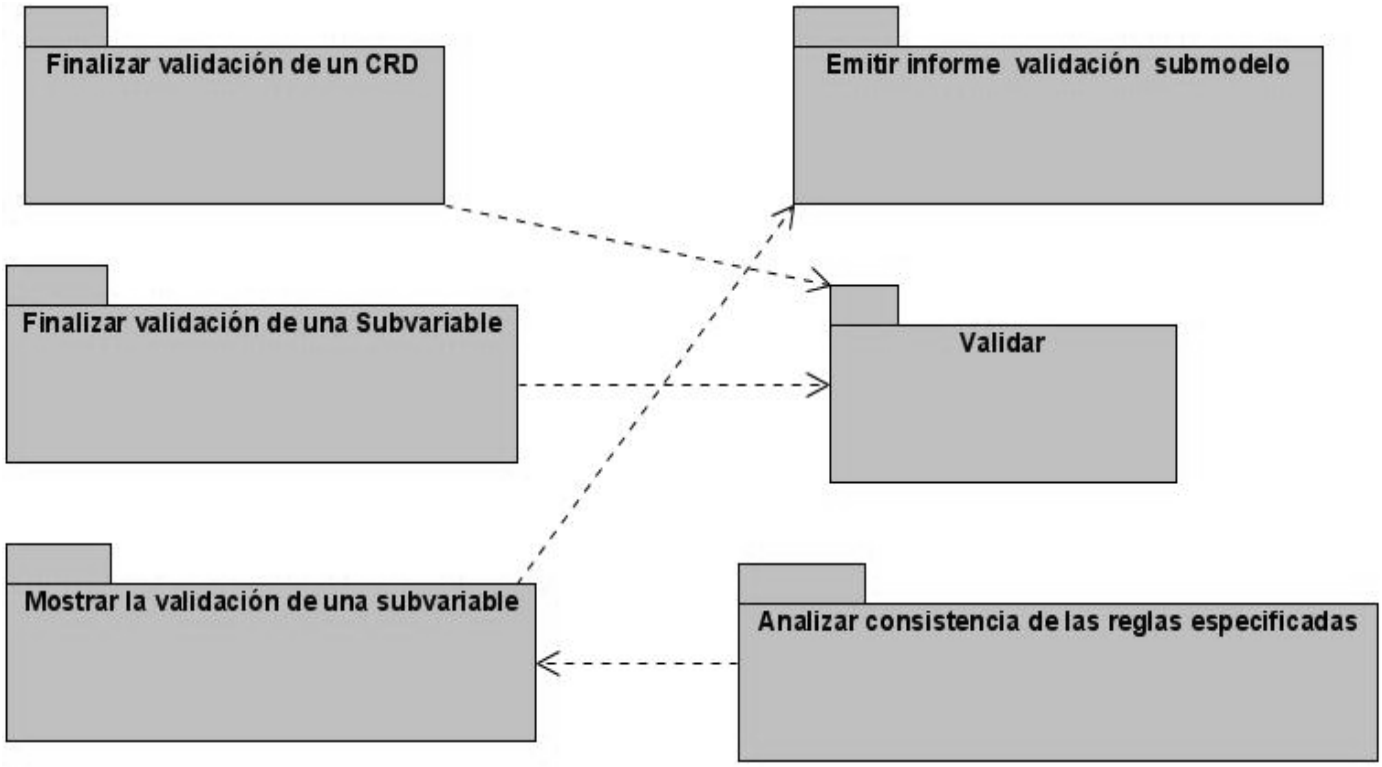

Figura 11 Paquetes del Diseño.

Descripción de las clases del diseño

Tabla 1 Descripción de la clase del Diseño: FinalizarValSubvariableActions.

Nombre: FinalizarValSubvariableActions	
Tipo: Clase (Esta clase contiene todas las funciones correspondientes para finalizar la validación de una subvariable. Llama a la clase FinalizarValSubvariableFachadaValidacion la cual contiene los métodos de acceso a datos con el objetivo de utilizar las funciones relacionadas con el paquete Modelo)	
Responsabilidades	
Nombre:	**Descripción:**
executeCasos()	Muestra la página casos, brinda la posibilidad de finalizar la validación de una subvariable.
executeMostrarVariablesSubmodelo()	Llama a la función CambiarEstadoSubvariable ($idModelo, idSubmodelo, $idVariable, $idSubvariable) de la clase FinalizarValSubvariableFachadaValidacion, el atributo validado de la clase Psubvariable que inicialmente es false toma el valor true. Luego muestra la página con todas las subvariables correspondientes del submodelo y se muestra un icono al lado de la subvariable validada.

Tabla 2 Descripción de la clase del Diseño: ValidarActions.

Nombre: validarActions	
Tipo: Clase (Esta clase contiene todas las funciones correspondientes a la validación de una subvariable. Llama a la clase ValidarFachadaValidacion la cual contiene los métodos de acceso a datos con el objetivo de utilizar las funciones relacionadas con el paquete Modelo)	
Responsabilidades	
Nombre:	**Descripción:**
executeCasos()	Muestra la página casos con la tabla de estado, brinda la posibilidad de establecer las reglas de validación.
executeMostrarReglasValTipo()	Muestra las reglas de validación posibles para la subvariable a validar en dependencia del tipo de dato. Busca el tipo de dato de la subvariable a validar llamando a la función BuscarTipoSubvariable($idModelo, $idSubmodelo, $idVariable, $idSubvariable) de la clase ValidarFachadaValidacion. En dependencia del tipo de dato de la subvariable llama a la función CargarReglasVal () de la clase ValidarFachadaValidacion la cual carga los valores posibles a tomar por esa subvariable.
hasErrorMostrarReglasValTipo()	Función que se ejecuta cuando existe error en la validación del formulario.
executeGuardarReglasVal()	Llama a la función GuardarReglasVal () de la clase ValidarFachadaValidacion para guardar la regla de validación seleccionada por el usuario. Llama a la función executeCargarReglasVal ().
executeCargarReglasVal()	Llama a la función CargarReglasVal () de la clase ValidarFachadaValidacion para cargar la regla de validación seleccionada por el usuario para el caso en cuestión y mostrarla en la tabla de estado en la página casos, para ello llama a la función executeCasos ().

Tabla 3 Descripción de la clase del Diseño: FinalizarValidacionCRDActions.

Nombre: FinalizarValidacionCRDActions	
Tipo: Clase (Esta clase contiene todas las funciones correspondientes para finalizar la validación de un CRD. Llama a la clase FinalizarValCRDFachadaValidacion la cual contiene los métodos de acceso a datos con el objetivo de utilizar las funciones relacionadas con el paquete Modelo)	
Responsabilidades	
Nombre:	**Descripción:**
executeModelos()	Llama a la función VerificarValidacionModelos($idEc) de la clase FinalizarValCRDFachadaValidacion para verificar si todas la subvariables del CRD están validadas, si esto se cumple llama a la función CambiarEstadoModelos (modelos) y el atributo validado de la clase PModelo que inicialmente es false toma el valor true, luego muestra la página con todos los modelos correspondientes a un Ensayo Clínico y se muestra un icono al lado del modelo indicando que ya se finalizó la validación de todas las subvariables. De lo contrario si queda alguna subvariable por validar el sistema muestra el mensaje "No puede finalizar validación del CRD y permite volver a validar".

Tabla 4 Descripción de la clase del Diseño: MostrarValSubvariableActions.

Nombre: MostrarValSubvariableActions	
Tipo: Clase (Esta clase contiene todas las funciones correspondientes para mostrar la validación de una subvariable. Llama a la clase MostrarValFachadaValidacion la cual contiene los métodos de acceso a datos con el objetivo de utilizar las funciones relacionadas con el paquete Modelo)	
Responsabilidades	
Nombre:	**Descripción:**
executeVariablesSubmodelo()	Muestra las páginas con todas las subvariables de un submodelo, brinda la posibilidad de ver la validación de una subvariable.
executeTablaEstado()	Llama a la función CargarDatosTablaEstado ($idModelo, $idSubmodelo, $idVariable, $idSubvariable) de la clase MostrarValFachadaValidacion para cargar las variables de dependencia, los casos, las condiciones, las reglas de derivación y las reglas de validación de la subvariable seleccionada, todos estos datos se mostrarán en la página tabla de estado.

Tabla 5 Descripción de la clase del Diseño: EmitirInformeValidacionActions.

Nombre: EmitirInformeValidacionActions	
Tipo: Clase (Esta clase contiene todas las funciones correspondientes para emitir el informe de validación de un submodelo. Llama a la clase EmitirInformeValFachadaValidacion la cual contiene los métodos de acceso a datos con el objetivo de utilizar las funciones relacionadas con el paquete Modelo)	
Responsabilidades	
Nombre:	**Descripción:**
executeSubmodelos()	Muestra todos los submodelos de un modelo, brinda la posibilidad de emitir el informe de validación de un submodelo.
executeListarVariables()	Llama a la función BuscarSubvariable ($idModelo, $idSubmodelo) de la clase EmitirInformeValFachadaValidacion para listar todas las subvariables del submodelo seleccionado, luego llama a la función CargarDatos ($idModelo, $idSubmodelo, $idVariable, $idSubvariable) para cargar las variables de dependencia, los casos, las condiciones, las reglas de derivación y las reglas de validación de las subvariables del submodelo seleccionado, todos estos datos se mostrarán en la tabla de estado en la página ver Información**.**

Tabla 6 Descripción de la clase del Diseño: AnalizarConsistenciaReglasActions.

Nombre: AnalizarConsistenciaReglasActions	
Tipo: Clase (Esta clase contiene todas las funciones correspondientes al paquete del diseño Analizar Consistencia de las reglas especificadas. Llama a la clase AnalizarConsistenciaFachadaVal con el objetivo de utilizar las funciones relacionadas con el paquete Modelo)	
Responsabilidades	
Nombre:	**Descripción:**
executeAnalizarConsistenciaReglasVal()	Muestra la página casos con la tabla de estado que da la opción de analizar la consistencia de las reglas especificadas. Llama a la función AnalizarConsistenciaReglas($id_modelo, id_submodelo, $id_variable, $id_subvariable) de la clase AnalizarConsistenciaFachadaVal la cual verifica que las reglas no se solapen y no haya inconsistencias entre ellas. Si no hay inconsistencias muestra un mensaje y habilita el botón finalizar validación de la subvariable de lo contrario muestra un mensaje diciendo que hay inconsistencias y el usuario puede arreglar los errores.

Diagrama de Clases del Diseño

Con el uso del framework Symfony se introduce el patrón arquitectónico Modelo-Vista-Controlador, para lograr una mejor visualización los elementos del diseño se agruparan en las 3 capas que este patrón describe:

Modelo: en esta capa se encuentran representadas las clases que tienen la lógica del negocio y que se encargan de la abstracción de la base de datos y del acceso a los datos. Además se cuenta con una clase Fachada que es la intermediaria entre las capas Controlador y Modelo, además de simplificar y controlar de esta forma el acceso a los datos.

Vista: La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes:

- La plantilla: es la representación de los datos de la acción que se está ejecutando.
- El layout: contiene el código HTML común a todas las páginas, puede modificarse para definir zonas en las que se insertan componentes externos y otros elementos.

Controlador: En Symfony, el controlador, contiene el código que vincula la lógica de negocio con la presentación, está dividido en varios componentes que se utilizan con diversos fines, en los diagramas de clases del diseño están representados los siguientes:

- Acciones: contienen la lógica de la aplicación, verifican la integridad de las peticiones, preparan los datos requeridos por la capa de presentación y definen las variables que son fundamentales para las vista.
- Controlador frontal: es el único punto de entrada a la aplicación, carga la configuración y determina la acción a ejecutarse. Las peticiones Web son manejadas por un solo controlador frontal.

A continuación se representan los diagramas de clases del diseño correspondientes a cada uno de los casos de usos del sistema.

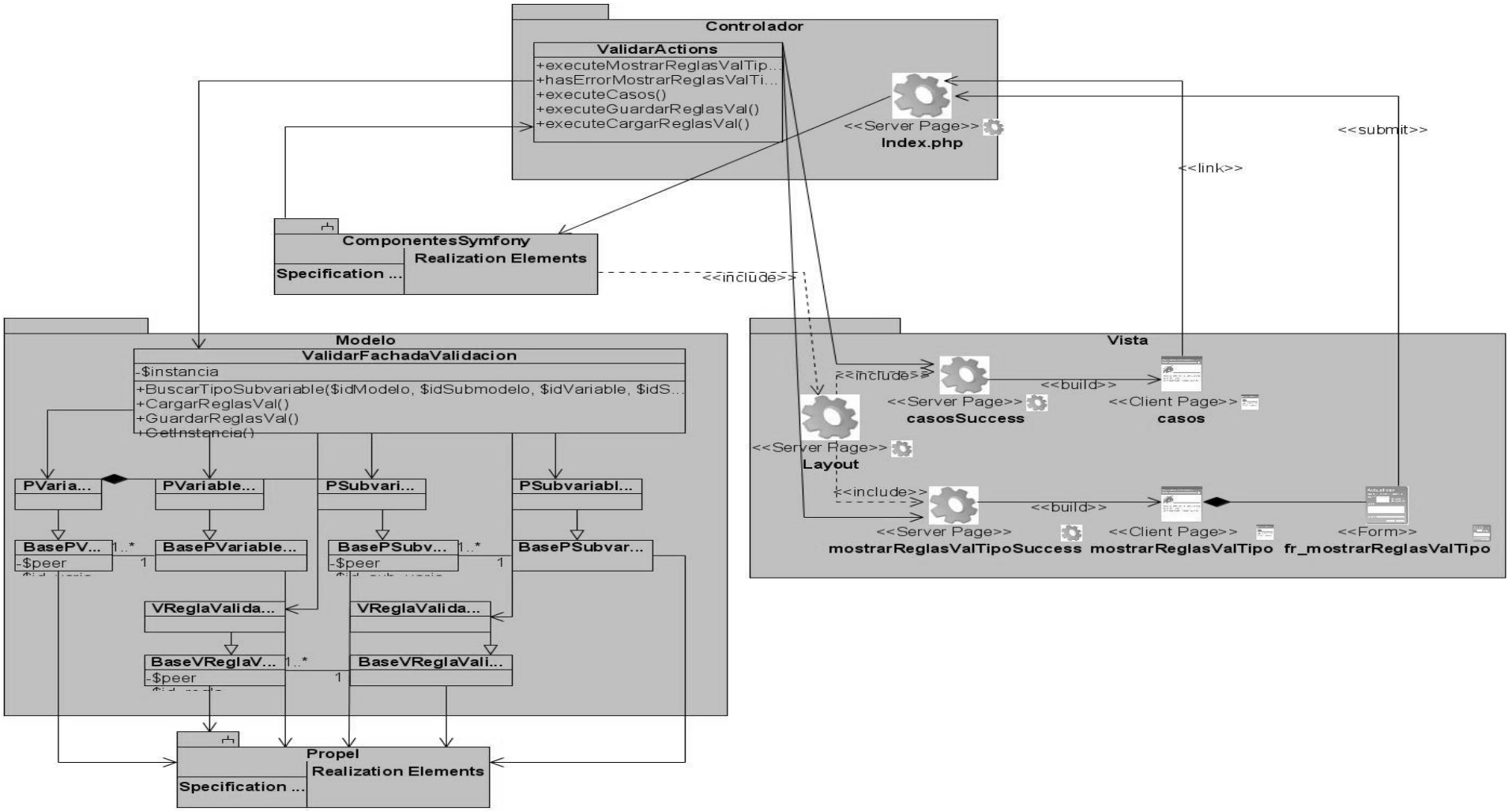

Figura 12 Diagrama de clases del Diseño del CU Validar.

