Universidad de las Ciencias Informáticas

Facultad 6

Título: “Sistema Informático para la Red Nacional de Genética Médica: módulo Teleconsulta versión 2.0.”

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

Autoras: Marta Alvarado Garrigó.
Nelvis Borges Fernández.

Tutora: Ing. Yadira Barroso Rodríguez.

Consultante: Dra. Estela Morales Peralta.

Ciudad de la Habana, Cuba.

Junio 2008.

“Año 50 de la Revolución”

"El verdadero objeto de la enseñanza es preparar al hombre para que pueda vivir por sí decorosamente, sin perder la gracia y generosidad del espíritu, y sin poner en peligro con su egoísmo o servidumbre la dignidad y fuerza de la patria."

José Martí.

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras 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 _____.

Nelvis Borges Fernández	Marta Alvarado Garrigó	Ing. Yadira Barroso Rodríguez
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

AGRADECIMIENTOS

Difícil es el camino para llegar a donde todos, familiares y amigos, sueñan desde nuestros inicios escolares. Ha llegado el momento de demostrar que somos capaces de enfrentarnos solos al mundo como profesionales. Al término de esta etapa de nuestras vidas, quisiéramos expresar un profundo agradecimiento a quienes con su ayuda, apoyo y comprensión nos alentaron a lograr esta hermosa realidad.

Antes que a nadie tenemos que agradecer de forma muy especial a nuestro gran amigo de siempre: Maikel Muñoz Roja, por su apoyo incondicional, su paciencia, sacrificio, por no cansarse nunca, bridándonos siempre su ayuda durante los 5 años de estudio en la universidad. Sin su ayuda no hubiera sido posible la culminación de este nuestro sueño. Gracias por tu dedicación. De todo corazón gracias.

Nos gustaría agradecértelo de todo corazón, pero para tí, querido amigo, nuestro corazón no tiene fondo.

A la Revolución por habernos dado la oportunidad de prepararnos como ciudadanas útiles para retribuirle a la patria lo que ha hecho por nosotras, por educarnos en el concepto martiano de que "Patria es Humanidad", formándonos para servir a nuestro pueblo y a otros pueblos hermanos que lo necesiten. Por habernos guiado por el camino de los que fundan y aman.

A nuestra tutora Yadira Barroso Rodríguez por su paciencia y ayuda oportuna siempre que la necesitamos, porque gracias a su apoyo y consejo hemos llegado a realizar la más grande de nuestras metas.

A la doctora Estela Morales por sus explicaciones y recomendaciones.

A nuestros padres que sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarnos y educarnos.

A nuestro líder de proyecto Alfonso por su dedicación e interés en conseguir que este proyecto funcione y por prestarnos su apoyo desde el primer momento.

Agradecer a Jorge Bedoya Rusenko por su ayuda brindada en el desarrollo del proyecto, a Ramsés por sus explicaciones de Symfony, en fin a todos los compañeros y profesores del proyecto que siempre estuvieron dispuestos a aclarar cualquier duda.

A nuestras amistades de la vocacional, que la queremos mucho y no las olvidamos, entre ellas están: Ariadna, Rosayda, Janet, Aimee, Yasumari, Laura y Elisa.

Gracias a todos aquellos que nos brindaron su amistad y nos ayudaron a lo largo de la carrera, a todos nuestros compañeros (as) de aula, de apartamento, que nos apoyaron y hicieron más felices nuestras vidas durante estos 5 años de carrera, gracias.

DEDICATORIA

De: Marta Alvarado Garrigó

A mis padres por su comprensión y confianza, por su ejemplo de superación constante, por su amor y amistad.

A Carlín y Amarilis por su preocupación ante mis problemas.

A mis abuelos por el cariño y los conocimientos que me trasmitieron durante la niñez y por ser ejemplo de incondicionalidad.

A mis tíos y primos por brindarme todo su cariño cuando estamos juntos y por tenerme siempre presente.

A mis amigas Nelvis, Anita, Claudia y Leidy, por su apoyo en los momentos difíciles y por hacer de estos años en la universidad los más inolvidables de mi vida.

A mi novio Yosuán por ser de esa clase de personas que todo lo comprenden y dan lo mejor de sí sin esperar nada a cambio.

De: Nelvis Borges Fernández

A las personas más importantes de mi vida, quienes me han permitido llegar hasta aquí y ser lo que hoy soy: a mis queridos padres, que son el centro de mi vida, por permitirme nacer y escalar hasta el lugar donde me encuentro hoy. Gracias por todo su amor, su apoyo, su tiempo, su cariño, comprensión; gracias por todo el sacrificio que hicieron para darme una buena educación y por ser un gran ejemplo para mi, gran parte de mi triunfo se lo debo a ustedes:

A mi madre, por ser la amiga que siempre ha estado presente para darme el consejo adecuado, se que hoy cumplo con unos de sus mayores anhelos.

En especial con todo mi corazón a mi padre por ser la estrella que ha iluminado mi camino, por ser la fuerza que me hace levantar la cabeza y continuar adelante, aun en tiempos de debilidad.

A mi familia sin la cual no habría sido posible convertir en realidad este sueño, en especial a mi tía Norma por el apoyo incondicional y a todas las personas que estuvieron presentes en los momentos más importantes. A aquellos que no son mi familia de sangre, pero que saben que lo llevo en mi corazón y que los quiero mucho, gracias por existir y siempre estar pendiente de mí. En mi retina permanecerán siempre estos instantes.

No puedo dejar de sentir tristeza por no haber tenido el placer de compartir esta bonita experiencia con personas que ya no están a mi lado, en especial mi abuelita que aunque ya no está físicamente no sale de mi mente y de mi corazón.

Quiero dedicarle a una persona que un día me dió alas para volar. Agradecerle la confianza que depositó en mí, el gran cariño que siempre me demostró, los buenos consejos que me ofreció. Esa persona eres tú, Maikel. Se que sin tí, mis expectativas no se hubiesen cumplido, gracias por darme inteligencia a lo largo de toda mi carrera y para desarrollar esta tesis, gracias por tu comprensión y apoyo incondicional, por darme tu gozo y tu paz, sin tí definitivamente no lo hubiera logrado.

A mi amiga del alma Marta Alvarado Garrigó, que la quiero mucho y estoy muy orgullosa de haber compartido toda las etapas de la universidad a su lado y sobre todo que haya sido mi dúo de tesis, no se que hubiera sido mi vida en la universidad sin tí, quiero que sepas que nunca te olvidaré, y eres y serás muy importante en mi vida.

A mis amigos; Marta, Leidy, Anita, Daniela, Maikel y Herlyn, ¿qué puedo decirles?: los amo con todo mi corazón. Definitivamente este tiempo en la universidad no habría sido lo mismo sin ustedes, no la habría pasado tan bien. Gracias por todo su amor, correos, apoyo y comprensión. Gracias por las desveladas, por todo su tiempo, sus consejos, su confianza, por los cumpleaños, por estar cuando más lo necesitaba, he aprendido mucho de cada uno de ustedes. Gracias por saber reír con el que ríe y llorar con el que llora.

En fin a todas las personas por las cuales late mi corazón. Gracias

RESUMEN

Las enfermedades genéticas, aunque en su conjunto son abundantes, individualmente son raras, lo que hace difícil su identificación y para los médicos en ocasiones no son suficientes sus conocimientos, la revisión de la literatura, ni el intercambio con otros especialistas. Por estos motivos se desarrolló la versión 1.0 de la Teleconsulta con el objetivo de acelerar y automatizar los procesos que se llevan a cabo en las consultas de referencia nacional. Sin embargo esta versión no se pudo integrar a la red nacional de salud cubana (Infomed).

El presente trabajo contribuye al diagnóstico y tratamiento de pacientes que actualmente deben ser remitidos y recorrer largas distancias, mejorando los servicios de atención a la salud y facilitando el acceso a zonas remotas de la geografía. Permite además controlar las consultas a distancia del Centro Nacional de Genética Médica (CNGM) y los diferentes centros de genética del país, de forma que se puedan responder las peticiones de toda la red nacional de genética. También se almacena toda la información referente a los pacientes de los casos a valorar en las solicitudes, discusiones e informes generados por los casos discutidos y propicia un debate a distancia que genera una solución, esta información está al instante accesible solamente para los genetistas autorizados. De este modo, la versión 2.0 de la Teleconsulta se integrará al Sistema Informático para la red nacional de Genética Médica (SIGM) para la colaboración en línea entre diferentes especialistas de la salud.

Palabras claves: teleconsulta, genética.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El futuro de la humanidad dependerá en gran medida del potencial humano y de los conocimientos que se alcancen en las próximas décadas. La informática en sus diferentes manifestaciones tiene asegurado un papel protagónico en este futuro. Hoy día con el desarrollo de la informática y las comunicaciones se está llevando a cabo en todo el mundo un proceso de transformación en disímiles esferas. Cuba, a pesar de estar sometida a un bloqueo cruel y genocida se encuentra inmersa en una lucha constante para construir una sociedad informatizada. La estrategia de informatización, como expresión del proceso revolucionario cubano, tiene al ciudadano en el centro de sus objetivos, buscando elevar su calidad de vida en diferentes esferas de la sociedad como la educación, la economía, la cultura y la salud, estando esta última estrechamente relacionada con el desarrollo de la informática del país. En el proceso de informatización de la salud cubana el mayor reto es lograr equidad y calidad en los servicios que se prestan y para ello debe alcanzarse un equilibrio entre la atención primaria de salud y el uso de las tecnologías médicas en sus distintos niveles.

El MINSAP es el organismo rector del Sistema Nacional de Salud (SNS), el cual desde el año 2003 definió como una prioridad su informatización, convocando para ello a un grupo de instituciones de este sector y del Ministerio de la Informática y las Comunicaciones (MIC), para de manera conjunta, definir los proyectos a desarrollar, tomando como punto de partida en algunos casos los sistemas ya desarrollados en el país. Ejemplos de estos centros son: DESOFT, SOFTEL, la Universidad de las Ciencias Informáticas (UCI), Infomed y las Direcciones Nacionales del Ministerio de Salud Pública implicadas directamente en los primeros productos. [1]

La UCI, como centro de estudio superior adscrito al MIC constituye una respuesta a la demanda de recursos humanos de alto nivel en el sector de las tecnologías de la información y las comunicaciones (TICs) con vistas a la realización exitosa de los proyectos de informatización de la sociedad cubana, que desde hace varios años se vienen ejecutando. En ella se desarrollan producciones de software para la salud y equipos médicos, educación, telecomunicaciones, bioinformática, entre otras; vinculadas con diferentes instituciones, como: el Centro de Ingeniería Genética y Biotecnología (CIGB), Centro de Control Estatal de Equipos Médicos (CCEEM), Centro Nacional de Investigaciones Científicas (CNIC) y el Centro Nacional de Genética Médica (CNGM), sede del Programa de la Revolución para el desarrollo de la genética médica en el país.

El 5 de agosto del 2003 fue inaugurado este centro por el Comandante en Jefe, quien señaló en el propio acto los objetivos de esta institución y su nueva concepción hacia la investigación. Dentro de sus principales funciones tiene las investigaciones básicas y aplicadas en el campo de la genética médica, la inmunología, la bioquímica y otras disciplinas afines dirigidas a la obtención de nuevos conocimientos, evaluación y desarrollo de nuevas tecnologías, productos y procedimientos de trabajo.