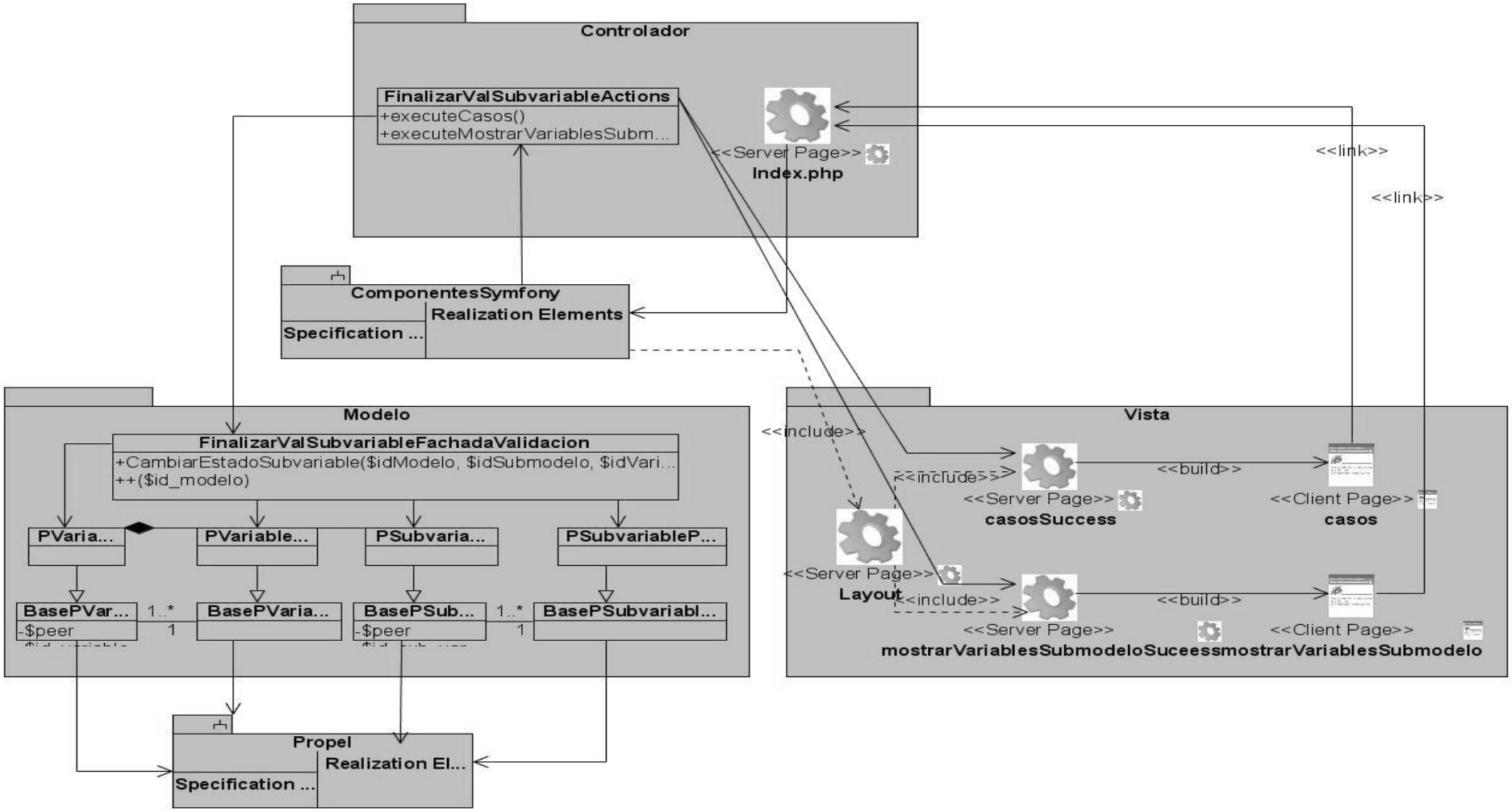

Figura 13 Diagrama de clases del Diseño del CU Finalizar validación de una subvariable.

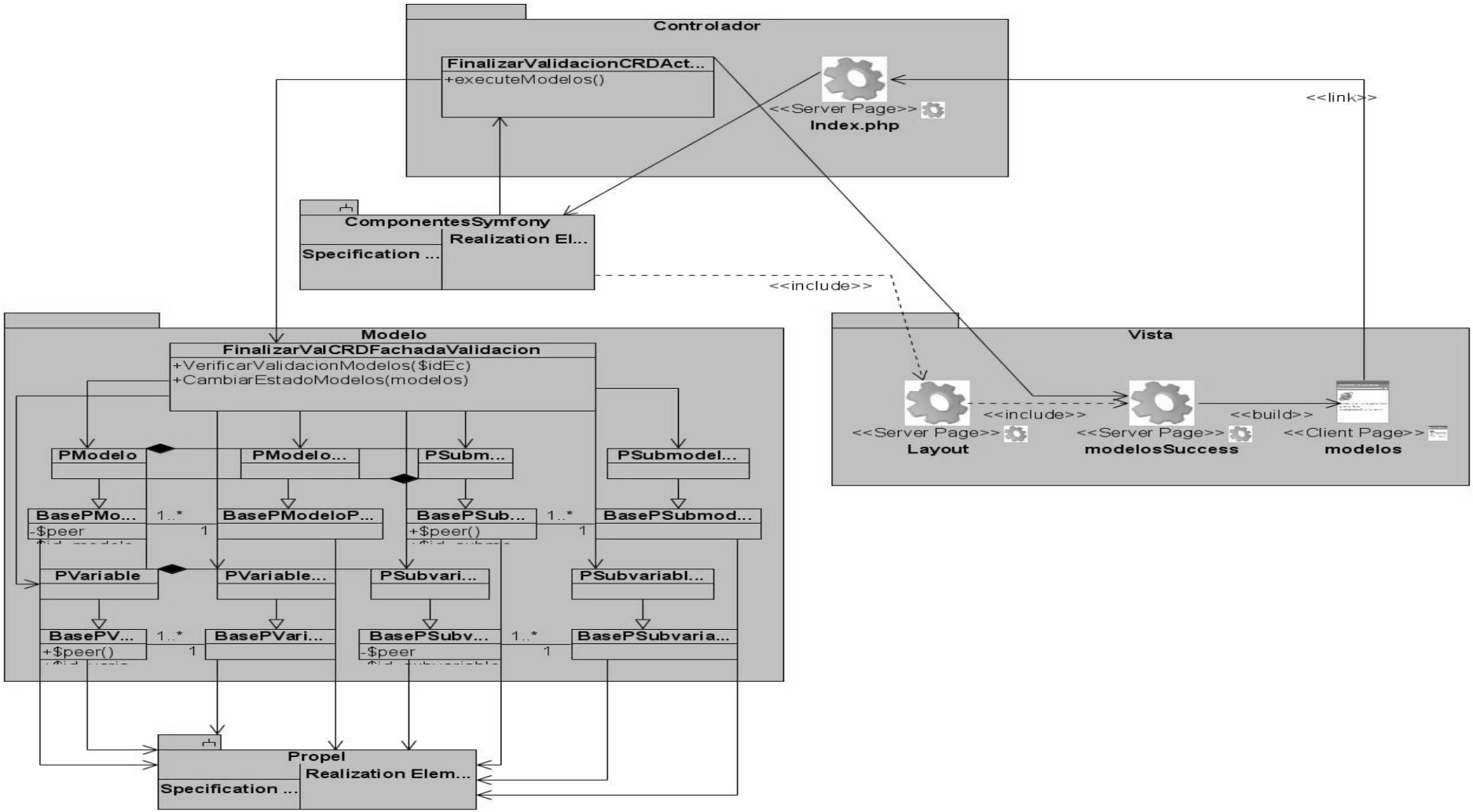

Figura 14 Diagrama de clases del Diseño del CU Finalizar validación del CRD.

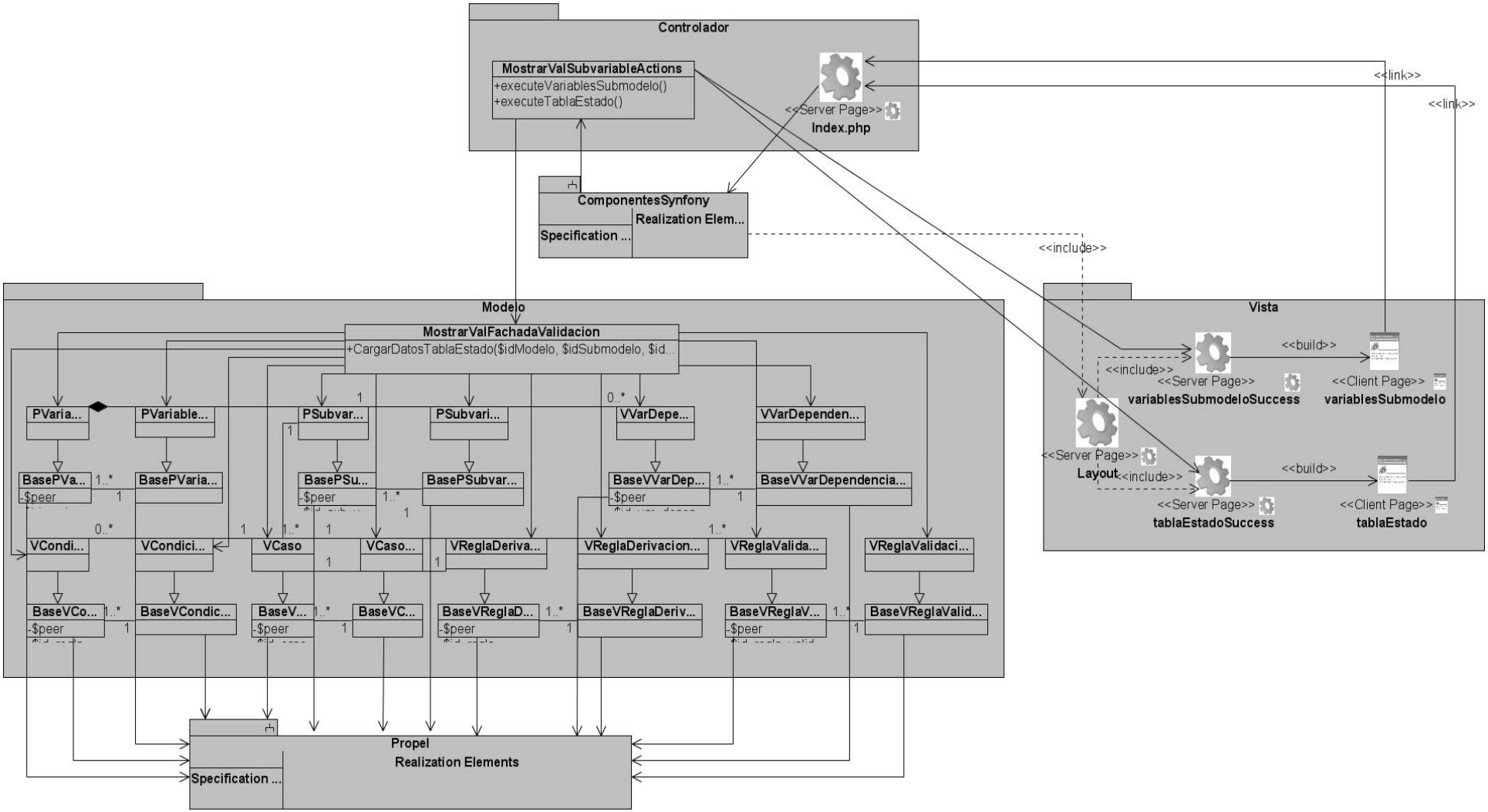

Figura 15 Diagrama de clases del Diseño del CU Mostrar validación de una subvariable.