Las enfermedades genéticas, aunque en su conjunto son abundantes, individualmente son raras, lo que hace difícil su identificación y para los médicos en ocasiones no son suficientes sus conocimientos, la revisión de la literatura, ni los medios que empleaban para el intercambio con otros especialistas. Por estos motivos se desarrolla la versión 1.0 de la Teleconsulta con el objetivo de acelerar y automatizar los procesos que se llevan a cabo en las consultas de referencia nacional. [2]

La Teleconsulta consiste en el diagnóstico a distancia, es una técnica de gran impacto dadas las múltiples ventajas que presenta, y posee un amplio margen para la explotación de los medios tecnológicos. Consiste en evaluar un paciente desde un centro hospitalario remoto, haciendo uso de las telecomunicaciones para llevar a cabo esta acción. Permite desarrollar espacios de interacción con la creación de servicios de información, consulta y discusión que favorezcan el empleo del espacio virtual de intercambio permanente entre los especialistas involucrados. Se trata esencialmente de conformar redes humanas, con el soporte de las TICs que motiven y potencien, en forma dinámica y sistemática, la interacción entre las personas, que fortalezcan la generación, diseminación e intercambio de información y conocimiento en función de los objetivos profesionales, institucionales y sociales.

En 1992 surge la red nacional de salud con el propósito de facilitar el acceso a la información relacionada con la medicina, sin embargo la versión existente de la Teleconsulta creada en la UCI en julio del 2007, no se puede integrar a Infomed, imposibilitando la comunicación entre especialistas que se encuentran separados geográficamente por largas distancias.

Por la situación planteada anteriormente sobre el estado de la versión 1.0 de la Teleconsulta y por no poderse integrar a Infomed surge el **problema científico**: ¿Cómo obtener un producto funcional a partir de los requerimientos identificados para el módulo Teleconsulta versión 2.0 integrado a la red nacional de salud?

Teniendo como **objeto de estudio** el proceso de gestión de información de la salud, y como **campo de acción** derivado del mismo, el proceso de desarrollo de software para la gestión de la información de la teleconsulta del SIGM.

Siendo el **objetivo general** de este trabajo:
Desarrollar el diseño y la implementación del módulo Teleconsulta versión 2.0 del SIGM que permita gestionar el proceso de teleconsultas entre genetistas.
Para lograr este objetivo, se proponen los siguientes **objetivos específicos:**

- Diseñar el módulo Teleconsulta versión 2.0.
- Implementar el módulo Teleconsulta versión 2.0.

Para dar cumplimiento a estos objetivos y lograr una solución adecuada a la situación problémica especificada se plantean las siguientes **tareas investigativas:**

1. Análisis de la versión existente de la teleconsulta.
2. Análisis de la arquitectura de software establecida para el desarrollo de sistemas informáticos para el MINSAP.
3. Diseño de las clases del módulo Teleconsulta versión 2.0.
4. Implementación de los componentes del módulo Teleconsulta versión 2.0.
5. Validación de la aplicación a nivel de desarrollador.

El documento está formado por un resumen, introducción, 3 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 brinda una descripción de algunos aspectos relacionados con la teleconsulta, después de una investigación realizada. Se describe y se aborda el estado del arte, así como las tecnologías y las herramientas a utilizar. Se señalan además los patrones de diseño que se usarán y los roles que se generan producto de las necesidades actuales para el desarrollo del módulo Teleconsulta.

Capítulo2: Diseño del Sistema.

En este capítulo se exponen elementos importantes para una solución satisfactoria tales como los requisitos funcionales y no funcionales así como una breve descripción de los casos de uso del sistema. Además se realiza el diseño del sistema, obteniéndose los diagramas de clases del diseño web, los diagramas de interacción (secuencia) y el diagrama de despliegue. Igualmente se presenta la descripción de la clase tlAction y finalmente se exponen los elementos asociados a la seguridad y validación incorporados a Symfony.

Capítulo 3: Implementación del Sistema.

La implementación comienza precisamente con los resultados arrojados por el modelo de diseño, pero en esta ocasión la concepción del sistema se llevará al nivel más bajo posible, o sea al nivel de componentes. Se realiza la codificación y finalmente se realizan pruebas unitarias para asegurar la calidad del software. Como resultado de este capítulo se genera el artefacto diagrama de componentes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se brinda una descripción de algunos aspectos relacionados con la teleconsulta, después de una investigación realizada. Se describe y se aborda el estado del arte, así como las tecnologías y las herramientas a utilizar. Se señalan además los patrones de diseño que se usarán y los roles que se generan producto de las necesidades actuales para el desarrollo del módulo Teleconsulta.

1.2 Estado del arte.

Por más de 4.000 años, las personas han observado que ciertas enfermedades se presentan en determinadas familias, pero "la razón" de esta observación era desconocida hasta que la ciencia moderna mostró como se trasmite la información genética. La medicina actual reconoce que las enfermedades genéticas se heredan sobre la base de la naturaleza del ADN, los genes y los cromosomas. Ahora que se ha completado la secuencia del genoma humano, los científicos pueden estudiar mejor como los cambios en el ADN causan enfermedad en los humanos y finalmente esto ayudará en el diagnóstico y tratamiento de trastornos genéticos.

La asesoría genética y el diagnóstico prenatal brindan a los padres el conocimiento para tener la oportunidad de tomar decisiones inteligentes respecto a un posible embarazo y sus consecuencias. Algunos padres se deciden por el embarazo y hacen que el estado patológico del feto sea determinado al comienzo del embarazo, el cual continúa si el feto está libre de la enfermedad. Si se identifica la anomalía genética en el feto, los padres que deciden continuar el embarazo pueden prepararse mejor para cuidar al bebé, informándose acerca de la enfermedad con anticipación.

La Teleconsulta consiste en el intercambio de información especializada entre médicos sobre opiniones o conocimientos de un determinado tema. Las aplicaciones de teleconsulta entre lugares lejanos con transmisión multimedia de datos e imágenes son actualmente una realidad.

El objetivo es poder recibir una “segunda opinión” a través de la teleconsulta, una verdadera aplicación de telemedicina en tiempo real. [3]

1.2.1 Sistemas de teleconsultas existentes en el ámbito internacional.

En los últimos tiempos ha proliferado en el mundo el empleo de servidores para la teleconsulta, con carácter estrictamente electrónico, o combinados con consultas convencionales, a continuación se presentan algunos.

El II Congreso Virtual Hispanoamericano de Anatomía Patológica desarrollado del 1 de Junio al 31 de Julio de 1998 habilitó una **Sala de Seminarios Clínico-Patológicos** que permitió estudiar los seminarios disponibles, enviar nuevos Seminarios Clínicos-Patológicos o modificar seminarios ya enviados. Hoy sólo se pueden consultar los casos discutidos de los seminarios disponibles, conformados por la historia clínica del paciente (HC), la anatomía patológica, los diagnósticos y comentarios recibidos y una pequeña discusión del caso realizada por los médicos responsables del seminario seleccionado. [4]
Esta herramienta no está disponible para la realización de teleconsultas para el diagnóstico de pacientes en escenarios reales, quedando restringido su uso a estudios y actividades académicas, además de que está diseñado para la discusión de casos de patología exclusivamente.

El proyecto Red de Teledismorfología de diagnóstico y consejo genético. La Red de Teledismorfología de Chile es un servicio de interconsulta genética en línea gratuita, conformado por 20 genetistas del país y ofrecido a los médicos de Chile que atiendan niños con cuadros malformativos y que no tengan un diagnóstico claro del mismo. El médico que solicite la interconsulta debe llenar una ficha, la cual debe acompañar de fotografías del paciente. Toda esta información debe enviarse vía Internet por e-mail. El Comité de Teledismorfología redirecciona a su grupo de médicos genetistas clínicos de apoyo la información, recepciona sus opiniones y envía una respuesta consenso al médico interconsultante en un plazo máximo de 72 horas. [5]
Este proyecto exige que dentro de las fotos digitales que se le tomen al paciente existan algunas de cuerpo entero lo que no coincide con el código de ética definido por el CNGM que plantea que no se brindará información personal a través de la que se pueda revelar la identidad personal de los pacientes que son consultados. Esta herramienta no está disponible para la realización de teleconsultas para el diagnóstico de pacientes en escenarios reales.

IV Congreso Virtual Hispanoamericano de Anatomía Patológica, 4 de noviembre al 15 de diciembre de 2002.

En el Instituto de Patología (Institute of Pathology), Humboldt University of Berlín, Germany, se trabaja con un **Microscopio de Control Remoto** manejado a través de un navegador de Internet. Se trata de un microscopio automatizado conectado a una computadora que tiene funciones de servidor de Internet. Cualquier usuario de Internet puede acceder a este servidor y controlar el microscopio mediante un navegador de Internet que soporte Java. El sistema se acompaña de un chat para discutir las imágenes, o los pormenores del caso. En caso de duda diagnóstica, en vez de remitir las preparaciones a un experto, uno puede consultar simultáneamente la opinión de muchos, alrededor del globo. El medio es tan sencillo como transmitir las imágenes por e-mail y al mismo tiempo, conversar mediante video, audio o texto conferencia realizando un tipo de interconsulta global. [6]

Este sistema sólo se emplea para estudio de las lesiones celulares, tejidos y órganos, así como para ver sus consecuencias estructurales y funcionales y por tanto de las repercusiones en el organismo. Los expertos sólo cuentan para el diagnóstico del caso con las imágenes tomadas del paciente, lo que no es suficiente pues el mismo puede necesitar datos que estén en su historia clínica.

La Facultad de medicina de la Pontificia Universidad Católica de Chile cuenta con una serie de aplicaciones de Telemedicina. Así se realizan reuniones clínicas a distancia con el Hospital Sótero del Río, que a través de videoconferencia permiten interactuar a médicos y estudiantes que se encuentran separados por 20 kms, compartiendo casos clínicos, realizando clases. También se ha montado un sistema que permite la visualización de las imágenes de un microscopio remoto, realizándose diagnósticos a distancia. Un sistema similar, permite la visualización y diagnóstico a distancia de radiografías. Diariamente, especialistas situados en el Hospital Clínico de la UC, realizan el diagnóstico a distancia de ecografías obstétricas y ginecológicas efectuadas en el Centro de Diagnóstico, situado a 12 kms. [7]

Estas aplicaciones se limitan al empleo sólo de imágenes para el diagnóstico, lo que no es suficiente pues se pueden necesitar datos como los que están en la historia clínica del paciente. Su objetivo fundamental es de carácter docente.

1.2.2 Sistemas de teleconsultas existentes en el ámbito nacional.

En Cuba desde 1970 se han realizado experimentos en la transmisión de señales a través de teléfono o radio, con la finalidad de buscar mayor calidad del diagnóstico mediante consulta de segunda opinión, es por esto que la teleconsulta no ha pasado inadvertida. Actualmente se emplean en el país determinados sistemas con el fin de desarrollar servicios de teleconsulta entre los especialistas de la salud separados geográficamente. A continuación se presentan algunos ejemplos.

La Universidad Virtual de Salud cuenta con la sección de **Clínica Interactiva**, donde se comparten enfoques médicos novedosos y se participa en la discusión diagnóstica. La sección brinda temas periódicamente en los que se puede ser un actor más en el debate de este espacio de intercambio de conocimientos y creación. Este servicio no es un sitio para la consulta médica de pacientes; está dirigido a los profesionales de la salud y les permite consultar a un grupo de expertos en las diferentes especialidades y áreas de las Ciencias de la Salud, quienes atenderán su solicitud de información clínica especializada, con vistas a apoyar la toma de decisiones sobre temas que requieran opinión autorizada. Su objetivo principal es el desarrollo de habilidades para el diagnóstico y solución de problemas por los estudiantes.

Se brinda el servicio de pregunta al experto que está dirigido solamente a los profesionales de la Salud y les permite consultar a un grupo de expertos en las diferentes especialidades y áreas de las ciencias de la salud, los que atenderán la solicitud de información clínica especializada, que demande de un experto en el tema. Las preguntas realizadas a los expertos son con carácter docente. Como contribución a la educación y a la toma de decisiones promueve el desarrollo y la operación de instrumentos de apoyo a la educación continuada y el aprendizaje a distancia. En la sección Discusión Diagnóstica, los profesionales de la salud pueden incorporarse a las discusiones diagnósticas en línea, en las diferentes especialidades clínicas médicas. [8]

Este sistema, al igual que todos los anteriores, no permite la utilización de un modelo de solicitud personalizada, a los efectos de realizar un sistema de teleconsultas más formal, y para agilizar su desarrollo, al contar así el especialista desde un inicio, con todos los datos necesarios para la obtención del diagnóstico y el tratamiento adecuado.

Sistema de teleconsulta existente en la UCI.

En la universidad se realizó en Julio del 2007 un sistema de teleconsulta que facilita la obtención de segundas opiniones de médicos y especialistas en el proceso de diagnósticos de pacientes pues hace

posible el intercambio de imágenes médicas, habilita de forma progresiva servicios especializados de atención a la salud donde no existen y también permite ayudar a la formación de los futuros médicos.

La nueva versión del módulo Teleconsulta facilitará la mejora de los servicios de atención a la salud, proporcionará el acceso a zonas remotas de la geografía para tener la atención de los especialistas en genética; permitirá la interacción médico-médico, para lo cual resulta necesario el intercambio de imágenes y textos entre los genetistas. Mediante el uso del chat, los genetistas de la salud podrán debatir en tiempo casi-real. El CNGM cuenta con los elementos necesarios para la elaboración de un modelo de solicitud que contenga los aspectos fundamentales para que los especialistas tengan en principio el conjunto de elementos necesarios para valorar y diagnosticar cualquier caso. Se va a implementar un sistema de teleconsulta con el uso del modelo de solicitud que posee la mayoría de las situaciones, otorgando mayor formalidad al proceso de solicitud de la teleconsulta por parte de los genetistas. Es necesario que el sistema de teleconsulta alcance la mayor velocidad posible teniendo en cuenta que en muchas ocasiones las solicitudes tienen niveles elevados de urgencia médica. Permitirá además la gestión de especialistas a participar en la discusión, quedando así restringido el acceso únicamente a los especificados por las personas autorizadas a planificar las discusiones. El sistema será desarrollado sobre la arquitectura de software propuesta por el MINSAP para sus productos informáticos, y será hospedado en Infomed para estar accesible desde cualquier lugar del país.

1.3 Desarrollo del SIGM: Módulo Teleconsulta versión 2.0

Cuba a través del CNGM ha desarrollado diversos estudios asociados al crecimiento del volumen de información que generan los estudios genéticos y el descubrimiento vertiginoso del genoma humano, los cuales han aportado una gran cantidad de información, detectándose además la inexistencia de un sistema que integre la gestión integral de los elementos de una consulta de genética médica. El SIGM realizó una herramienta que permite la integración de la información de distintos estudios que hoy se realizan en Cuba. La Teleconsulta brinda al SIGM la posibilidad de intercambio a distancia de la información genética. Estos estudios llevan asociados una gran cantidad de información necesaria para definir los diagnósticos genéticos y el seguimiento, así como el tratamiento de los pacientes. Anteriormente esta información se encontraba en papel y debía ser almacenada pues no se podía prescindir de ella, ya que aún cuando el paciente fallezca su información es muy importante para la toma de decisiones locales, regionales y nacionales.

El SIGM consta de 7 módulos o registros, con el objetivo de integrar todas las funcionalidades que pretende brindar este sistema, facilitando la comunicación entre todos los genetistas del país, siendo estos: Registro Cubano de Historias Clínicas Genéticas, Registro Cubano de Enfermedades Genéticas, Registro Cubano de Malformaciones Congénitas, Registro Cubano de Discapacitados, Registro Cubano de Retraso Mental, Registro Cubano de Gemelo y el módulo Teleconsulta para la discusión a distancia de casos de los que no se tiene un diagnóstico certero, siendo este último el tema a tratar en el presente trabajo de diploma.

1.4 Tecnologías de desarrollo y herramientas a utilizar.

Para el desarrollo del módulo Teleconsulta se necesita de la implementación de un sistema informático que gestione el establecimiento de la comunicación y la transmisión de datos, por ser la red el medio más idóneo para la transmisión de la información en el caso que ocupa este trabajo.

Cuando la navegación de un usuario a través de los recursos colocados en la web tiene implicaciones para los negocios respectivos, se dice que se está en presencia de una aplicación web. [9] Estas aplicaciones no necesitan ser descargadas, instaladas y configuradas, pudiendo ser accedidas sin importar la configuración o el hardware del cliente. Actualizar o hacer cambios en el software es sencillo y sin riesgos de incompatibilidades por existir sólo una versión en el servidor lo que implica que no hay que distribuirla entre las demás computadoras, haciéndose el proceso de cambio rápido y limpio. Se facilita el trabajo a distancia permitiendo que se trabaje desde cualquier PC o computadora portátil con conexión a Internet. Además pueden ser utilizadas por múltiples usuarios al mismo tiempo, y al funcionar en un navegador se requiere un conocimiento básico de informática para utilizarla. Por lo que a continuación se explican diferentes tipos de herramientas, metodologías y tecnologías de desarrollo necesarias para darle solución a los objetivos propuestos, teniendo en cuenta que estas sean factibles en la red nacional de salud cubana.

El CNGM cuenta con una red conformada por los centros de todos los municipios y provincias del país, que de manera coordinada conducen el Programa Nacional para el diagnóstico, manejo y prevención de enfermedades genéticas en Cuba. Dicha red está disponible en Infomed, que entre sus políticas de desarrollo tiene el uso de la plataforma LAMP, la cual hace referencia a Linux como sistema operativo,

Apache como Servidor Web, MySQL como sistema gestor de bases de datos y PHP como lenguaje de programación, de ahí que el módulo Teleconsulta se integre a Infomed, para su correcto desempeño.

1.4.1 Metodología de desarrollo de software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos informáticos. Consiste en ir llevando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además las personas que deben participar en el desarrollo de las actividades y el papel que deben tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. En la actualidad se encuentran entre las más usadas: Rational Unified Process (RUP) [10], que será la utilizada en este trabajo de diploma.

Rational Unified Process: RUP.

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- **Inicio:** Su objetivo es determinar la visión del proyecto.
- **Elaboración:** Su objetivo es determinar la arquitectura óptima.
- **Construcción:** El objetivo es llegar a obtener la capacidad operacional inicial.
- **Transición:** El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el 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.
Las actividades en RUP se han agrupado en 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los 3 últimos como de apoyo. [11]

Existen tres características importantes en RUP, y que lo convierten en único, ellas son:

- Dirigido por los casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

1.4.2 Lenguaje Unificado de Modelado: UML.

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. 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. Es además expresivo, claro y uniforme, que no garantiza el éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que han existido muchas notaciones y métodos, usados para el diseño orientado a objetos, ahora los diseñadores sólo tienen que aprender una única notación. UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. [11]

1.4.3 Herramientas CASE: Visual Paradigm 6.1.

Es una herramienta CASE que utiliza “UML” como lenguaje de modelado. Está orientada a la creación de diseños usando el paradigma de programación orientada a objetos (POO).

Visual Paradigm para UML es un galardonado producto que facilita las organizaciones del diagrama visual y de diseño, integrar y desplegar sus aplicaciones empresariales de misión crítica y de sus bases de datos subyacentes. Es una poderosa herramienta para visualizar y diseñar elementos de software, utilizando el lenguaje UML. [12]

1.4.4 Gestor de Base de Datos: MySQL 5.

MySQL es un sistema de gestión de base de datos (BD) relacional, multihilo y multiusuario con más de seis millones de instalaciones. Es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Su popularidad como aplicación web está muy ligada a PHP,

que a menudo aparece en combinación con MySQL. Es una BD muy rápida en la lectura cuando utiliza el motor no transaccional MySAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación.

MySQL 5 añade nuevas características que lo hacen más atractivo. Las más interesantes son: los procedimientos almacenados, los desencadenadores (triggers) y las vistas. [13]

- Soporte multiplataforma.
- Triggers, Cursors.
- Vistas actualizables.
- Soporte a VARCHAR.
- En MySQL 5, los clientes y servidores Windows se pueden conectar usando memoria compartida.

1.4.5 Servidor Web: Apache 2.0.

El servidor HTTP Apache es un software libre de código abierto. Apache 2.0 presenta entre otras características mensajes de error altamente configurables, BD de autenticación y negociado de contenido, pero ha sido criticado por la falta de una interfaz gráfica que ayude en su configuración. La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas puede en la mayoría de los casos ser abusada solamente por los usuarios locales y no puede ser accionada remotamente. Sin embargo, algunas de las ediciones ante dichas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache. La arquitectura del servidor Apache es muy modular. Algunas características de la versión 2.0 de Apache son: [14]

- Nuevo sistema de configuración y compilación.
- Nueva interfaz de programación (API) de Apache.
- Mensajes de error en diferentes idiomas.
- Configuración simplificada.

1.4.6 Entorno de desarrollo: Eclipse.

Eclipse es un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) para todo tipo de aplicaciones. La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura

formada por un núcleo y muchos plugins que van conformando la funcionalidad final. La forma en que los plugins interactúan es mediante interfaces o puntos de extensión; así, los nuevos aportes se integran sin dificultad ni conflictos.

Se utilizó como interfaz de desarrollo web Quanta Plus para el diseño de las páginas web. Se seleccionó este IDE para el desarrollo de la herramienta, principalmente por:

- Permitir el desarrollo de aplicaciones utilizando Subversion como sistema de control de versiones.
- Por el potente editor de código que presenta.
- Por brindar la facilidad de agregarles las librerías de los framework a utilizar y una vez agregadas realizar completamiento de código del mismo. [15]

1.4.7 Framework: Symfony 1.0.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

En el desarrollo del presente trabajo de diploma se escogió a **Symfony** por sus características y funcionalidades, después de un estudio que realizó el grupo de arquitectura del proyecto. Symfony es uno de los frameworks PHP más populares entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos a la vez que crean código de más calidad y más fácil de mantener. Incluye varias herramientas que permiten automatizar las tareas más comunes de la ingeniería del software. Es maduro, estable, profesional y está muy bien documentado. Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para comenzar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación. Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos como MySQL, PostgreSQL, Oracle y SQL

Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, Linux, como en plataformas Windows. [16]

Características de Symfony.

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

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos (SGBD).
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.4.8 Sistema de control de versiones: Subversion 1.4.

Subversion 1.4 es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se lo conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion 1.4 es que, a diferencia de CVS, 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.
- Puede ser servido mediante Apache. Esto permite que clientes, utilicen Subversion en forma transparente.
- 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). [17]

1.4.9 Tecnologías del lado del servidor. Lenguajes de programación: PHP 5.0.

Uno de los aspectos más novedosos de PHP 5.0 es Zend Engine II, que entre otras características, presenta un modelo basado en objetos, que mejora la funcionalidad general. PHP 5.0 soporta XML, que para esta versión se ha reescrito íntegramente, también soporta MySQLi, una nueva ampliación de MySQL, la cual además de la interfaz habitual, encierra una interfaz basada en objetos. Estas son algunas de las mejoras que este incluye:

- Zend Engine II, con un nuevo modelo de objetos más avanzado y robusto que su predecesor de PHP 4.
- Soporte para XML reescrito desde cero, todas las extensiones relacionadas están escritas ahora entorno a la excelente librería libxml2.
- Nueva extensión MySQL denominada MySQLi para los desarrolladores que utilicen MySQL 4.1 y versiones posteriores. Esta extensión incluye una interfaz orientada a objetos como adición a la interfaz tradicional; así como soporte para las numerosas nuevas funciones de MySQL.
- El soporte para streams ha sido mejorado, incluyendo soporte para acceder a operaciones de bajo nivel sobre los sockets en streams. [18]

1.5 Patrones de Diseño y de Arquitectura.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. [19]

En el desarrollo del presente trabajo de diploma se evidencian algunos patrones de diseño utilizados y el patrón de arquitectura.