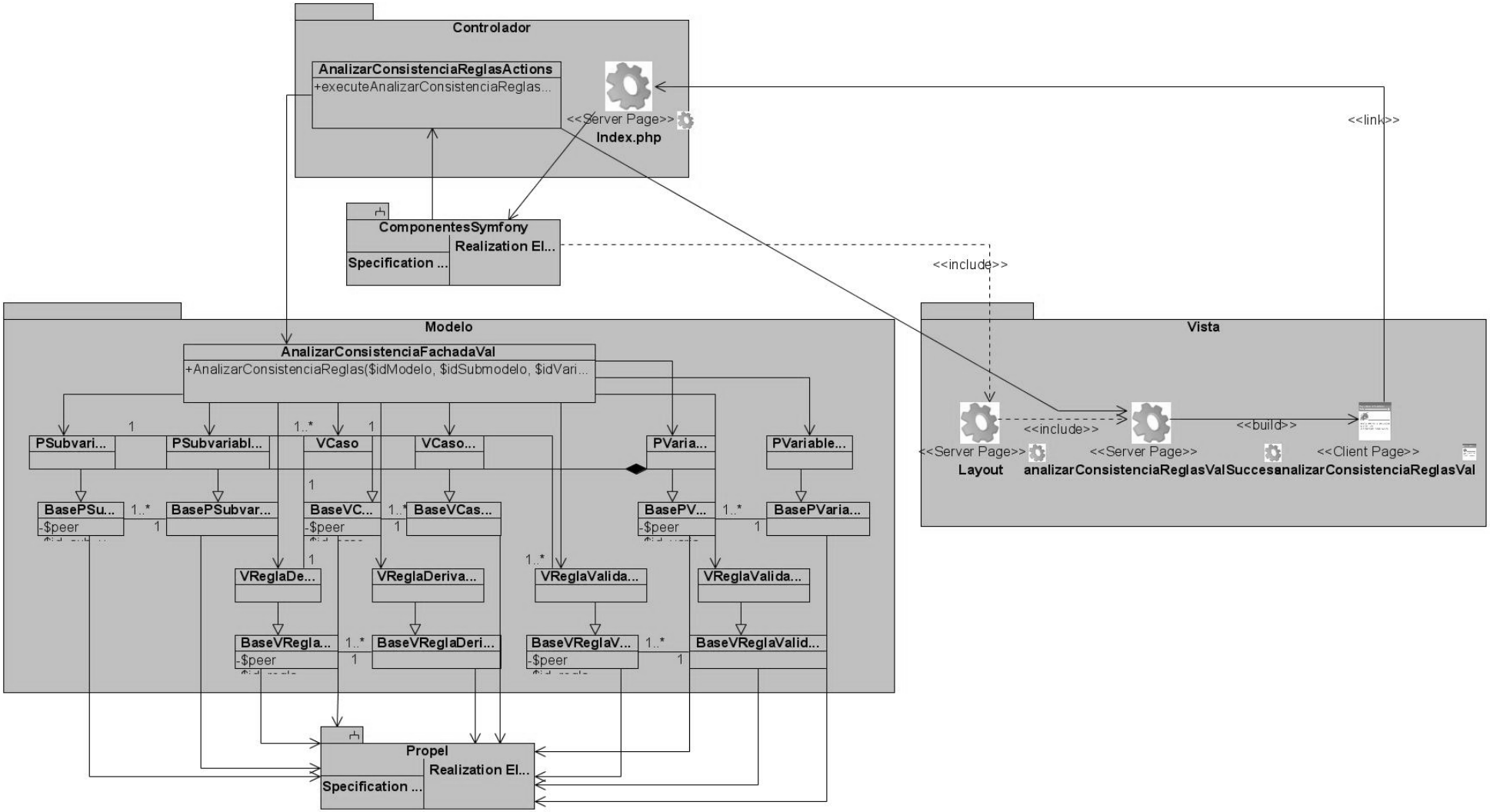

Figura 16 Diagrama de clases del Diseño del CU Analizar consistencia de las reglas especificadas.

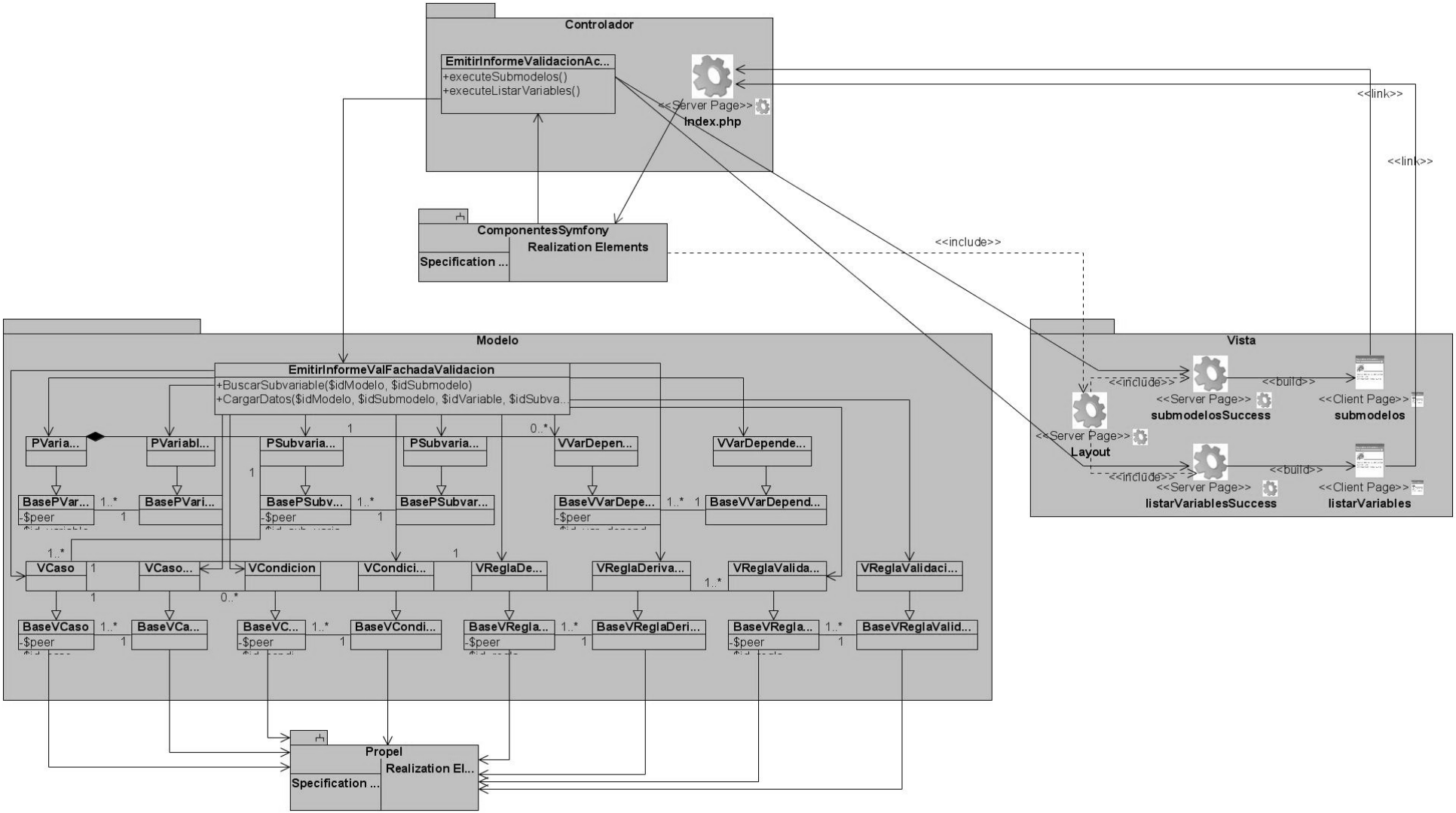

Figura 17 Diagrama de clases del Diseño del CU Emitir informe de la validación de un submodelo.

Diagramas de Interacción.

Los diagramas de interacción son aquellos que muestran los flujos de mensajes con cada acción y métodos que deben ejecutarse en un momento determinado.

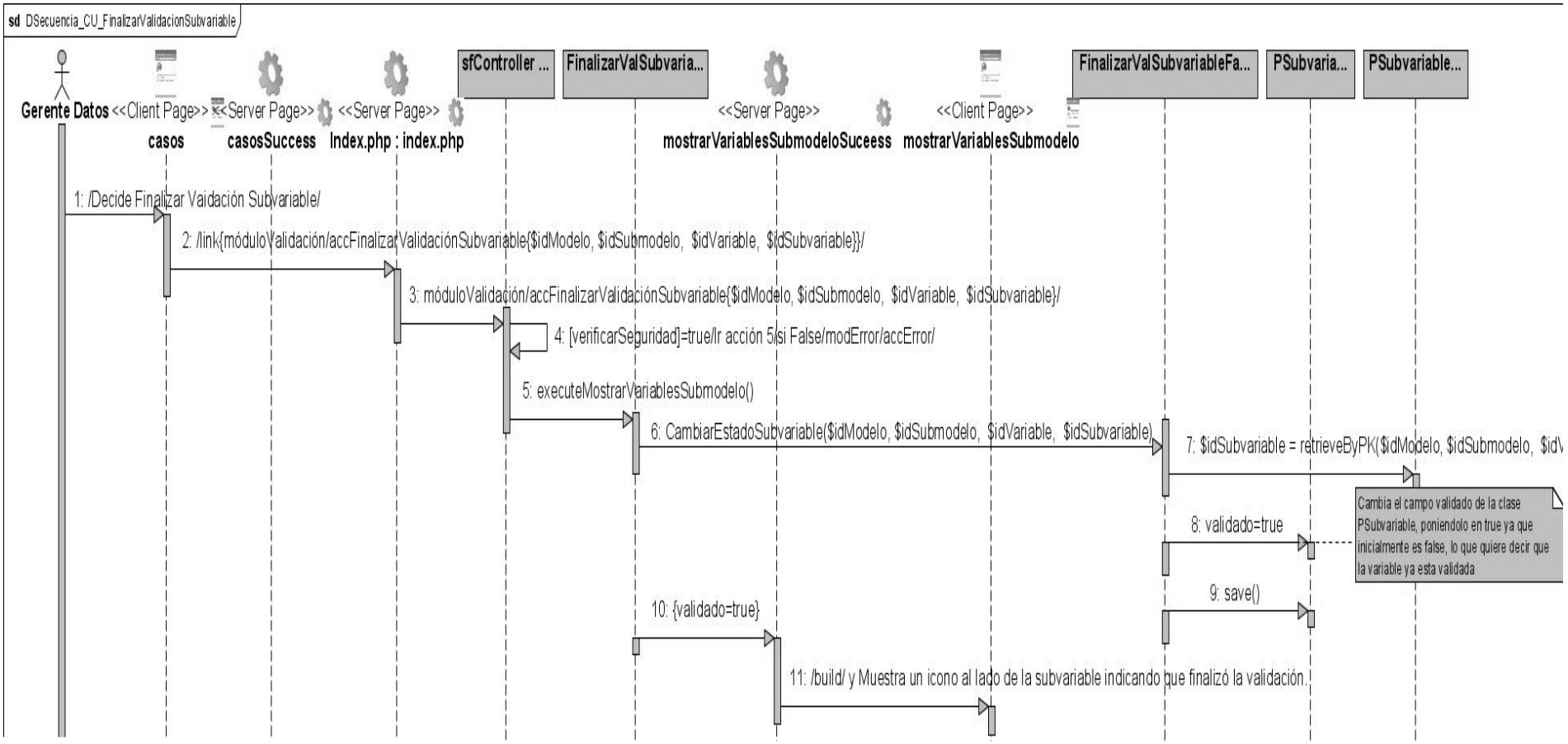

Figura 18 Diagrama de Secuencia para el Diseño del CU Finalizar validación de una subvariable.

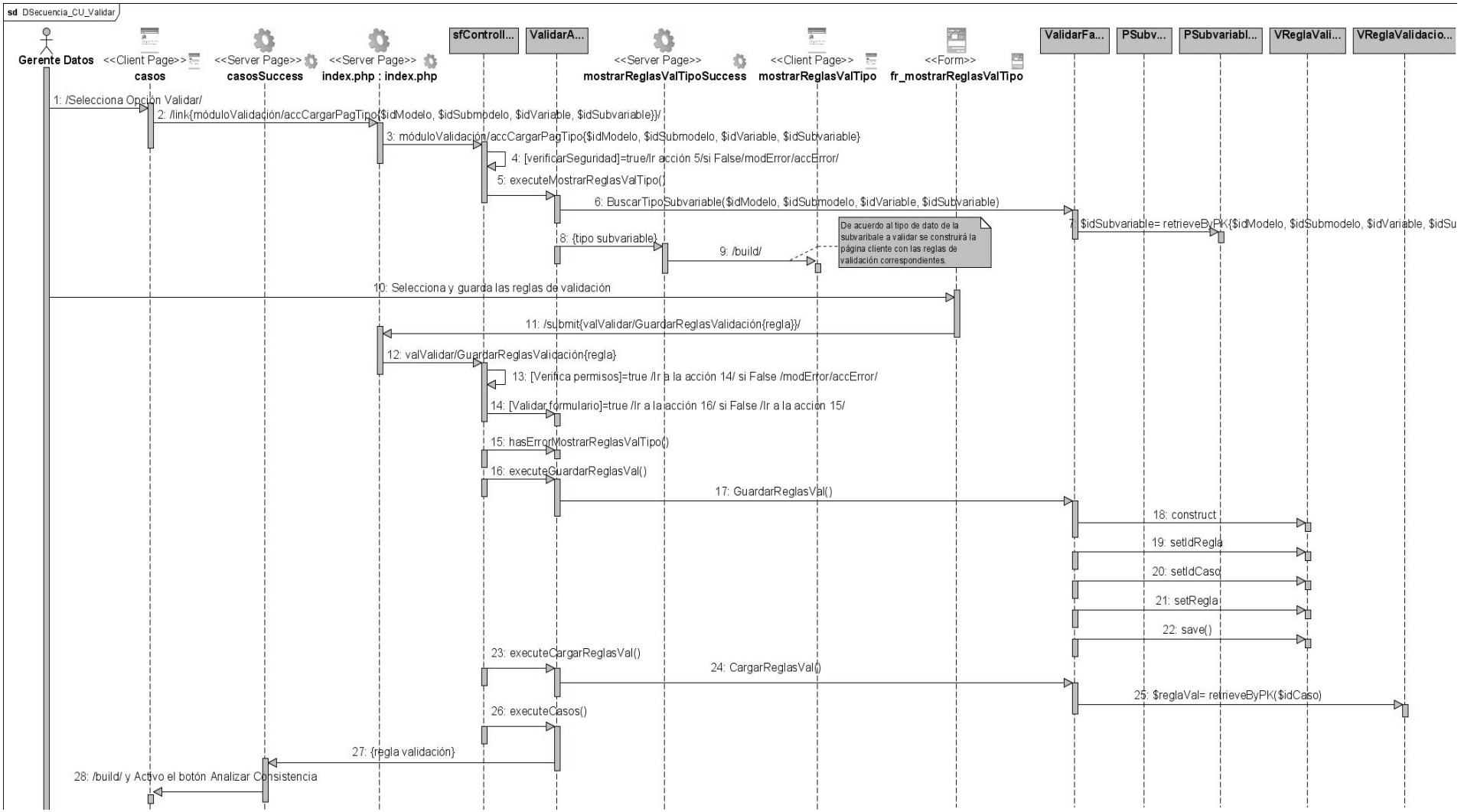

Figura 19 Diagrama de Secuencia para el Diseño del CU Validar.

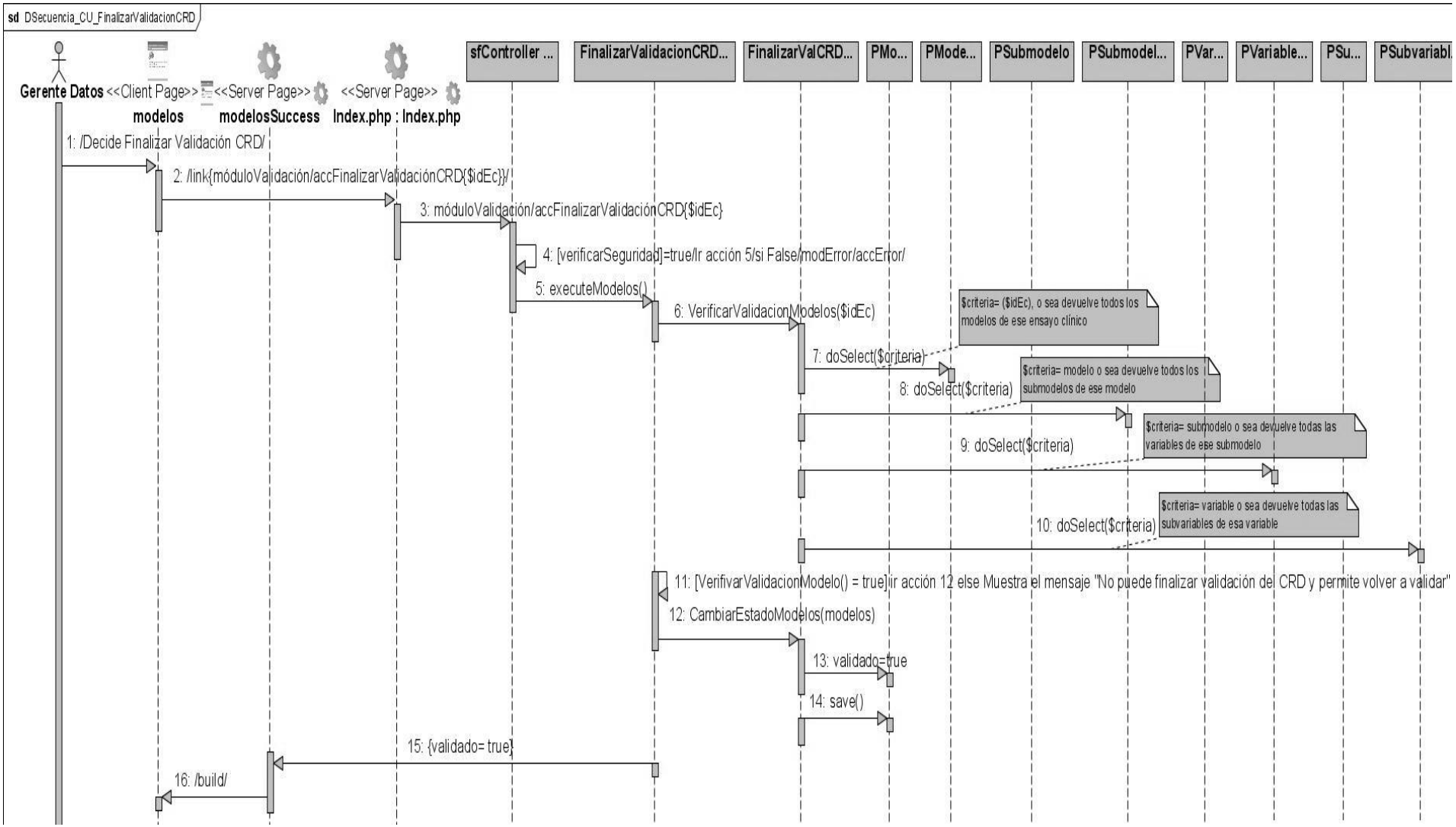

Figura 20 Diagrama de Secuencia para el Diseño del CU Finalizar validación del CRD.

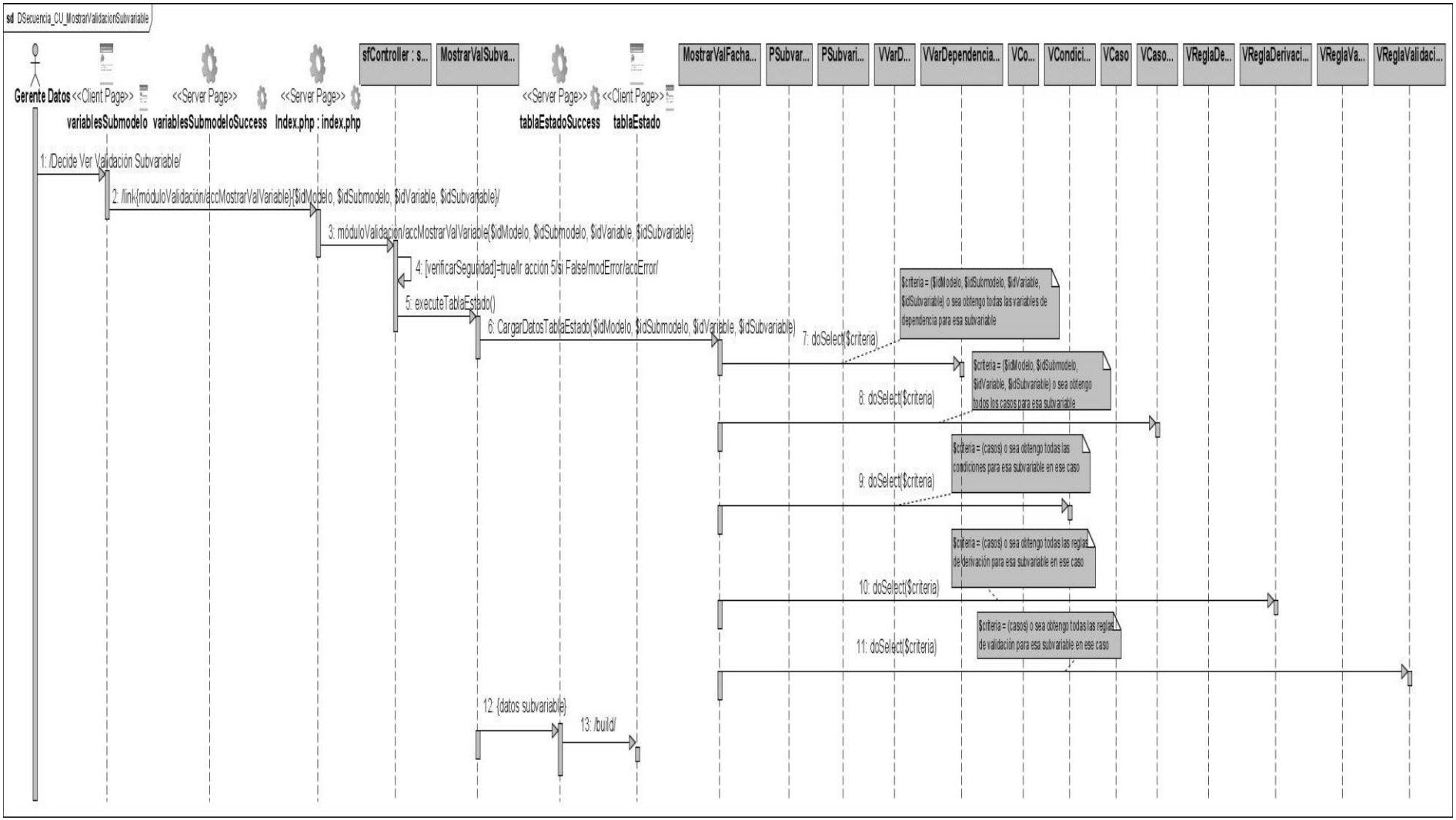

Figura 21 Diagrama de Secuencia para el Diseño del CU Mostrar validación de una subvariable.

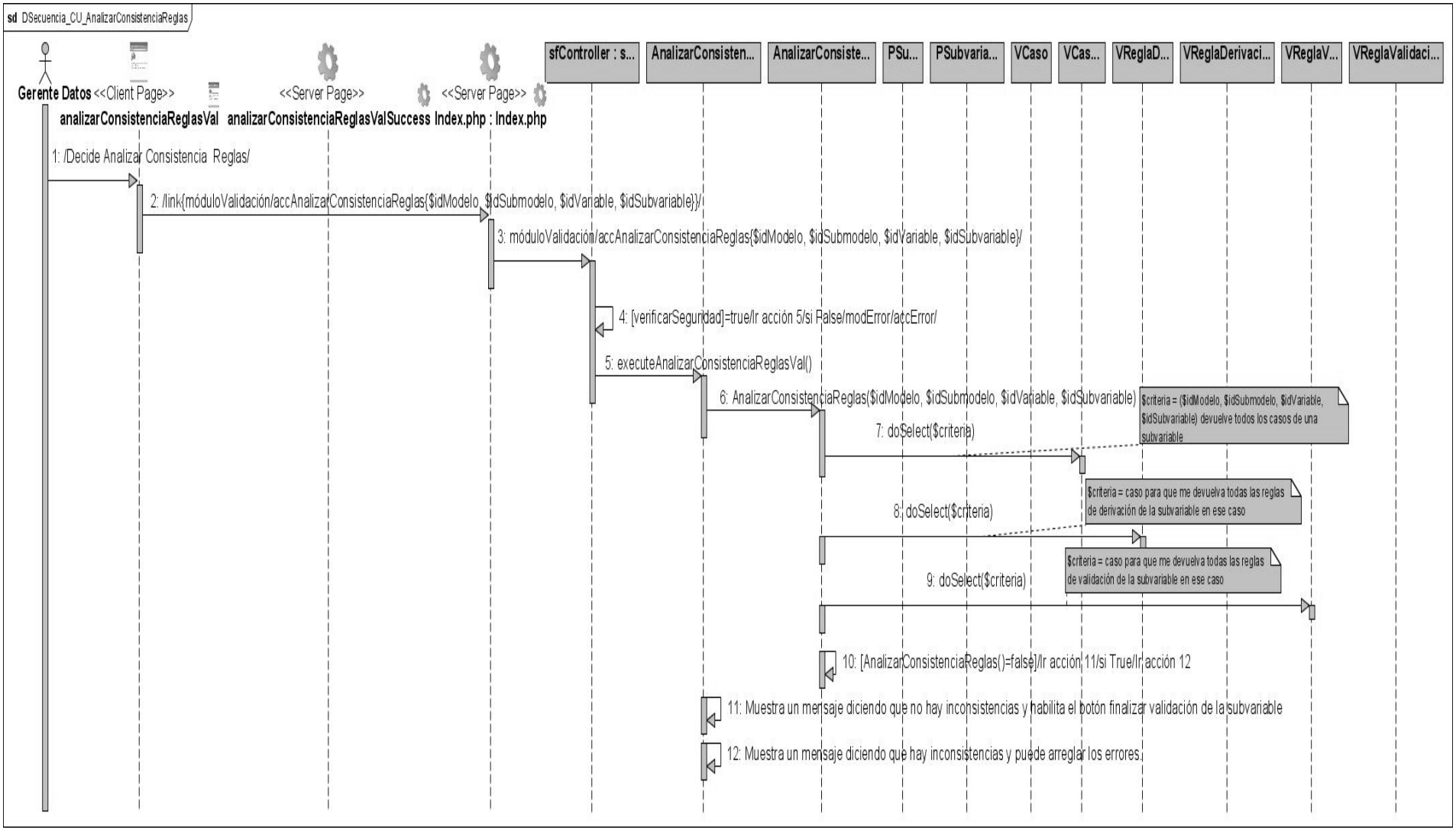

Figura 22 Diagrama de Secuencia para el Diseño del CU Analizar consistencia de las reglas especificadas.

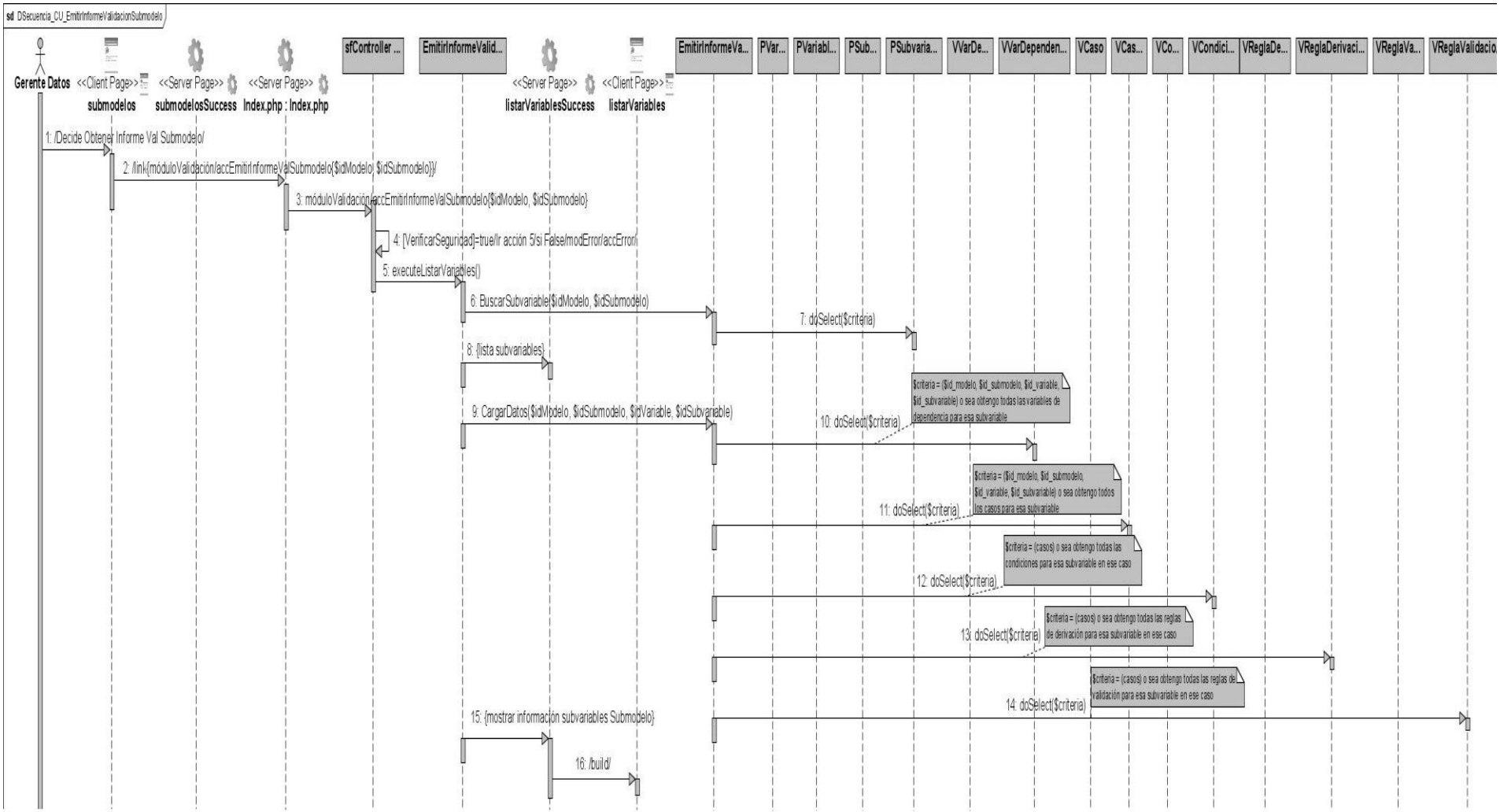

Figura 23 Diagrama de Secuencia para el Diseño del CU Emitir informe de la validación de un submodelo.