1.5.1 Patrones GRASP.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y aplicable. Los patrones básicos se refieren a cuestiones y aspectos fundamentales del diseño, algunos de estos patrones utilizados en este trabajo son:

- **Experto:** Este se encarga de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- **Creador:** Este patrón se encarga de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.
- **Alta Cohesión:** Este patrón se encarga de asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande.
- **Bajo Acoplamiento:** Este patrón se encarga de asignar una responsabilidad para mantener bajo acoplamiento. Las clases deben comunicarse con un número pequeño de clases tanto como sea posible.
- **Controlador:** Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. [19]

1.5.2 Patrones GoF.

Los patrones GoF se clasifican en 3 categorías:

- **Patrones de creación:** Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones normalmente serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos otro delegará responsabilidades.
- **Patrones estructurales**: Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.

- **Patrones de comportamiento**: Se utilizan para organizar, manejar y combinar comportamientos. [20]

Tipos de patrones GoF que implementa Symfony.

El framework Symfony que se utiliza para la implementación de la solución que aquí se propone, utiliza una serie de patrones GoF:

En la categoría de creacionales Symfony utiliza el patrón:

- **Singleton** (Instancia única, patrón de creación): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

En la categoría de estructurales Symfony utiliza los patrones:

- **Decorator** (Envoltorio, patrones estructurales): Responde a la necesidad de añadir dinámicamente funcionalidad a un objeto.
- **Command:** Encapsula peticiones en forma de objetos permitiendo así parametrizar los clientes utilizando distintas peticiones, encolar las peticiones y ofrecer la posibilidad de deshacer las operaciones. Permite solicitar operaciones sin tener que saber como o quien lleva a cabo esas operaciones.

Otros Patrones

- **Registry**: Es un medio simple y eficiente de compartir datos y objetos en la aplicación sin tener que preocuparse de mantener numerosos parámetros o hacer uso de variables globales. [21]

1.5.3 Patrones de Arquitectura.

Los patrones de arquitectura ayudan a especificar la estructura fundamental de una aplicación, expresan el esquema fundamental de organización para sistemas de software, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Cada patrón de arquitectura ayuda a conseguir una propiedad específica en el sistema global; por ejemplo, la adaptabilidad de la interfaz de usuario. Dentro de los patrones de arquitectura se puede encontrar el patrón Modelo Vista Controlador (MVC) y el patrón Modelo de Tres Capas.

Como ya se ha reflejado para el desarrollo de la aplicación informática que acompaña el presente trabajo de diploma se utilizará el framework Symfony, el cual está basado en el patrón MVC, que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el sistema de gestión de base de datos y el controlador representa la lógica del negocio, que está formado 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.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. [22]

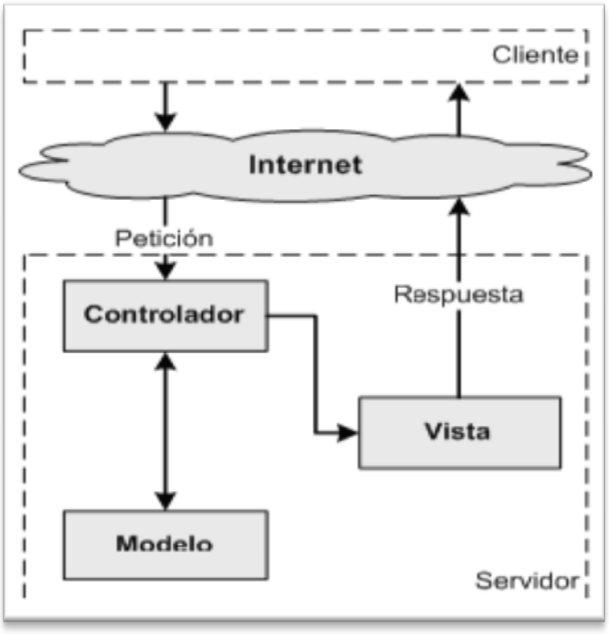


Figura 1. 1: Modelo Vista Controlador

1.6 Roles y artefactos.

Un rol es una definición abstracta de un conjunto de actividades realizadas y de artefactos obtenidos. Los roles son realizados típicamente por un individuo, o un conjunto de individuos, trabajando juntos en equipo. RUP define grupos de roles, agrupados por participación en actividades relacionadas. Estos grupos son: analistas, desarrolladores, gestores, apoyo, especialistas en pruebas y otros. [23]
En el presente trabajo de diploma se desarrollarán los roles de diseñador e implementador pertenecientes al grupo de roles definido por RUP.

1.6.1 Rol Diseñador.

En la metodología RUP, el diseñador es el responsable de diseñar una parte del sistema cumpliendo con las restricciones de los requerimientos, arquitectura y proceso de desarrollo del proyecto, identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño. Debe asegurarse que el diseño es consistente con la arquitectura del software y que está detallado al punto que se puede proceder con la implementación.
En el presente trabajo de diploma los artefactos que generará el diseñador son:

- **Realización de casos de uso del diseño:** Es una colaboración en el modelo de diseño que describe como se realiza un caso de uso específico, y como se ejecuta en términos de casos de uso del diseño. Una realización de caso de uso del diseño proporciona una traza directa a una realización de caso de uso del análisis en el modelo de análisis.

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

- **Clases del diseño:** Una clase es una descripción de un conjunto de objetos que comparten las mismas responsabilidades, las relaciones, las operaciones, atributos, y la semántica.

- **Subsistema de diseño:** Es una parte del sistema que encapsula el comportamiento, incluye interfaces y paquetes. [24]

1.6.2 Rol Implementador.

En la metodología RUP, el rol Implementador es responsable de desarrollar y de probar componentes de acuerdo con los estándares adoptados del proyecto para la integración en subsistemas más grandes. Cuando los componentes de prueba, tales como drivers o partes se deben crear para apoyar la prueba, el implementador es también responsable de desarrollar y de probar los componentes de prueba y los subsistemas correspondientes.

En el presente trabajo de diploma el artefacto que generará el implementador es:

- **Elementos de implementación:** Son la parte física de la implementación, incluyen los archivos y directorios. Incluyen ficheros de código (fuentes, binarios o ejecutables), ficheros de datos y de documentación como ficheros de ayuda online. [24]

1.7 Conclusiones.

De acuerdo a lo antes analizado se llegó a la conclusión de que la versión existente de la Teleconsulta no se puede integrar a la red de salud de Cuba, provocando que la misma no pueda ser utilizada por todos lo genetistas del país. Se realizó el análisis del estado del arte, lo que permitió dejar definido la posición de los autores en cuanto a la necesidad de crear una nueva versión de la Teleconsulta. Para obtener un buen desarrollo de esta nueva versión se realizó una fundamentación de la metodología, técnicas y herramientas a utilizar, todo esto con el objetivo de garantizar la adecuada prestación de los servicios de Teleconsulta por el CNGM hacia todos los centros de genética del país.

CAPÍTULO 2: DISEÑO DEL SISTEMA

2.1 Introducción.

En este capítulo se exponen elementos importantes para una solución satisfactoria tales como los requisitos funcionales y no funcionales así como una breve descripción de los casos de uso del sistema. Además se realiza el diseño del sistema, obteniéndose los diagramas de clases del diseño web, los diagramas de interacción (secuencia) y el diagrama de despliegue. Igualmente se presentan las descripciones de las clases y finalmente se exponen los elementos asociados a la seguridad y validación incorporados a Symfony.

2.2 Características del Sistema.

Requisitos Funcionales del Sistema.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Estos no alteran la funcionalidad del producto, es decir que se mantienen invariables sin importarle con cuales propiedades o cualidades se relacionen. [25]
Para dar cumplimiento a las funcionalidades del sistema se detectaron 28 requerimientos funcionales, a continuación se listan los mismos:

R1. Mostrar código de ética.
R2. Introducir datos de la solicitud.
R3. Buscar historia clínica.
R4. Mostrar datos primarios de un paciente seleccionado.
R5. Mostrar listado de las solicitudes en espera de aprobación organizadas por el nivel de urgencia y la fecha de arribo.
R6. Aprobar solicitudes.
R7. Denegar solicitudes.
R8. Enviar mensaje notificando al solicitante que la solicitud fue denegada y su explicación.
R9. Mostrar listado de casos a discutir planificados con su fecha y hora de realización.
R10. Mostrar listado de los casos aprobados sin planificar organizados por el nivel de urgencia y la fecha de arribo.

R11. Adicionar participantes.
R12. Eliminar participantes.
R13. Buscar participantes por los criterios: nombre, apellidos, municipio, provincia.
R14. Mostrar datos de los participantes.
R15. Establecer fecha, hora de inicio y de fin de un caso a discutir.
R16. Eliminar caso a discutir.
R17. Modificar caso a discutir (fecha, hora de inicio y de fin, participantes).
R18. Enviar mensaje de citación a todos los participantes incluidos en la discusión de un caso.
R19. Mostrar listado de salas de conferencias planificadas para el día de hoy.
R20. Mostrar caso en discusión.
R21. Resolver caso.
R22. Remitir caso.
R23. Posponer caso.
R24. Mostrar datos del caso.
R25. Mostrar sala de conversación.
R26. Visualizar informe para ser aprobado.
R27. Registrar informe del caso discutido.
R28. Mostrar informes de casos aprobados.

Un buen diseño se logra cuando se conocen las características principales del sistema, especificando todas las acciones y los actores que intervienen en las mismas, estas relaciones son reflejadas en el siguiente diagrama de casos de uso del sistema. A partir de los requisitos funcionales identificados se realizó el diagrama de casos de usos del sistema, donde se evidencian la relación entre los actores y funcionalidades del módulo Teleconsulta, con un total de 11 casos de usos identificados. Dentro de estos, se encuentra el CU Gestionar Datos Primarios (incluido), perteneciente al Registro Cubano de Historias Clínicas, que permite al médico Solicitante buscar un paciente sobre el que se llevará a cabo la discusión de un caso y a su vez seleccionarlo para obtener sus datos primarios.