Prototipos no funcionales

Los prototipos de interfaz de usuario facilitan la comunicación entre el cliente y la aplicación Web, y permiten comprobar si realmente el sistema que será implementado cumple con los requerimientos funcionales, además permiten verificar si coincide con las necesidades planteadas por el cliente. Estos prototipos 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.

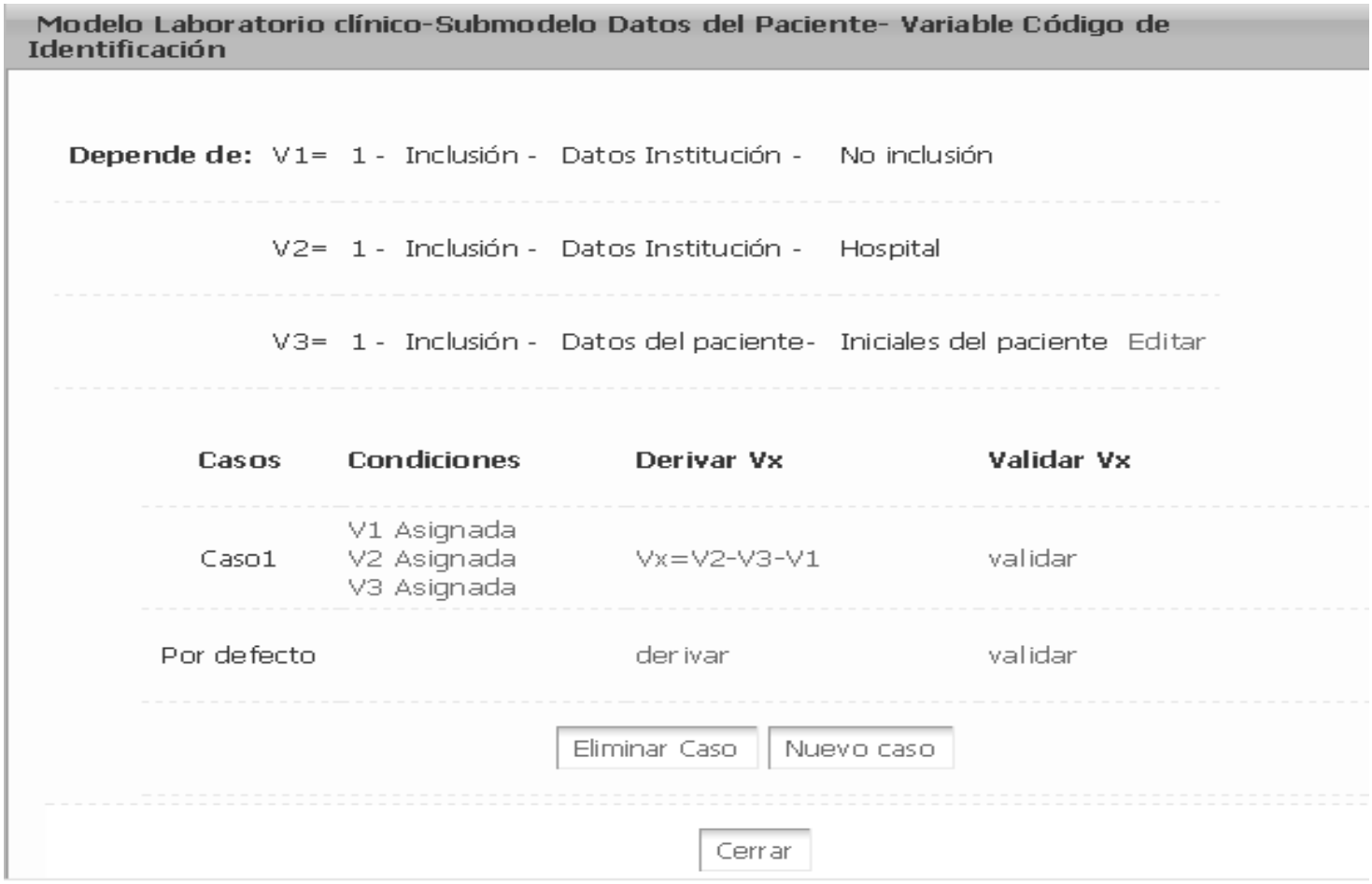

Figura 24 Página Casos. CU Validar.

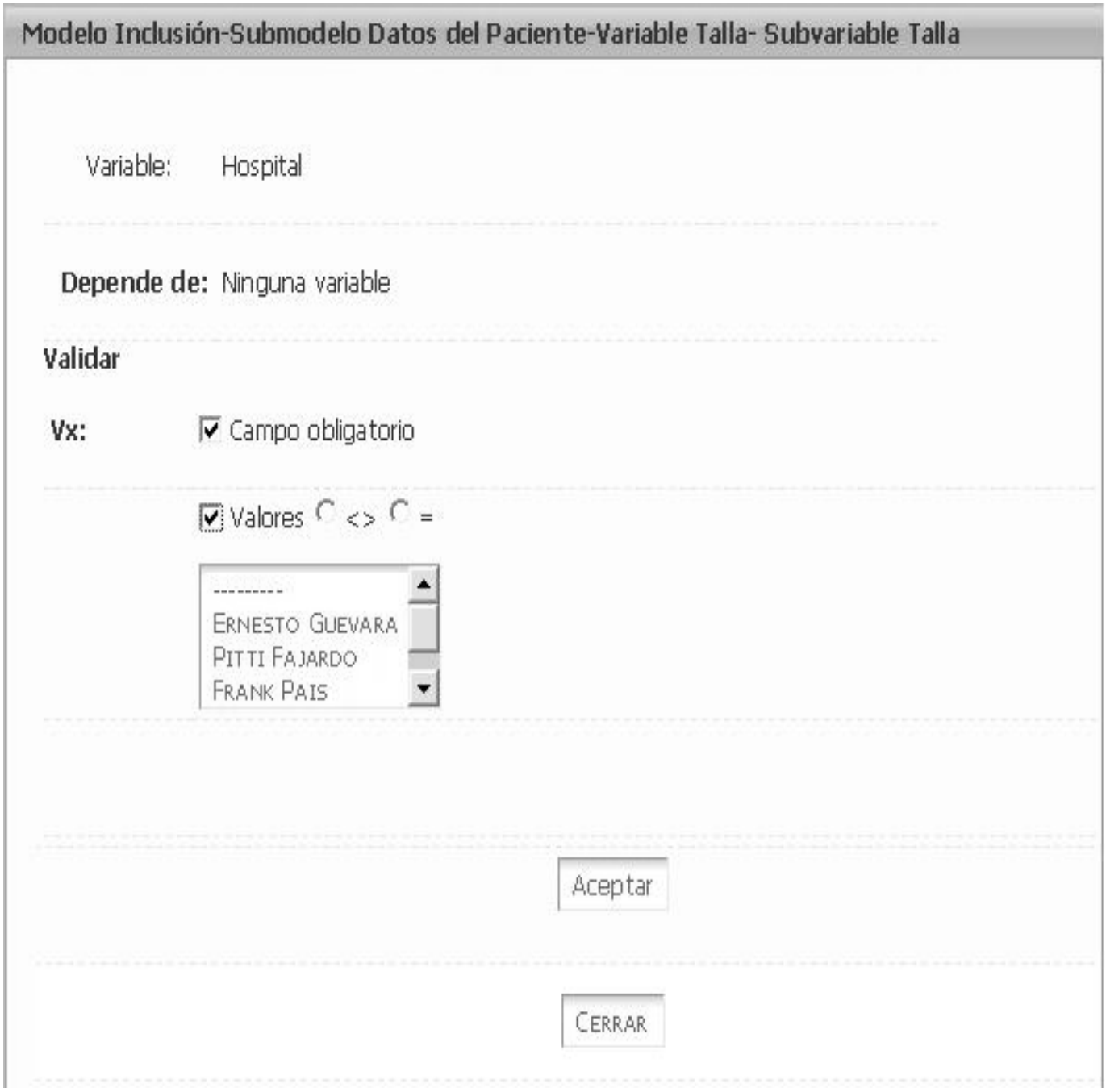

Figura 25 CU Validar.

Figura 26 CU Analizar Consistencia de las reglas especificadas y CU Finalizar validación de una subvariable.

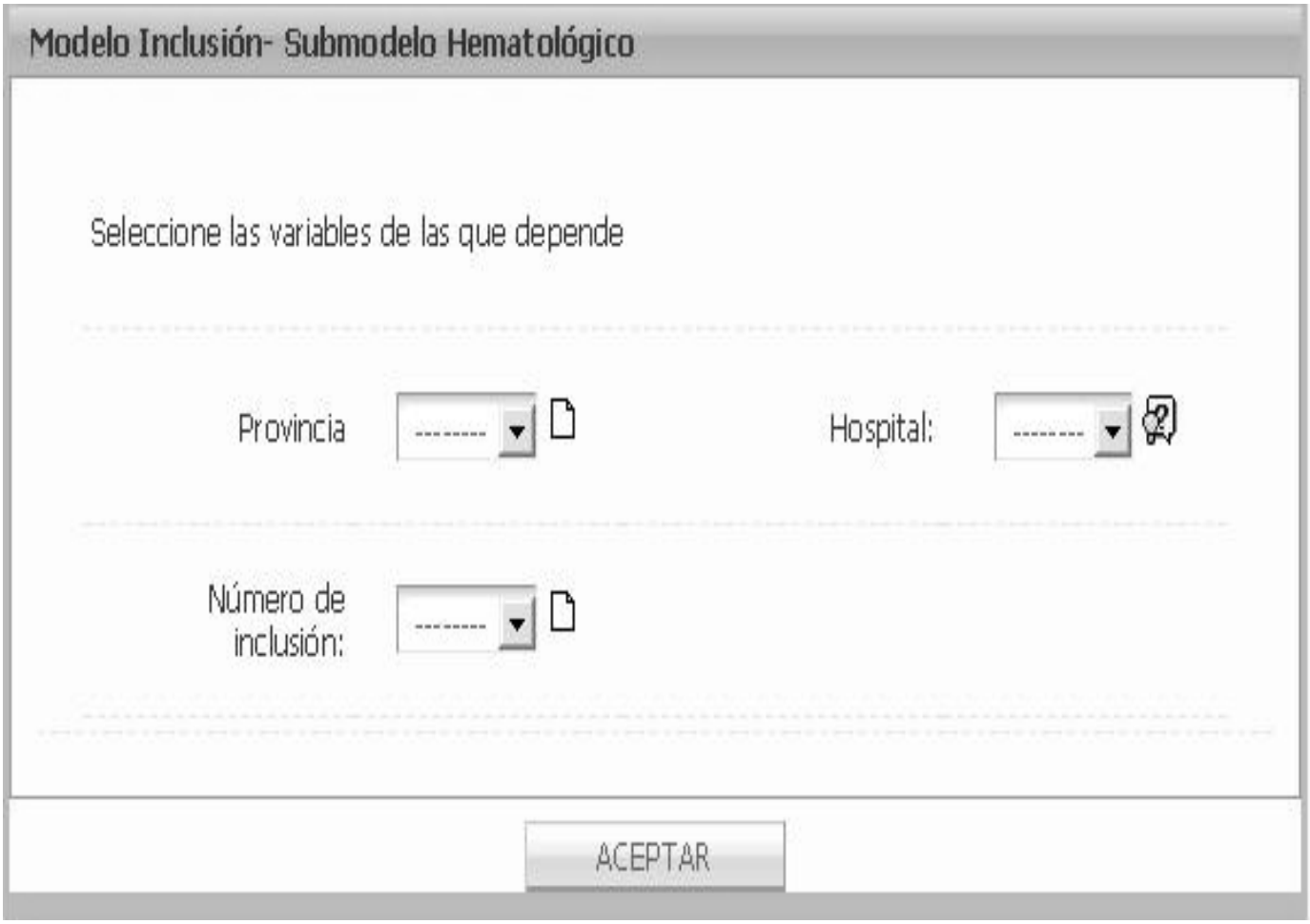

Figura 27 CU Mostrar validación de una subvariable.

Figura 28 CU Mostrar validación de una subvariable.

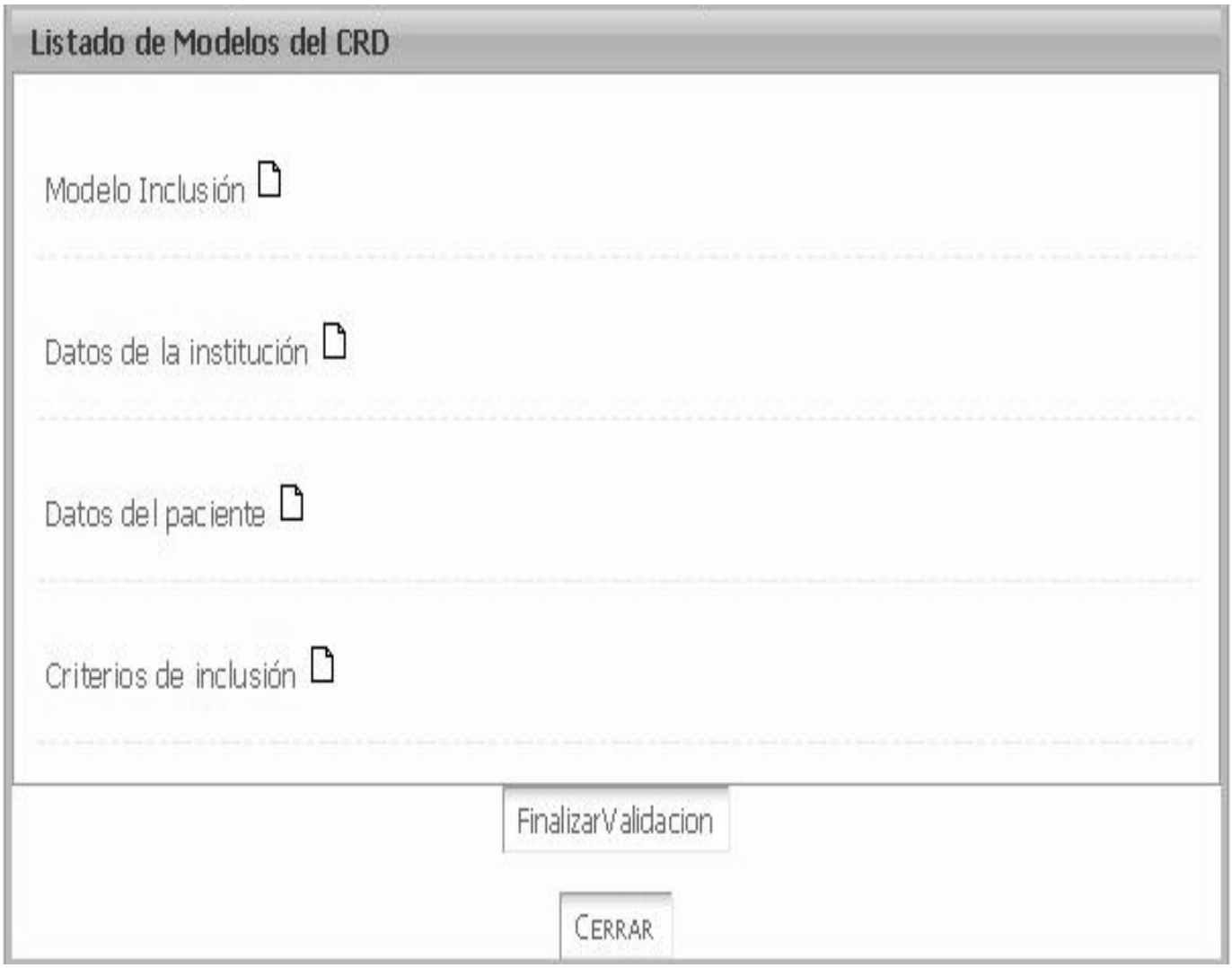

Figura 29 CU Finalizar validación del CRD.

Figura 30 CU Finalizar validación del CRD.

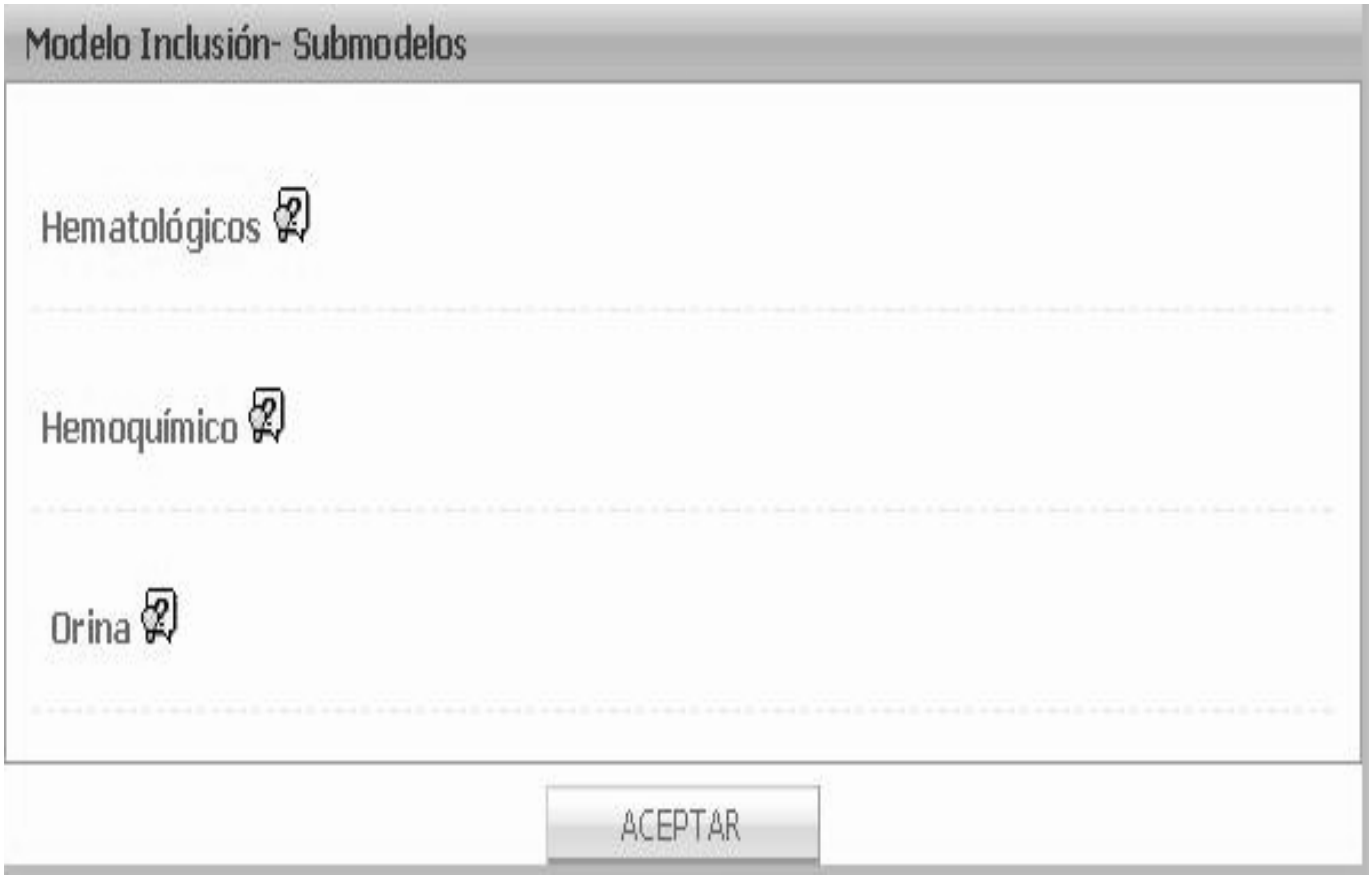

Figura 31 CU Emitir Informe de la validación de un submodelo.

Modelo Inclusió-Submodelo datos de la Institusión

Subvariable: Hospital

Casos	Condiciones	Derivar Vx	Validar Vx
Caso1	V1 Asignada V2 Asignada V3 Asignada	Vx=V2-V3-V1	-Campo obligatorio -Valores: E. Guevara, Frank Pais
Caso2	V1 Asignada	Asignada	Campo obligatorio

SUBVARIABLE: PROVINCIA

Casos	Condiciones	Derivar Vx	Validar Vx
Caso1	V1 Asignada V2 Asignada	Vx=V2--V1	-Campo obligatorio -Valores: Frank Pais
Caso2	V1 Asignada	Asignada	Valores: E. Guevara, Frank Pais

Subvariable: Talla

Casos	Condiciones	Derivar Vx	Validar Vx
Caso1	V1 Asignada V2 Asignada	Vx=V2-V1	Campo obligatorio
Caso2	V1 Asignada	Asignada	Campo obligatorio

Cerrar

Figura 32 CU Emitir Informe de la validación de un submodelo.

2.4 Mapa de Navegación

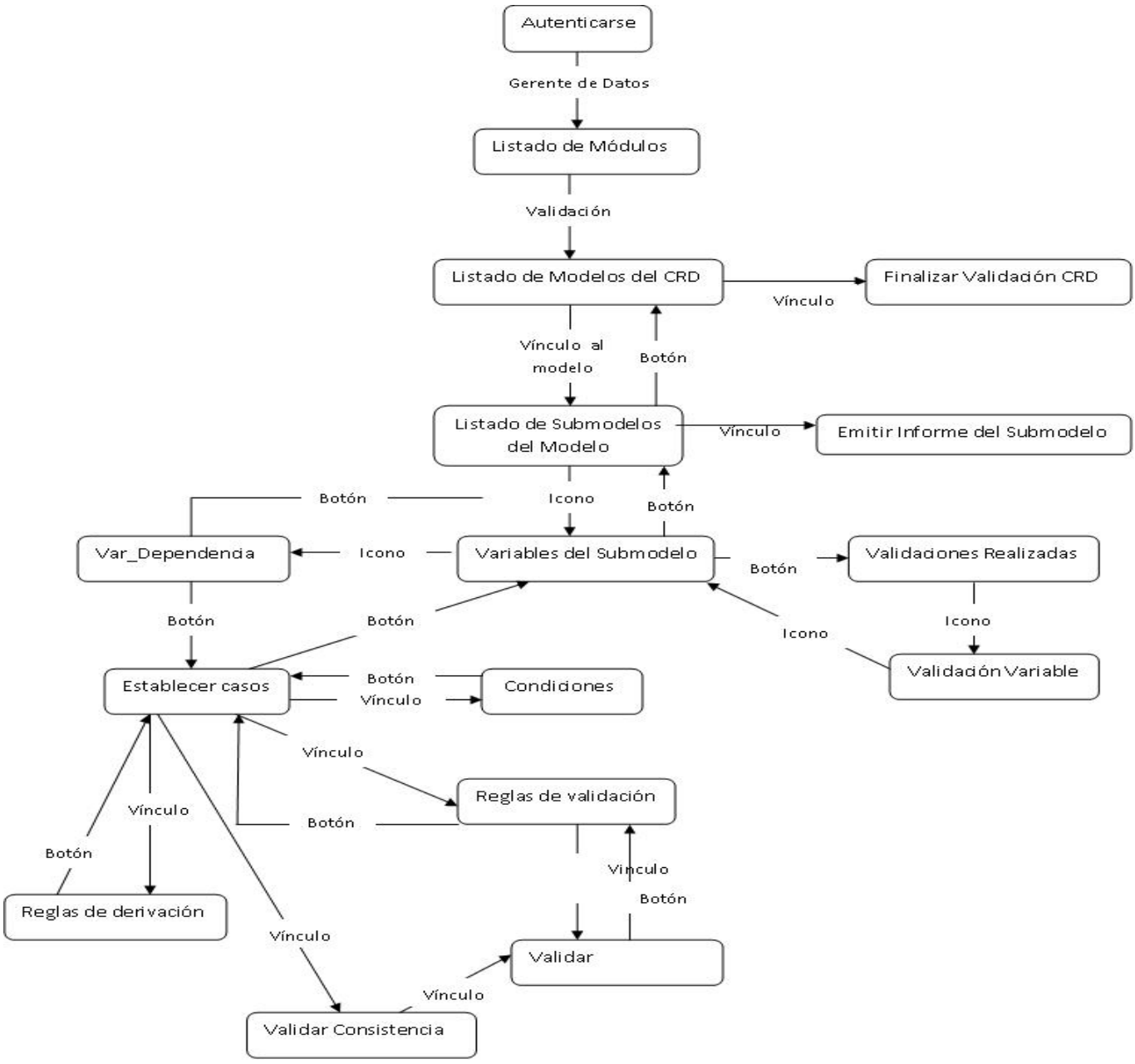

Figura 33 Mapa de navegación del módulo Validación.

2.5 Diagrama de clases persistentes

El diagrama de clases persistentes se utiliza para modelar la estructura lógica de la BD, dichas clases son aquellas que representan tablas y atributos que luego formarán parte de la base de datos. Para el análisis del diagrama de clases persistente del proyecto SIMDECC, remitirse al Expediente de Proyecto.

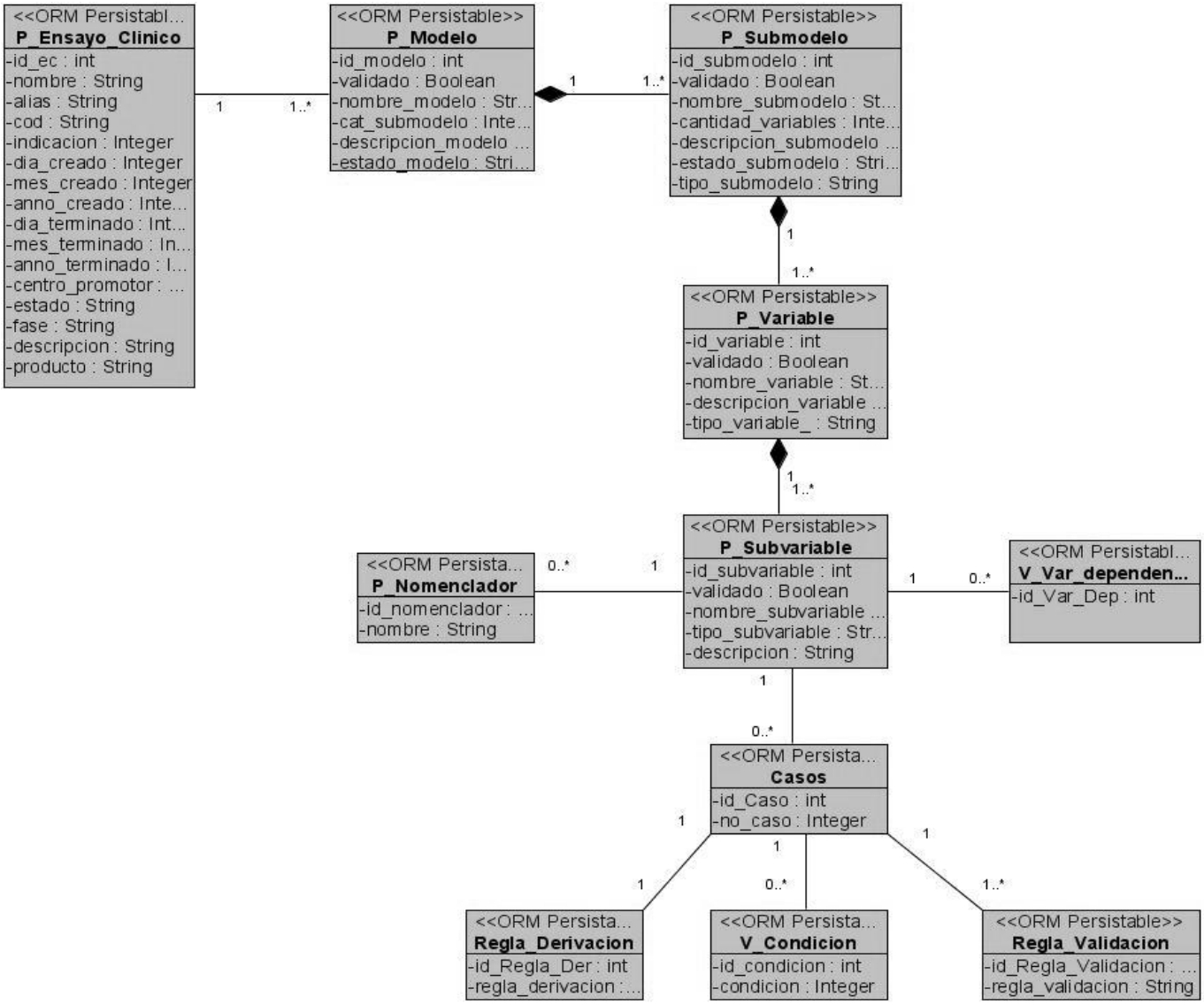

Figura 34 Diagrama de Clases Persistentes.