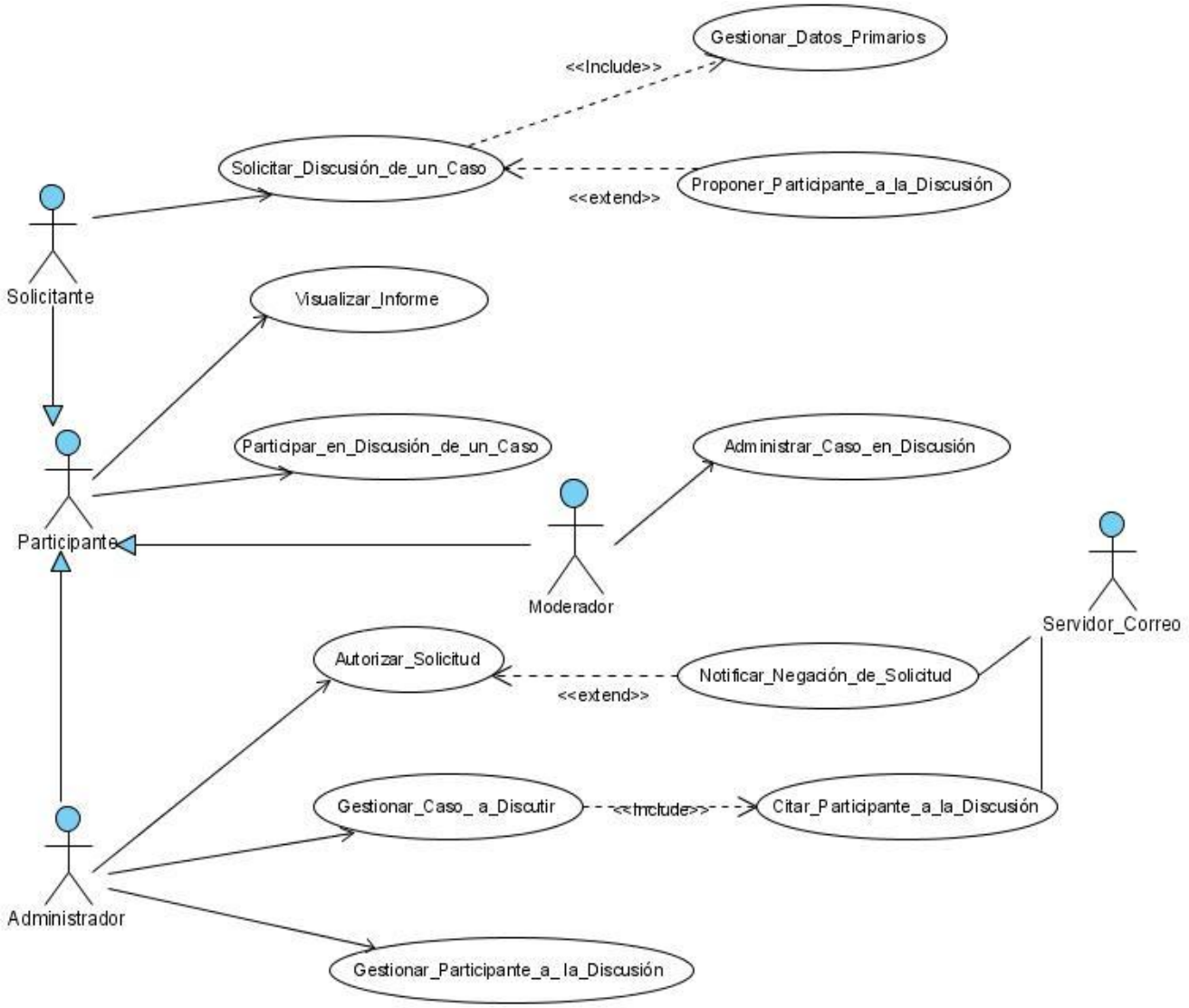


Figura 2. 1: Diagrama de casos de usos del sistema.

Requisitos No Funcionales del Sistema.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades se ven como las características que hacen al producto atractivo, usable, rápido o confiable [25].

Además, el sistema debe cumplir ciertas características que se resumen en la siguiente lista de requisitos no funcionales para el sistema en cuestión:

- **Apariencia o interfaz externa:** Se deben utilizar imágenes y colores identificados con el negocio del sistema. La interfaz externa debe estar diseñada para verse en cualquier resolución igual o superior a 1024x768.

- **Usabilidad:** La aplicación informática debe garantizar un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios. Este podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora y del ambiente web.

- **Hardware:** Para el desarrollo y ejecución de la aplicación se necesitará:
  Para el servidor de aplicación:
  Microprocesador Pentium IV a 3.0 GHz
  1 GB de RAM

  Para el Cliente:
  Microprocesador Pentium a 233 MHz (Recomendado: Pentium a 500MHz o superior).
  64 MB de RAM (Recomendado: 512MB o superior).
  52 MB de espacio de disco duro. (Recomendado: 120MB para la instalación completa del Internet Explorer 5.5).
  Conexión al servidor a través de MODEM o tarjeta de red.

Además es necesario contar con una impresora para poder imprimir los informes existentes de los casos discutidos.

- **Rendimiento:** Los tiempos de respuestas deben ser los más rápidos al igual que la velocidad de procesamiento de la información.

- **Soporte:** Se debe asegurar el soporte para los usuarios de manera que se puedan satisfacer sus necesidades a partir de mejoras, una vez puesta en marcha la aplicación. Para ello se crearán una serie de manuales de usuarios y videos tutoriales, y se mantendrá la asistencia a los usuarios.

- **Seguridad:** El sistema debe tener un mecanismo propio para gestionar la seguridad a través de niveles de acceso a la información. Los permisos al ejecutar cualquier acción deben estar de

acuerdo con el nivel jerárquico de acceso que presente el usuario en cada módulo, el cual es definido por los administradores del sistema.

- **Software:** Se requiere para el funcionamiento del sistema disponer de un servidor que cuente con Sistema Operativo Linux, Apache 2.0 y MySQL 5.0 o versiones superiores. Los usuarios del sistema deberán contar con un navegador Internet Explorer 5.5 o Mozilla Firefox 2.0 o superior, para poder acceder a las opciones que brinda el sistema.

- **Disponibilidad:** El sistema debe ser capaz de funcionar por si solo en caso de que los servicios de los diferentes componentes de SISalud no estén disponibles. Se debe garantizar 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. Se deben crear copias de respaldo periódicas que puedan restaurar el sistema en caso de fallo crítico o pérdida total de la información.

- **Persistencia:** La información debe almacenarse en bases de datos con carácter permanente con el objetivo de poder realizar análisis de la misma con el transcurso de los años.

- **Requisitos Legales:** Las herramientas y las tecnologías en que estará basada la aplicación informática deberá cumplir con las licencias de software libre.

2.2.1 Breve descripción de los casos de usos del sistema. (Ver Anexo #2).

2.3 Pautas del Diseño.

Para lograr una agradable apariencia y facilitar el uso del software se definieron algunas pautas de diseño como son:

1. Los formularios que sean creados deben estar centrados, con un ancho de un 90 % y con bordes de valor 0.
2. La primera fila del formulario debe contener el nombre de la operación que se pretende realizar con el formulario.

3. Cuando se le solicita datos al usuario se hará a través de una tabla de dos columnas. En la columna izquierda se ubicará el nombre del dato solicitado al usuario, alineado a la derecha y en la columna de la derecha el componente adecuado para la solicitud.
4. Los botones para efectuar operaciones sobre el formulario se ubicarán en la parte inferior derecha del formulario. Se posicionarán de izquierda a derecha teniendo en cuenta el peso de la operación que representan.
5. Para representar campos que deben ser de entrada obligatoria se colocará al lado derecho del componente en el cual el usuario entrará los datos un asterisco.
6. Los mensajes de error ocurridos durante la validación del formulario se mostrarán en la parte superior del campo validado en el cual ocurrió el error.

Estilos a usar en los componentes web: [26]

Componentes	Estilos
TextBox	entradaplana
ComboBox	entradaplana
Botones	sbttn
ListBox	entradaplana
NombredeCampos	nombrecampo

2.4 Aplicación de algunos patrones GRASP en Symfony.

Son muchos los patrones que se aplican en la implementación con Symfony, a continuación se mencionan algunos ejemplos de los evidenciados, ubicándolos en las capas de Modelo y Control que plantea el patrón arquitectónico MVC.

Experto: Es uno de los patrones más utilizados al trabajar con Symfony, con la inclusión de Propel para mapear la BD se justifica de algún modo. Propel genera las clases para la gestión de las entidades con las responsabilidades asignadas correctamente según el patrón Experto, las clases de abstracción de datos (Peer del Modelo) cuentan con un conjunto de funcionalidades relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

Creador: En la clase tlActions se encuentran las acciones definidas para el módulo Teleconsulta. En dichas acciones se crean los objetos de las clases que son los que instancias a las clases del modelo, lo que evidencia que la clase tlActions es "creador"de dichas entidades. Ejemplo del uso de este patrón es en la acción executeAdicionarSolicitud mediante la creación de las instancias de las clases entidades que contienen los datos de la solicitud. Ejemplos de algunas funciones utilizadas en la clase tlActions son: doSelect (), retrieveByPK (), doSelectOne ().

Alta Cohesión: El trabajar con Symfony permite la organización del trabajo en cuanto a la estructura del proyecto, esto proporciona crear y trabajar con clases con una alta cohesión. Ejemplo de esto se evidencia en la clase tlActions, la cual está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, teniendo un sentido común y un propósito único, siendo las mismas las encargadas de controlar las acciones de las plantillas.

Bajo Acoplamiento: En el modelo también se encuentran las clases que implementan la lógica de negocio y de acceso a datos, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja, poniéndose de manifiesto este patrón.

Controlador: Este patrón se observa en las clases sfFrontController, sfWebFrontController, sfContex, los "actions" y el index.php del ambiente. La arquitectura del framework (MVC) ayuda desde el principio, existiendo una capa específicamente para los controladores, que son el núcleo del mismo, el puesto de mando.

2.5 Aplicación de algunos patrones GoF que Implementa Symfony.

Patrón Singleton

Clase sfRouting – método getInstance

```
public static function getInstance()
{
if (!isset(self::$instance))
{
self::$instance = new sfRouting();
}

return self::$instance;
}
```

La clase sfRouting es una de las que utiliza el controlador frontal (sfWebFrontController), esta clase es muy utilizada porque es la encargada de enrutar todas las peticiones que se hagan a la aplicación. El singleton sfRouting define otros métodos muy útiles para la gestión manual de las rutas: ClearRoutes (), hasRoutes (), getRoutesByName ().

Patrón Command

Este patrón se pone de manifiesto en la clase sfWebFrontController, en el método dispatch (), que es la encargada de determinar cual módulo y acción usar en dependencia de la petición del usuario. Esta clase es la que está por defecto. La clase sfRouting también aplica este patrón, se encuentra desactivada por defecto y actúa en dependencia de las necesidades del administrador del sistema donde se aplique el framework, la cual se puede activar o desactivar. Pero donde más se evidencia este patrón es en la clase sfWebFrontController.

```
/**
   * Dispatches a request.
   *
   * This will determine which module and action to use by request
parameters specified by the user.
   */
  public function dispatch()
  {
    try
    {
      if (sfConfig::get('sf_logging_enabled'))
      {
        $this->getContext()->getLogger()->info('{sfController} dispatch
request');
      }

      // reinitialize filters (needed for unit and functional tests)
      sfFilter::$filterCalled = array();

      // determine our module and action
      $request    = $this->getContext()->getRequest();
      $moduleName = $request->getParameter('module');
      $actionName = $request->getParameter('action');

      // make the first request
      $this->forward($moduleName, $actionName);
    }
    .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
}
```

En este método la URL es parseada para definir los parámetros de la misma y así saber el Actions que debe responder a la petición.

Patrón Decorador

```
public function getDecoratorTemplate()
  {
    return $this->decoratorTemplate;
  }
```

Este método corresponde a la clase abstracta sfView, padre de todas las vistas, las que contienen un decorador para permitir añadir funcionalidades dinámicamente.

El archivo llamado layout.php es el que contiene el Layout de la página. Este archivo, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado “Decorator”.

Figura 2. 2: Planilla decorada con un layout.

Patrón Registry

El patrón Registry, pese a su simplicidad, es un patrón sumamente útil para los desarrolladores en la POO. En resumidas palabras, el Patrón Registry, es un medio simple y eficiente de compartir datos y objetos en la aplicación sin tener que preocuparse de mantener numerosos parámetros o hacer uso de variables globales. La aplicación de este patrón se ve en la clase sfConfig la cual es la encargada de

almacenar todas las variables de uso global en la aplicación, decir también que esta clase aplica el patrón Singleton. [27]

Symfony también aplica el patrón “Front Controller” (Controlador frontal) y por tanto tiene una estructura bien organizada de controladores, que parte desde el “index.php” del ambiente y terminan en los “Actions”. Aquí cada clase en esta capa tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, otros sólo velan por identificar mediante unos datos las clases que deben realizar determinadas tareas (Patrón GoF Command, clase sfRouting), las clases relacionadas con la configuración del sistema (sfConfig, y sfConfigHandler).