2.6 Modelo físico

Un modelo de datos es un sistema formal y abstracto que permite describir y manipular los datos de acuerdo con reglas y convenios predefinidos.

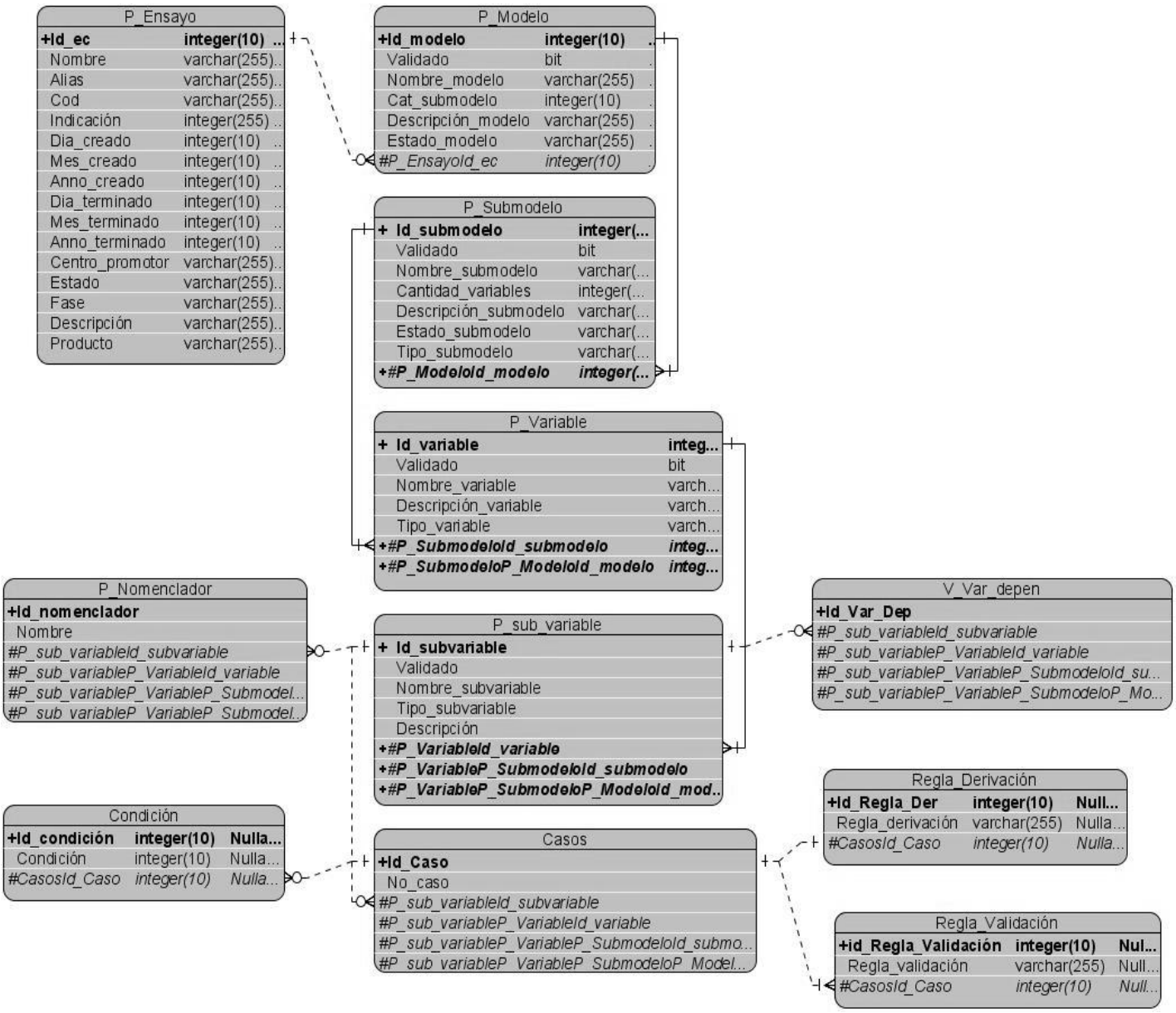

Figura 35 Modelo físico.

2.7 Diagrama de Despliegue

La aplicación Web SIMDEC 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 las 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 ya que los ensayos clínicos realizados deben almacenarse por 15 años.

La aplicación principal estará en el CIM.

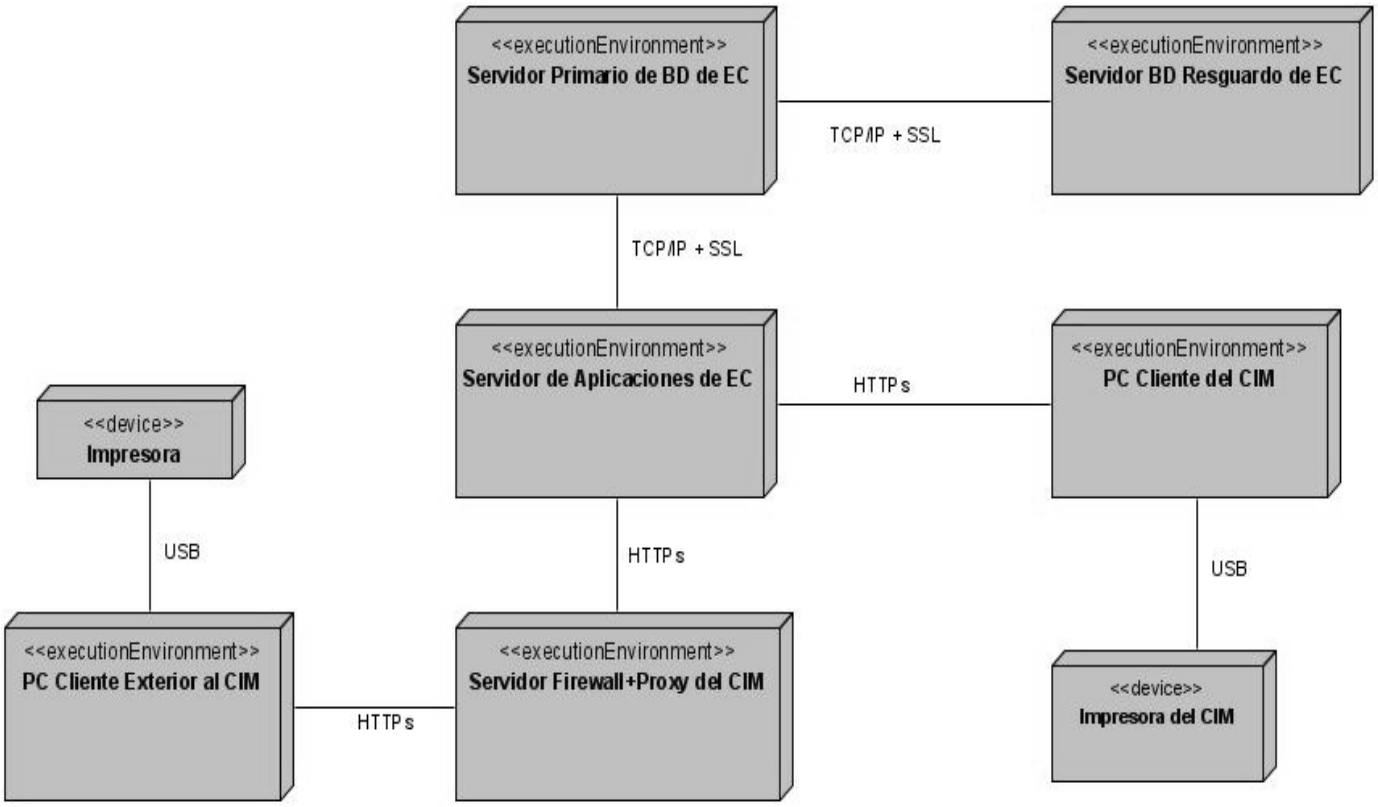

Figura 36 Diagrama de Despliegue.

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

- En el nodo Servidor de Aplicaciones del Ensayos Clínicos se ubican 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 estará la Base de datos y salvas, respectivamente (no forman parte de las capas lógicas, se presentan para una mayor claridad). [18]

2.8 Validación del Diseño

Para que una aplicación se implemente de forma correcta se debe obtener un buen diseño mediante el cual los programadores entiendan la lógica que requiere el sistema.

Para validar el diseño de la presente investigación se realizaron entrevistas a los futuros programadores del submódulo, los que plantean comprender el Modelo de Diseño dígase diagramas de clases del diseño y diagramas de interacción. Afirman poder implementar el sistema a partir de los resultados de esta investigación pues los prototipos no funcionales, los diagramas de clase del diseño y los diagramas de secuencia tienen total correspondencia con la descripción de los casos de uso del sistema y con los requisitos funcionales. Además estos programadores tienen experiencia en el trabajo con el framework, ya que han realizado implementaciones en el proyecto.

De esta forma se demostró que la solución propuesta en esta investigación servirá cómo punto de partida para la futura implementación.

Opinión de los futuros implementadores del submódulo.

Destacar que el diseño realizado por las estudiantes, es muy bueno, dicho diseño puede ser de gran ayuda para la implementación del módulo Validación, ya que aporta nuevas ideas que dan un nuevo punto de vista para los programadores y que a su vez pueden ser mas fáciles para de implementar.

Leonelbys Herrera Pérez.

Como futuro programador del módulo validación pienso que el diseño esta apto para una posterior implementación. Cuenta con un buen 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. No quedando ningún tipo de duda con respecto a este diseño para la realización de mi futuro trabajo como implementador.

Leandro Evangelio Hernández Cuello.

2.9 Conclusiones

En el presente capítulo se obtienen las clases del diseño, organizadas por capas utilizando el patrón de arquitectura Modelo Vista Controlador y los patrones de diseño, cumpliendo con la arquitectura del proyecto y con los patrones que utiliza el framework Symfony. A partir de estas clases se diseñan los paquetes del diseño, además se obtienen los diagramas de interacción donde se muestra todo el flujo de mensajes entre las clases que conforman los diagramas de clases del diseño. Además se obtuvieron los prototipos no funcionales y el mapa de navegación, dando ejemplo al cliente de cómo quedará la futura aplicación informática SIMDECC. Se muestra el diagrama de despliegue de la aplicación, facilitando así el trabajo a los futuros programadores.

Conclusiones Generales

En la investigación se presentó el diseño del submódulo “Validación de las variables del Cuaderno de Recogida de Datos y el control de errores”, para el mismo se realizó:

- El diseño de los prototipos no funcionales del submódulo, permitiendo una mejor comunicación con el cliente, logrando una mayor comprensión del sistema por parte de este.
- Se obtuvieron los artefactos correspondientes a la etapa de diseño los que:
  - Se ajustan a los principios y patrones del diseño orientado a objetos.
  - Proporcionarán la comunicación entre los diseñadores y el equipo de desarrollo, siendo de gran utilidad para la futura implementación de la solución propuesta.

La investigación cumplió su objetivo, por tanto, se concluye que contribuirá a la disminución de los errores en los CRD, propiciando que los Ensayos Clínicos realizados en el CIM tengan la calidad y confiabilidad requerida para poder aplicar estos estudios y así combatir todas las enfermedades que afectan la salud del hombre.

Recomendaciones

Luego de haber concluido la presente investigación se recomienda:

- Utilizar la solución propuesta para desarrollar la implementación del submódulo “Validación de las variables del Cuaderno de Recogida de Datos y el control de errores“, para que los Ensayos Clínicos que se realicen tengan mejor calidad.
- Utilizar la solución propuesta en otros sistemas que necesiten una herramienta para la validación.

Referencias bibliográficas

1. **López Díaz, Aidacelys y Rodríguez García, Lucía.** *Sistema de Manejo de Datos de Ensayos Clínicos.* Ciudad Habana : s.n., 2007.
2. CRD http://www.uv.es/~docmed/documed/documed/591.html 13/12/2007
3. Software Open Clínica www.OpenClinica.org 1/01/2008.
4. Software_Macro.http://www.ehto.org/ht_projects/initial_project_description/macro.html.25/01/2008.
5. PhOSCo.com. [En línea] Agosto de 2006. http://www.phosco.com/.
6. **Mendoza Sánchez, María A.** Informatizate. *Metodologías De Desarrollo De Software.* [En línea] 7 de 6 de 2004. [Citado el: 3 de 1 de 2008.] http://www.willydev.net/InsiteCreation/v1.0/Descargas/cualmetodologia.pdf
7. Visual Paradigm.http://www.visual-paradigm.com/product/vpuml/ 2/02/2008. http://www.visual-paradigm.com/
8. UML. [En línea] http://www.uml.org. 3/02/2008. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.
9. HerramientaControlVersionesSubversion. http://www.osmosislatina.com/subversion/basico.htm 25/01/2008.
10. Kompozer.com http://www.alcancelibre.org/index.php?topic=aldesktop&page=8 28/01/2008
11. **Potencier, Fabien y Zaninotto, François**. Symfony, la guía definitiva. 2007.
12. **Jesús Vegas.** Desarrollo de Aplicaciones Web. [En línea] 21 de 3 de 2008*]*. [Citado el: 3 de 4 de 2008.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html
13. Ciberaula. [En línea] 2006. [Citado el: 5 de marzo de 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro/
14. **Larman, Craig.** UML y Patrones:Introducción al análisis y diseño orientado a objetos.(2). La Habana : Félix Varela, 2004.
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. http://dei.inf.uc3m.es/docencia/p_s_ciclo/tdp/curso0203/apuntes/decorator.pdf
17. Entorno Virtual de Aprendizaje. *Material de Apoyo Conferencia Diseño*. [En línea] 22 de febrero de 2008. [Citado el: 28 de marzo de 2008.] http://teleformacion.uci.cu/mod/resource/view.php?id=21363.
18. **Ballester Marsal, Andrés**. *Documento de Arquitectura de Software(Ensayos Clínicos)*. Ciudad Habana : s.n., 2008.