2.6 ¿Cómo se ponen de manifiesto el uso del patrón MVC que implementa Symfony?

El uso del framework que utiliza MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el framework. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador. Aplicar el patrón MVC a una aplicación resulta bastante útil además de restrictivo.

La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- sfController: Es la clase del controlador. Se encarga de decodificar la petición y transferirla a la acción correspondiente.
- sfRequest: Almacena todos los elementos que forman la petición (parámetros, cookies, cabeceras, etc.)
- sfResponse: Contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.
- El singleton de contexto (que se obtiene mediante sfContext::getInstance()): Almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. [27]

La vista se encarga de producir las páginas que se muestran como resultado de las acciones, aquí se encuentra el layout, que es común para todas las páginas. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones, asegurando la correcta combinación del layout y la página.

En el controlador se encuentran las acciones, siendo estas el corazón de la aplicación, ya que contienen toda la lógica de la aplicación. Las acciones utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición.

En el Modelo se encuentran las clases, las cuales se generan de forma automática en dependencia de la estructura de la BD. En Symfony, el acceso a los datos, se realiza mediante objetos. Propel es el motor generador que se encarga de esta generación automática para construir sus clases, creando la estructura y generando el código de las mismas. Se crean 4 archivos por cada tabla de la BD. Los cuales se expondrán brevemente a continuación:

A medida que el desarrollo del proyecto va avanzando, puede ser necesario añadir métodos y propiedades personalizadas en los objetos del modelo y con esto se aumenten las tablas o columnas. Además, cada vez que se modifica se deben regenerar las clases del modelo de objetos. Si se añaden los métodos personalizados en las clases que se generan, se borrarían cada vez que se vuelvan a generar esas clases.

Las clases con nombre Base son las que se generan directamente a partir del esquema. No se deberían modificar esas clases, porque cada vez que se genera el modelo, se borran todas las clases.
Por otra parte, las clases de objetos propias que heredan de las clases con nombre Base, no se modifican por lo que son las clases en las que se añaden los métodos propios. Estas clases heredan todos los métodos de la clase padre correspondiente, pero no le afectan las modificaciones en el esquema.

Nombre de la Clase	**Descripción**
BaseTlInformePeer	Es la clase que interactúa con el ORM utilizado

	(Propel), es la clase de abstracción a la BD, entre sus métodos se encuentran doSelectOne (), doSelect (), addJoin () entre otros.
TlInformePeer	Es una clase de abstracción de datos que tiene relación de herencia con la clase BaseTlInformePeer.
BaseTlInforme	Es una clase de acceso a datos, donde se encuentran algunos métodos como: save(), y todos los get y set de los datos correspondiente a la clase en la BD.
TlInforme	Es una clase de acceso a datos que hereda de la clase BaseTlInforme, es en esta clase donde se suele implementar algunos de los métodos de la lógica del negocio, entre estos se encuentran: getSolicitud (), getSalaConferencia ().

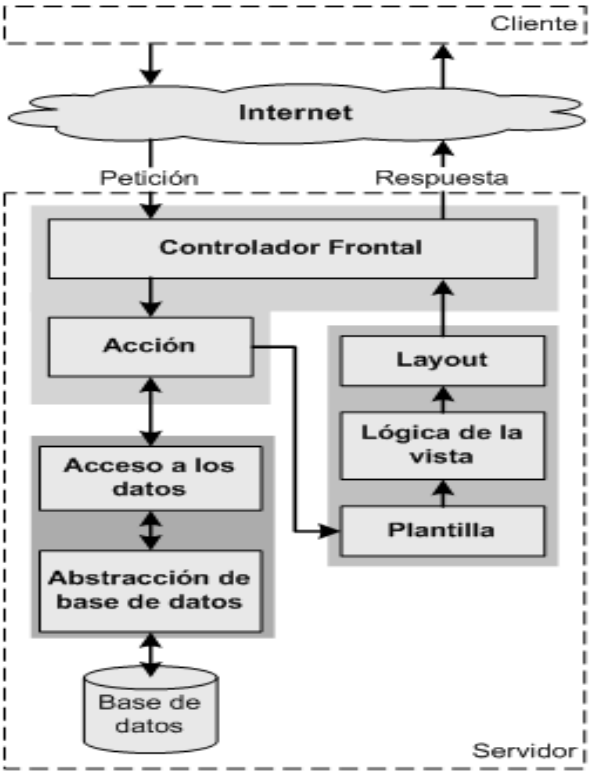


Figura 2. 3: El flujo de trabajo de Symfony

2.7 Modelo de diseño.

Para poder realizar un sistema que soporte todos los requerimientos especificados, tanto funcionales como no funcionales, se necesita modelarlo, lo cual se hace a través del diseño. 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. El modelo de diseño constituye un acercamiento a la arquitectura final de la aplicación y un esquema por el cual se regirán los desarrolladores.

2.7.1 Descripción de las clases del diseño.

Nombre de la Clase: tlActions
Nombre del método: executeCódigoÉtica()
Descripción: Se encarga de mostrar el código de ética para que sea leído y aceptado por el solicitante que desea discutir un caso. Después de aceptado una primera vez por el solicitante, no será necesario repetir esta acción, aunque el código de ética estará disponible en el menú del módulo.
Nombre del método: executeNuevaSolicitud()
Descripción: Se encarga de mostrar la solicitud con los datos primarios del paciente, un vínculo a los datos de la historia clínica, así como los participantes propuestos. Recoge la información de las nuevas solicitudes y las almacena en la base de datos.
Nombre del método: AdicionarSolicitud()
Descripción: Método privado que se invoca desde executeNuevaSolicitud(). Encapsula en sí la responsabilidad de insertar la solicitud de discusión de un caso al igual que los participantes propuestos.
Nombre del método: EliminarVariablesTemporalesNuevaSolicitud()
Descripción: Método privado invocado dentro de AdicionarSolicitud(). Se encarga de liberar las variables temporales utilizadas para almacenar los datos de la solicitud.
Nombre del método: MostrarParticipantesSolicitud()
Descripción: Método privado invocado en executeNuevaSolicitud() y handleErrorNuevaSolicitud(). Tiene la responsabilidad de recuperar las variables temporales con los datos de los participantes propuestos.
Nombre del método: VerificarExistenciaParticipante($id)

Descripción: Método privado utilizado dentro de MostrarParticipantesSolicitud() que verifica si un participante ha sido propuesto para no volverlo a proponer.
Nombre del método: EliminarParticipante($participante_e,$p)
Descripción: Método privado invocado dentro de executeNuevaSolicitud() en caso que la acción del usuario requiera eliminar un médico propuesto. Se encarga de liberar la variable temporal asociada a este participante.
Nombre del método: DatosPrimariosHistoriaClínica()
Descripción: Método privado que se invoca dentro de executeNuevaSolicitud() y handleErrorNuevaSolicitud(). Recupera los datos primarios de la historia clínica del paciente sobre el que se realiza la solicitud de discusión.
Nombre del método: getHCByIdPaciente($idPaciente)
Descripción: Método privado que se utiliza dentro de NuevaSolicitud() para obtener la historia clínica del paciente sobre el que se realiza la solicitud.
Nombre del método: handleErrorNuevaSolicitud()
Descripción: Se ejecuta cuando el archivo nuevaSolicitud.yml lanza un error, el resultado es que no se envíe el formulario y se muestren los mensajes de errores en la página**.** Se encarga de mantener la estructura de la página con su información.
Nombre del método: executeAutorizarSolicitud()
Descripción: Se encarga de mostrar un listado ordenado, por el nivel de urgencia y la fecha de arribo de las solicitudes realizadas sin aprobar, permitiendo ver los datos de las mismas para luego ser aprobadas o denegadas.
Nombre del método: DatosAutorizarSolicitud()
Descripción: Método privado invocado dentro de executeAutorizarSolicitud() que se encarga de mostrar las solicitudes que no han sido autorizadas ordenadas por el nivel de urgencia y la fecha de arribo.
Nombre del método: AutorizarSolicitudes()
Descripción: Método privado que se invoca desde executeAutorizarSolicitud() que encapsula la responsabilidad de recoger las solicitudes seleccionadas y cambiarle su estado en la base de datos a aprobadas.
Nombre del método: DenegarSolicitud($idSolicitud)
Descripción: Método privado invocado dentro de executeAutorizarSolicitud() cuando la acción del usuario así lo requiere. Elimina la solicitud seleccionada de la Base de Datos (BD).

Nombre del método: executeNotificación()
Descripción: Envía al solicitante una notificación a través del correo con la explicación del por qué fue denegada su solicitud de discusión de un caso.
Nombre del método: executePlanificarCaso()
Descripción: Muestra un listado de los casos sin planificar ordenados por la fecha de arribo y el nivel de urgencia y los ya planificados ordenados por la fecha de arribo y hora de inicio de la discusión, permitiendo además planificar nuevos casos, y consultar, modificar o eliminar los creados.
Nombre del método: handleErrorPlanificarCaso()
Descripción: Se ejecuta cuando el archivo planificarCaso.yml o la función validatePlanificarCaso lanzan un error, el resultado es que no se envíe el formulario y se muestren los mensajes de error en la página**.** Se encarga de mantener la estructura de la página con su información.
Nombre del método: validatePlanificarCaso()
Descripción: Similar a los ficheros .yml los métodos con el nombre validateXXX permiten realizar validaciones avanzadas, que no son posibles, o que son muy difíciles hacerlas en los .yml.
Nombre del método: getHoraInicioMilitar()
Descripción: Método privado invocado dentro de AdicionarSala() y ModificarSala(). Recibe el parámetro 'hora_inicio' pasado por el formulario y lo convierte a formato militar.
Nombre del método: getHoraFinMilitar()
Descripción: Método privado invocado dentro de AdicionarSala() y ModificarSala(). Recibe el parámetro 'hora_fin' pasado por el formulario y lo convierte a formato militar.
Nombre del método: ListadoSolicitudesTeleconsultas()
Descripción: Método privado utilizado por la función executePlanificarCaso(). Lista los casos sin planificar ordenados por el nivel de urgencia y la fecha de arribo y los planificados ordenados por la fecha y hora de inicio de la discusión.
Nombre del método: EliminarSalaConferencia()
Descripción: Método privado invocado dentro de executePlanificarCaso() en el caso que la acción del usuario así lo indique. Elimina un caso planificado seleccionado, y la solicitud correspondiente al mismo, notificando a través de un correo a los participantes de la decisión tomada.
Nombre del método: AgregarSala()
Descripción: Método privado invocado dentro de executePlanificarCaso() que se encarga de planificar un nuevo caso con su fecha, hora de inicio y fin de la discusión.
Nombre del método: NotificaciónEliminarSala()

Descripción: Método privado invocado en la función EliminarSalaConferencia(). Envía un correo automático a los participantes notificando que se eliminó el caso planificado.
Nombre del método: executeDatosCaso()
Descripción: Muestra los datos de un caso planificado y permite adicionar o eliminar los médicos que van a participar en la discusión, citando a los participantes a la misma.
Nombre del método: MostrarParticipantesAprobados()
Descripción: Método privado utilizado por executeDatosCasos() para encapsular la recuperación de la BD de los participantes aprobados.
Nombre del método: VerificarExistenciaAprobado($id)
Descripción: Método privado utilizado por la función MostrarParticipantesAprobados. Verifica que no se adicione un participante que ya está propuesto a participar en la discusión de un caso.
Nombre del método: EliminarParticipanteCaso($participante_e,$p)
Descripción: Método privado, invocado por MostrarParticipantesAprobados() en caso que la acción del usuario requiera eliminar un médico propuesto. Se encarga de liberar la variable temporal asociada a este participante.
Nombre del método: EnviarCitaciónSala()
Descripción: Método privado invocado dentro de executeDatosCaso(). Envía un correo automático citando a los participantes que van a participar en la discusión del caso con los datos del mismo.
Nombre del método: executeModificarCaso()
Descripción: Modifica los datos de un caso planificado seleccionado.
Nombre del método: handleErrorModificarCaso()
Descripción: Se ejecuta cuando el archivo modificarCaso.yml o la función validateModificarCaso() lanzan un error, el resultado es que no se envía el formulario y se muestren los mensajes de error en la página**.** Se encarga de mantener la estructura de la página con su información.
Nombre del método: validateModificarCaso()
Descripción: Permite realizar validaciones que no son triviales archivos .yml. Los métodos con el nombre validateXXX permiten realizar validaciones avanzadas, que no son posibles, o que son muy difíciles hacerlas en los .yml.
Nombre del método: DatosModificarCaso ()
Descripción: Método privado invocado dentro de executeModificarCaso() y handleErrorModificarCaso() que se encarga de encapsular la responsabilidad de mostrar los datos del caso a modificar.