Bibliografía Consultada

1. **López Díaz, Aidacelys y Rodríguez García, Lucía.** *Sistema de Manejo de Datos de Ensayos Clínicos.* Ciudad Habana : s.n., 2007.
2. Centro de Información Cardiovascular[En línea] 2007 .http://www.texasheartinstitute.org/HIC/Topics_Esp/FAQ/clinical_trials_span.cfm
3. CRD http://www.uv.es/~docmed/documed/documed/591.html 13/12/2007
4. Software Open Clínica http://www.OpenClinica.org 1/01/2008.
5. Software_Macro.http://www.ehto.org/ht_projects/initial_project_description/macro.html.25/01/2008.
6. Conallen, Jim. Building Web Applications with UML. [En línea] 2002.
7. PhOSCo.com. [En línea] Agosto de 2006. http://www.phosco.com/.
8. **Mendoza Sánchez, María A.** Informatizate. *Metodologías De Desarrollo De Software.* [En línea] 7 de 6 de 2004. [Citado el: 3 de 1 de 2008.] http://www.willydev.net/InsiteCreation/v1.0/Descargas/cualmetodologia.pdf
9. Visual Paradigm.http://www.visual-paradigm.com/product/vpuml/ 2/02/2008 http://www.visual-paradigm.com/
10. Hernández, José Alberto. [En línea] 2005. [Citado el: 14 de Diciembre de 2007.] Versión Cero. http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
11. UML. [En línea] http://www.uml.org. 3/02/2008. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.
12. HerramientaControlVersionesSubversion. http://www.osmosislatina.com/subversion/basico.htm 25/01/2008.
13. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. [En línea] La Habana, Cuba.
14. Kompozer.com http://www.alcancelibre.org/index.php?topic=aldesktop&page=8 28/01/2008.
15. **Potencier, Fabien y Zaninotto, François**. Symfony, la guía definitiva. 2007.
16. **Jesús Vegas.** Desarrollo de Aplicaciones Web. [En línea] 21 de 3 de 2008*]*. [Citado el: 3 de 4 de 2008.]

http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html

17. Ciberaula. [En línea] 2006. [Citado el: 5 de marzo de 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro/
18. **Larman, Craig.** UML y Patrones:Introducción al análisis y diseño orientado a objetos.(2). La Habana : Félix Varela, 2004.
19. **Rodríguez Luque, Daniel y Rodríguez Luque, David.** *Herramienta para la generación de código de aplicaciones Web.* Ciudad Habana : s.n., 2007.
20. **Dodero, Juan Manuel y Fernández Llamas, Camino**. *Patrones estructurales: Decorator.* Madrid : s.n., 2002.
    http://dei.inf.uc3m.es/docencia/p_s_ciclo/tdp/curso0203/apuntes/decorator.pdf
21. Entorno Virtual de Aprendizaje. *Material de Apoyo Conferencia Diseño*. [En línea] 22 de febrero de 2008. [Citado el: 28 de marzo de 2008.] http://teleformacion.uci.cu/mod/resource/view.php?id=21363.
22. Conferencia 1 IS2 Flujo de trabajo Análisis y Diseño. [En línea] 2005-2006. Universidad de las Ciencias Informáticas. La Habana, Cuba.
23. **Ballester Marsal, Andrés**. *Documento de Arquitectura de Software(Ensayos Clínicos)*. Ciudad Habana : s.n., 2008.
24. Conferencia 1 IS2 Flujo de trabajo Análisis y Diseño. [En línea] 2005-2006. Universidad de las Ciencias Informáticas. La Habana, Cuba.
25. Oktaba, Hanna. Introducción a Patrones. [En línea] Facultad de Ciencias, UNAM. http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html
26. **García, Luis**. Sistema de control de versiones: SUBVERSION . [En línea] Enero 17, 2008. http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=548.

Anexos

Anexo 1. Requisitos no funcionales

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 autentificación 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 computadora con capacidad de almacenamiento suficiente hará soportar la información de 15 años de Ensayos Clínicos.

Soporte

Descripción del 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 del 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 Kompozer.
- 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.

Anexo 2. Diagrama de CU del Sistema

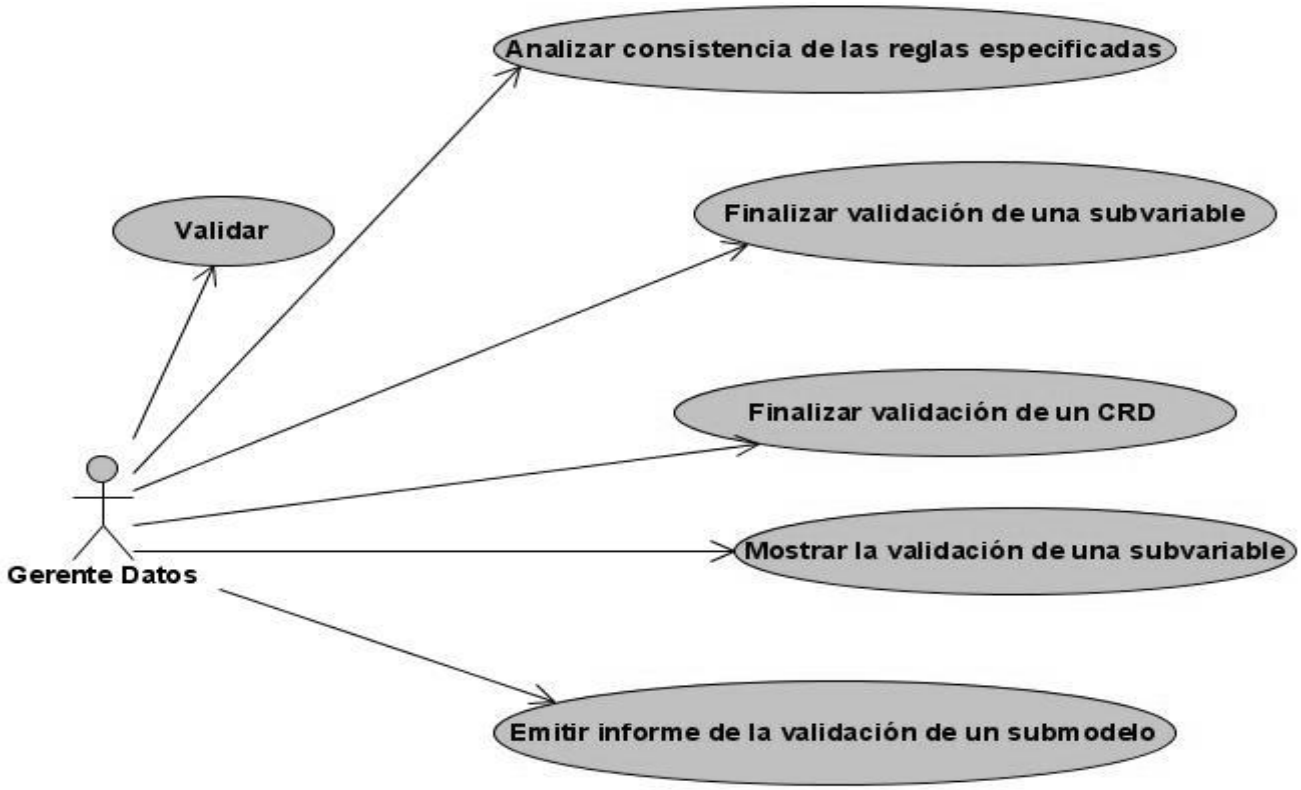

Figura 36 Diagrama de CU del sistema.

Anexo 3. Descripción de los CU del sistema

Descripción del CUS: Validar.

Nombre del CU	Validar	
Actores	Gerente de datos	
Propósito	Permitir establecer las reglas de validación para una subvariable en un caso.	
Resumen	El caso de uso se inicia cuando el Gerente de datos decide establecer las reglas de validación de la subvariable en un caso determinado. El sistema muestra las reglas de validación de acuerdo al tipo de la subvariable, y el Gerente selecciona las reglas que desea.	
Referencias		
Precondiciones	Deben estar definidas las reglas de derivación para el caso que se esta analizando.	
Curso Normal de los Eventos		
Acciones del Actor		**Respuesta del Sistema**
1. El Gerente de datos selecciona (en la tabla de estado) la opción de establecer las reglas de validación de una subvariable en un caso determinado. 2. El Gerente de datos selecciona las reglas de validación deseadas.		1.1 El sistema muestra las reglas de validación Campo obligatorio, Unidades de medida, Número de cifras enteras, Número de cifras decimales, Número de caracteres, Fecha y Rango, según el tipo de la subvariable. El Gerente de datos podrá escoger más de una regla de validación 2.1 El sistema guarda las reglas de validación seleccionadas. 2.2 El sistema actualiza la tabla de estado especificando en la fila del caso que se esta analizando, en la casilla *Validar,* las reglas de validación escogidas, activa el botón Analizar Consistencia y finaliza el caso de uso.
Poscondiciones		Quedan establecidas las reglas de validación de la subvariable.
Prioridad:	Crítico	

Descripción del CUS: Analizar consistencia de las reglas especificadas.

Nombre del CU	Analizar consistencia de las reglas especificadas.
Actores	Gerente de datos.
Propósito	Permitir que las reglas de validación hayan sido especificadas de forma correcta.
Resumen	El caso de uso se inicia cuando el gerente de datos decide analizar la consistencia de las reglas especificadas. El sistema analiza las reglas y muestra un mensaje indicando si estas han sido especificadas de forma correcta.
Referencias	
Precondiciones	Deben estar establecidas las reglas de validación de la subvariable en cuestión.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El gerente de datos decide analizar la consistencia de las reglas de validación de una subvariable.	1.1 El sistema analiza la consistencia de los casos de validación especificados, es decir, analiza que las reglas de validación no se solapen y no haya ningún error en ellas.
	1.2 Si hay algún error, el sistema muestra un mensaje indicando que las reglas tienen inconsistencias.
Poscondiciones	Queda finalizado el análisis de las consistencias.

Curso Alternativo de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1.2 Si no hay error el sistema muestra un mensaje indicando que las reglas están correctas, habilita el botón Finalizar validación de la subvariable y finaliza el caso de uso.

Descripción del CUS: Finalizar validación de una subvariable.

Nombre del CU	Finalizar validación de una subvariable
Actores	Gerente de datos
Propósito	Permitir que el Gerente de datos finalice la validación de una subvariable.
Resumen	El caso de uso se inicia cuando se han establecido las reglas de validación de la subvariable.
Referencias	
Precondiciones	Deben estar analizadas las consistencias entre las reglas de validación de la subvariable.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Gerente de datos decide finalizar la validación de una subvariable	1.1 El sistema cambia el estado de la subvariable y muestra un icono al lado de la misma indicando que ya se ha finalizado la validación de la subvariable. Finaliza el caso de uso.
Poscondiciones	Queda finalizada la validación de una subvariable.

Descripción del CUS: Mostrar la validación de una subvariable.

Nombre del CU	Mostrar la validación de una subvariable
Actores	Gerente de datos
Propósito	Mostrar la validación de una subvariable
Resumen	El caso de uso inicia cuando el gerente de datos decide ver la validación de una subvariable
Referencias	
Precondiciones	Deben estar establecidas las reglas de validación de la subvariable en cuestión.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Gerente de datos decide ver la validación de una subvariable.	1.1 El sistema muestra la tabla de estado de la subvariable indicando las variables de dependencia, los casos, las condiciones y las reglas que esta presenta. Finaliza el caso de uso.
Poscondiciones	Se muestra la validación de la subvariable.

Descripción del CUS: Finalizar validación de un CRD.

Nombre del CU	Finalizar validación de un CRD
Actores	Gerente de datos
Propósito	Permitir que el Gerente de datos finalice la validación de un CRD.
Resumen	El caso de uso se inicia cuando se han validado todas las subvariables de cada submodelo del diseño y el gerente de datos decide finalizar la validación. El sistema da la posibilidad de finalizar la validación.
Referencias	
Precondiciones	Deben estar finalizadas las validaciones de todas las subvariables de cada submodelo.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Gerente de datos decide finalizar la validación de todas las subvariables de un ensayo luego de haber establecido todas las reglas de validación para cada una.	1.1 El sistema verifica que hayan sido finalizadas las validaciones de todas las subvariables. 1.2 Si todas las subvariables han sido finalizadas, el sistema muestra un icono al lado de cada modelo indicando que ya se ha finalizado la validación de todas las subvariables de dichos modelos. Finaliza el caso de uso.
Poscondiciones	Queda finalizada la validación del CRD.
Curso Alternativo de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1.2 Si al menos una subvariable no ha sido validada, el sistema muestra un mensaje indicando que deben ser finalizadas todas las validaciones y finaliza el caso de uso.

Descripción del CUS: Emitir informe de la validación de un submodelo.

Nombre del CU	Emitir informe de la validación de un submodelo	
Actores	Gerente de datos	
Propósito	Emitir un informe de la validación de un submodelo.	
Resumen	El caso de uso inicia cuando el gerente de datos decide obtener un informe de la validación de un submodelo del CRD.	
Referencias		
Precondiciones	Debe estar validado el submodelo	
Curso Normal de los Eventos		
Acciones del Actor		**Respuesta del Sistema**
1. El Gerente de datos decide obtener un informe de la validación de las variables de un submodelo.		1.1 El sistema muestra la tabla de estado para cada variable del submodelo, indicando las variables de dependencia, los casos, condiciones, las reglas de derivación y las reglas de validación que esta presenta. Finaliza el Caso de uso.
Poscondiciones		Queda realizado el informe de la validación de un submodelo.

Glosario de términos

AR: Agencia Regulatoria**.**

Biomoléculas*:* son las moléculas constituyentes de los seres vivos formadas por sólo cuatro elementos que son el hidrógeno, oxígeno, carbono, y nitrógeno, que representan el 99 % de los átomos de los seres vivos.

CUS: Caso de uso del sistema.

CASE: Computer Aided Software Engineering.

CRD: Cuaderno de Recogida de Datos. Formulario diseñado para anotar las variables recogidas durante un ensayo clínico.

CIM: Centro de Inmunología Molecular.

EC: Ensayo Clínico.

InferMed: líder global del software de gestión de servicios farmacéutico y otros sectores de la investigación clínica.

Modelo: Grupo de variables que responden a un determinado examen o evaluación.

Protocolo: documento rector de un Ensayo Clínico, que contiene toda la documentación referida a dicho ensayo (reglas, pasos a seguir) y adjunto el cuaderno de recogida de dato.

Submodelo: Constituye un subconjunto del modelo.

Subvariable: parte de la variable, puede que una variable no tenga subvariables, si esto sucede la variable pasa a ser la subvariable, como por ejemplo la variable fecha tiene tres subvariables que son día, mes y año.

Variable: Cada uno de los datos que se recogen en el CRD como por ejemplo: el nombre del paciente, la edad del paciente, la provincia etc.