Nombre del método: MostrarParticipantesModificarAprobados()
Descripción: Método privado invocado dentro de executeModificarCaso() y handleErrorModificarCaso() que se encarga de encapsular la responsabilidad de mostrar los participantes que van a participar en la discusión de un caso planificado.
Nombre del método: VerificarExistenciaModificarAprobado($id)
Descripción: Método privado utilizado por MostrarParticipantesModificarAprobados(). Verifica que no se adicione un participante que ya está propuesto a participar en la discusión de un caso planificado que se quiere modificar.
Nombre del método: EliminarParticipanteModificarCaso($participante)
Descripción: Método privado utilizado por MostrarParticipantesModificarAprobados(). Elimina el participante que se seleccione del caso planificado que se quiere modificar, de la BD.
Nombre del método: EnviarModificaciónSala($dia_original,$hora_original)
Descripción: Método privado utilizado por executeModificarCaso(). Envía un correo automático actualizando de los cambios en la planificación a los participantes que van a participar en la discusión del caso con los datos del mismo.
Nombre del método: executeBuscarUMT()
Descripción: Este es el execute correspondiente a la página que incluye el componente que se utiliza para buscar participantes cuando se está modificando un caso.
Nombre del método: executeCasosDelDía()
Descripción: Muestra los casos que hay planificados para el día actual ordenados por la hora de inicio de la discusión.
Nombre del método: DatosCasosDelDía ()
Descripción: Método privado utilizado por executeCasosDelDía(). Muestra un listado con los casos planificados para el día de hoy.
Nombre del método: executeDatosSolicitud()
Descripción: Muestra los datos de la solicitud de discusión de un caso seleccionado.
Nombre del método: DatosSolicitud()
Descripción: Método privado utilizado por executeDatosSolicitud() que encapsula dentro de sí los datos de la solicitud de discusión de un caso seleccionado.
Nombre del método: executeDatosSolicitudPDF()
Descripción: Muestra en formato .pdf los datos de la solicitud de discusión de un caso seleccionado.
Nombre del método: executeInformaciónCaso()

Descripción: Muestra los datos de un caso planificado con los participantes del mismo.
Nombre del método: DatosInformaciónCaso()
Descripción: Método privado utilizado por executeInformaciónCaso (). Muestra los datos de un caso planificado.
Nombre del método: ParticipantesInformaciónCaso()
Descripción: Método privado utilizado por DatosInformaciónCaso (). Muestra los participantes de un caso seleccionado.
Nombre del método: executeInformaciónCasoPDF()
Descripción: Muestra en formato .pdf los datos del caso planificado.
Nombre del método: executeBuscarPac()
Descripción: Este es el execute de la página que incluye el componente que se utiliza para buscar al paciente sobre el que se va a ser la solicitud de discusión de un caso.
Nombre del método: executeCasoDiscutido()
Descripción: Presenta un formulario que recoge la información referente al informe que se debe elaborar del caso discutido.
Nombre del método: handleErrorCasoDiscutido ()
Descripción: Se ejecuta cuando el archivo casoDiscutido.yml lanza un error, el resultado es que no se envíe el formulario y se muestren los mensajes de error en la página**.** Se encarga de mantener la estructura de la página con su información.
Nombre del método: executeBuscarU()
Descripción: Este es el execute de la página que incluye el componente que se utiliza para buscar participantes cuando se está realizando la solicitud de un caso.
Nombre del método: executeBuscarUT()
Descripción: Este es el execute de la página que incluye el componente que se utiliza para buscar participantes cuando se está planificando el caso.
Nombre del método: executeBuscarUsuario()
Descripción: Muestra los resultados de la búsqueda de executeBuscarU(), executeBuscarUMT() y executeBuscarUT().
Nombre del método: BuscarUsuarioBD()
Descripción: Invocado por executeBuscarU(), executeBuscarUMT() y executeBuscarUT(). Realiza la búsqueda de médicos dentro de la base de datos.
Nombre del método: executeMnp()

Descripción: Devuelve un arreglo con todos los municipios de una provincia dada.
Nombre del método: executeHistoriaClínica()
Descripción: Muestra la historia clínica del paciente.
Nombre del método: executeCasosADiscutir()
Descripción: Muestra el caso que se va a discutir en la sala de conferencia el cual será iniciado y luego finalizado.
Nombre del método: executeChatEnviarMensaje()
Descripción: Se ejecuta periódicamente para enviar y recibir mensajes y mantener la ventana del chat actualizada. Es invocado desde la vista del executeCasosADiscutir().
Nombre del método: executeActualizarUsuarios()
Descripción: Se ejecuta periódicamente para mantener la ventana de usuarios conectados actualizada. Es invocado desde la vista del executeCasosADiscutir().
Nombre del método: TieneMensajesRecientes($idMédico,$idSala)
Descripción: Método privado utilizado en executeActualizarUsuarios(). Permite conocer si un usuario se mantiene activo dentro de la discusión.
Nombre del método: MédicoEnSala($user,$idSala)
Descripción: Método privado utilizado por executeActualizarUsuarios(). Verifica si un médico pertenece a una sala de conferencia.
Nombre del método: InicializarChat()
Descripción: Método privado utilizado por executeCasosADiscutir().Inicializa la sala de conferencia para la discusión del caso. Crea el historial.
Nombre del método: TerminarChat()
Descripción: Método privado utilizado por executeCasosADiscutir(). Termina la discusión del caso cerrando la sala de conferencia.
Nombre del método: executeAprobarInformes()
Descripción: Muestra un listado con los informes de los casos discutidos para ser aprobados.
Nombre del método: AprobarInformes()
Descripción: Método privado utilizado por executeAprobarInformes() .Se encarga de aprobar los informes que luego estarán visibles a los participantes.
Nombre del método: DatosAprobarInformes ()
Descripción: Método privado utilizado por executeAprobarInformes(). Encapsula dentro de sí la responsabilidad de recuperar de la BD los informes que aún no han sido aprobados.

Nombre del método: executeDatosInforme ()
Descripción: Muestra los datos de un informe.
Nombre del método: ParticipantesInforme($idSala)
Descripción: Método privado utilizado por executeDatosInforme () y executeInformePDF(). Devuelve un arreglo con los médicos que participaron en la discusión del caso
Nombre del método: executeInformePDF()
Descripción: Muestra en formato .pdf el informe seleccionado.
Nombre del método: sendMail($to,$subject,$message)
Descripción: Método privado invocado por varios executeXXX. Permite enviar correos, dado un arreglo de direcciones, un asunto y un cuerpo del mensaje.
Nombre del método: esSolicitante($idMédico, $idCaso)
Descripción: Método privado utilizado por múltiples funciones. Permite conocer si un médico es solicitante o no de la discusión de un caso dado.

2.7.2 Diagramas de clases web del diseño.

Un diagrama de clases del diseño representa las clases del diseño y sus objetos, así como los subsistemas del diseño.

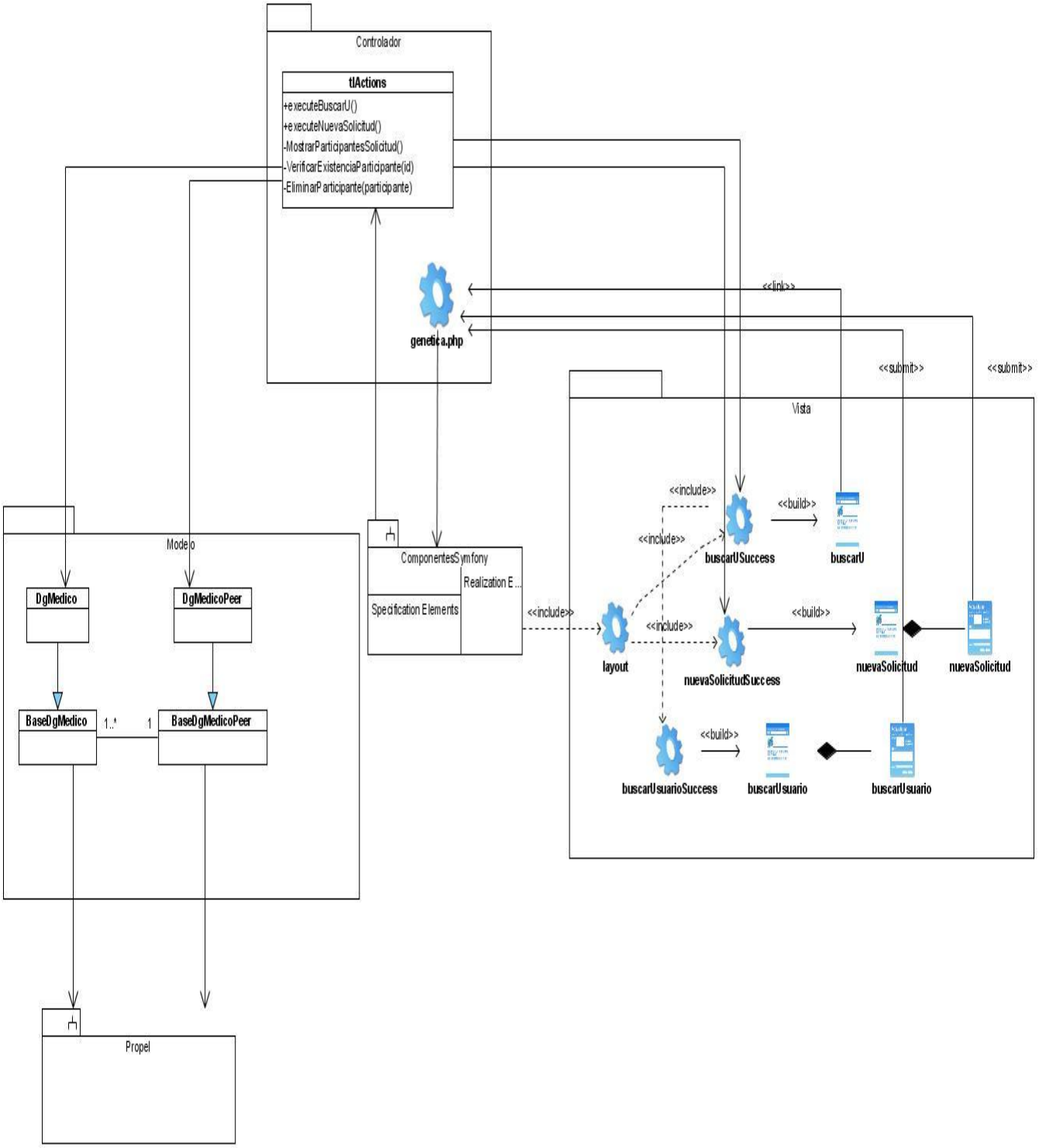


Figura 2. 4: Diagrama de clases del diseño CU: "Proponer participantes a la discusión".

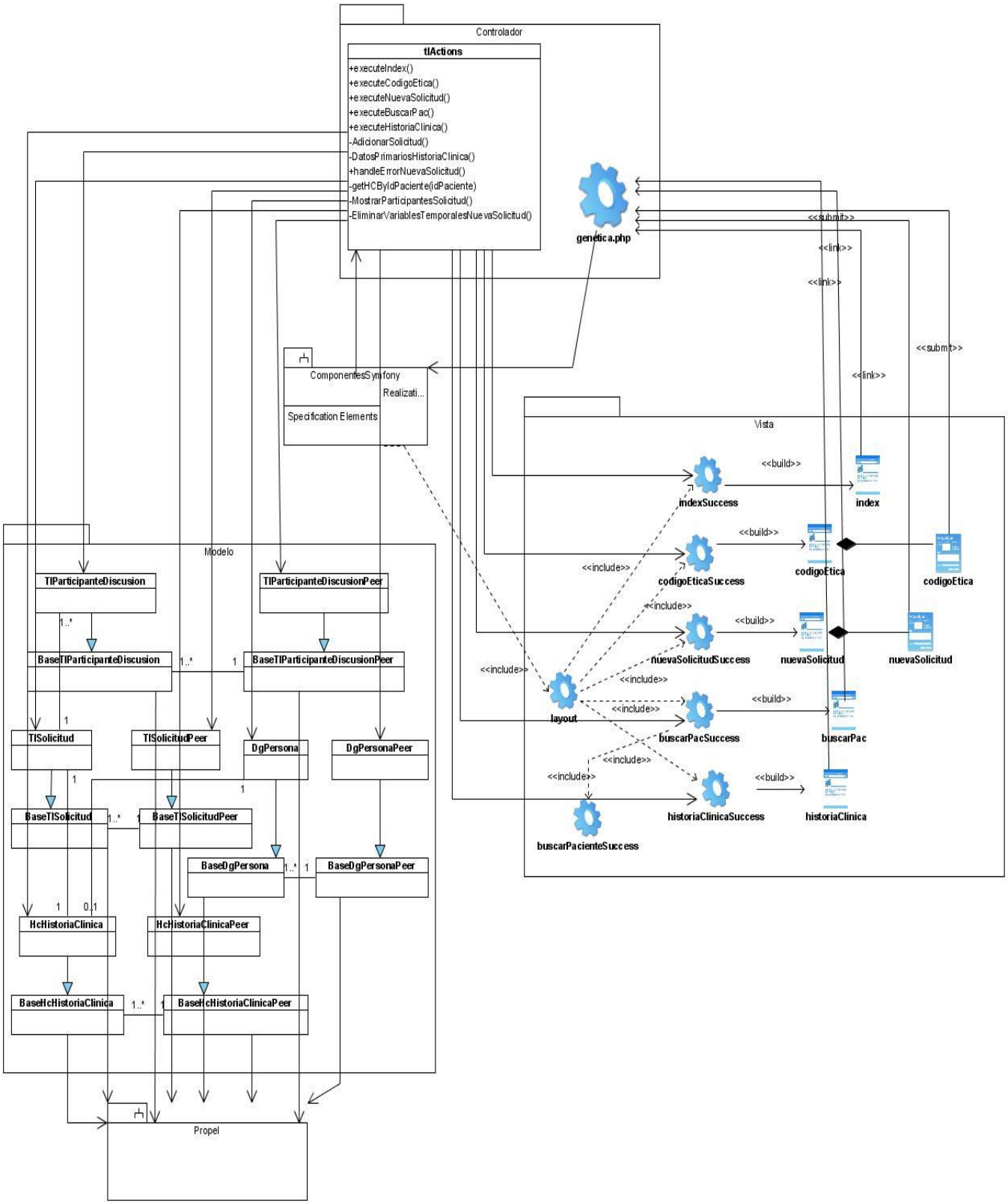


Figura 2. 5: Diagrama de clases del diseño CU: "Solicitar discusión de un caso".

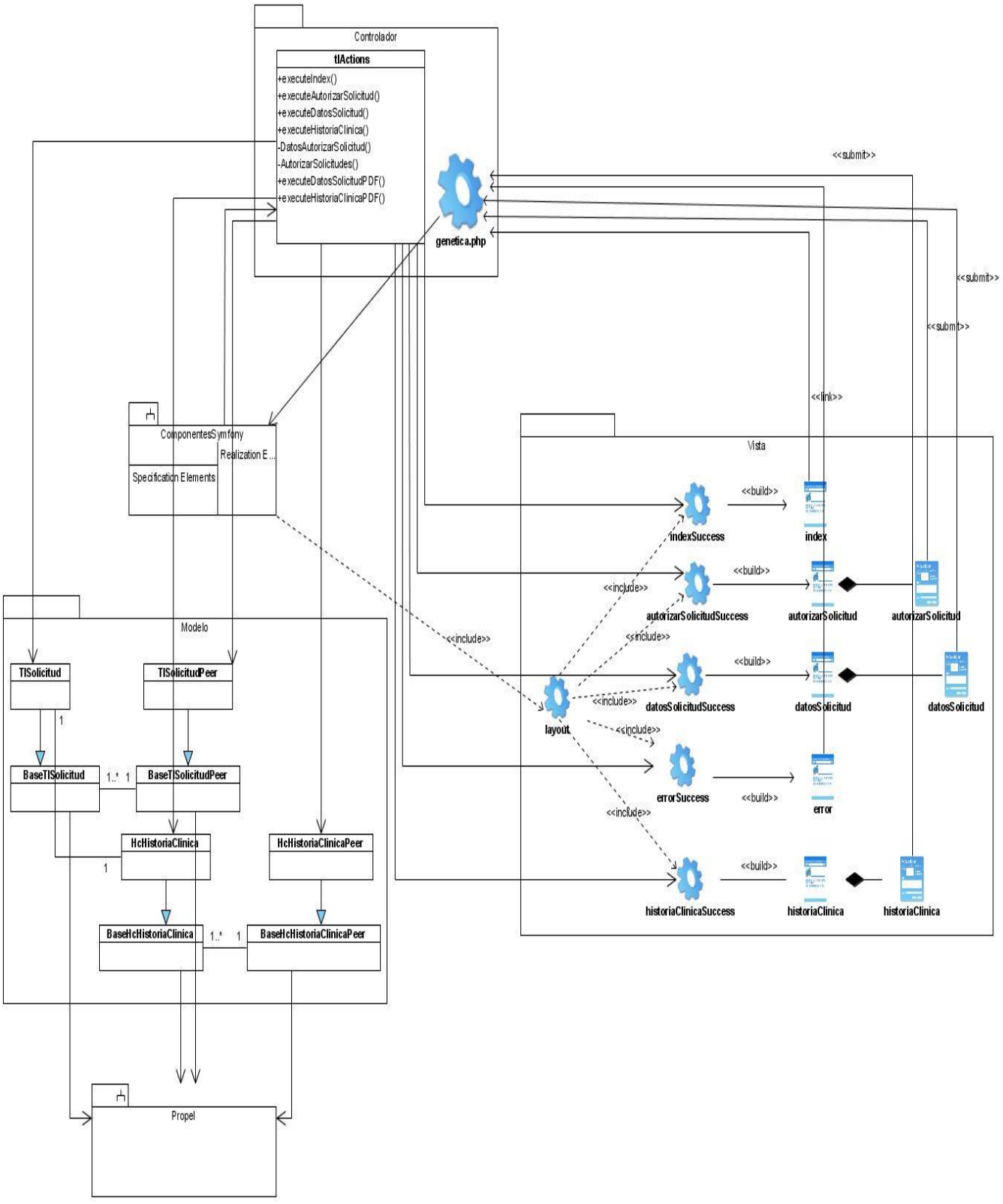


Figura 2. 4: Diagrama de clases del diseño CU: "Autorizar solicitud".

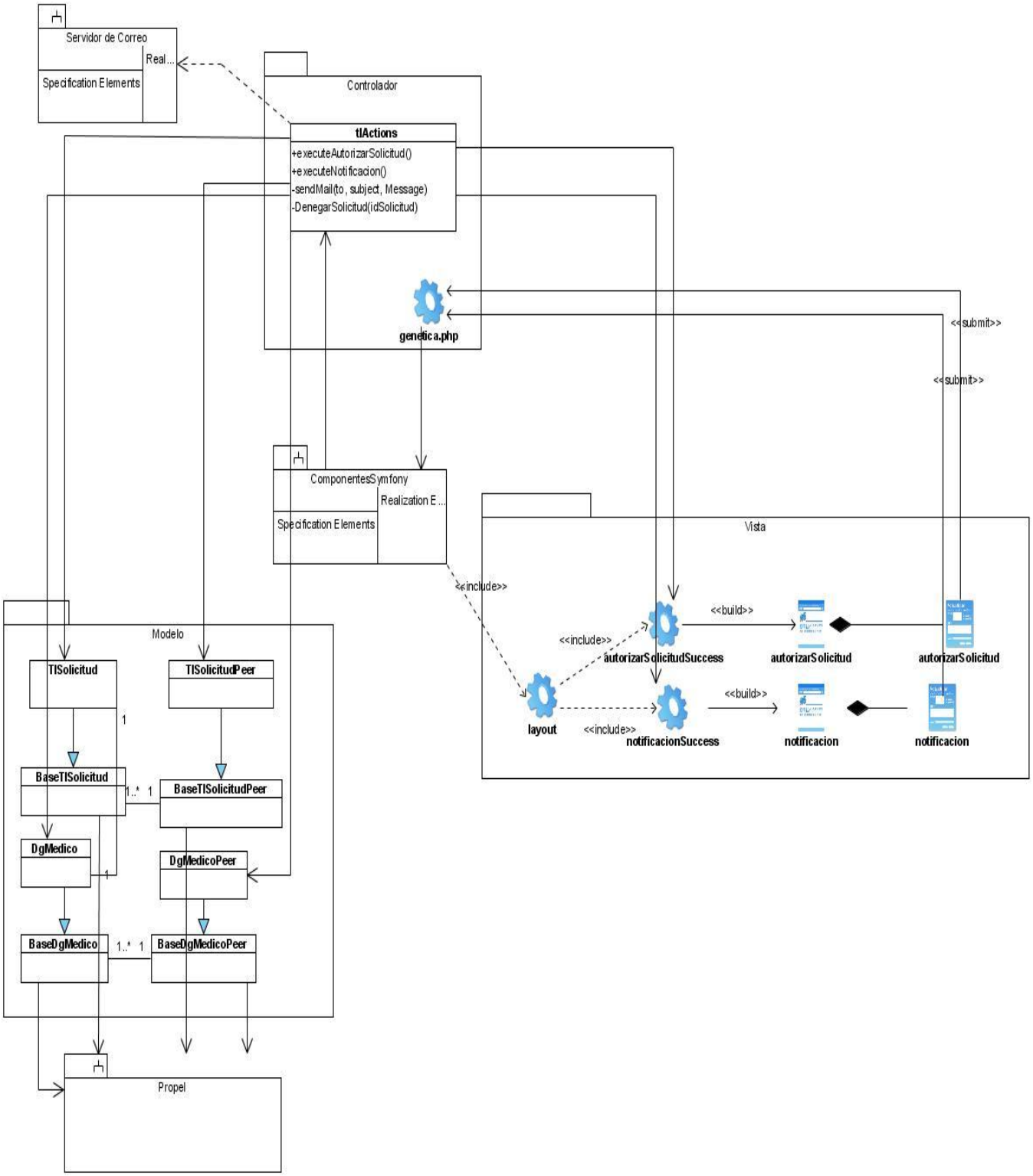


Figura 2. 5: Diagrama de clases del diseño CU: "Notificar negación".

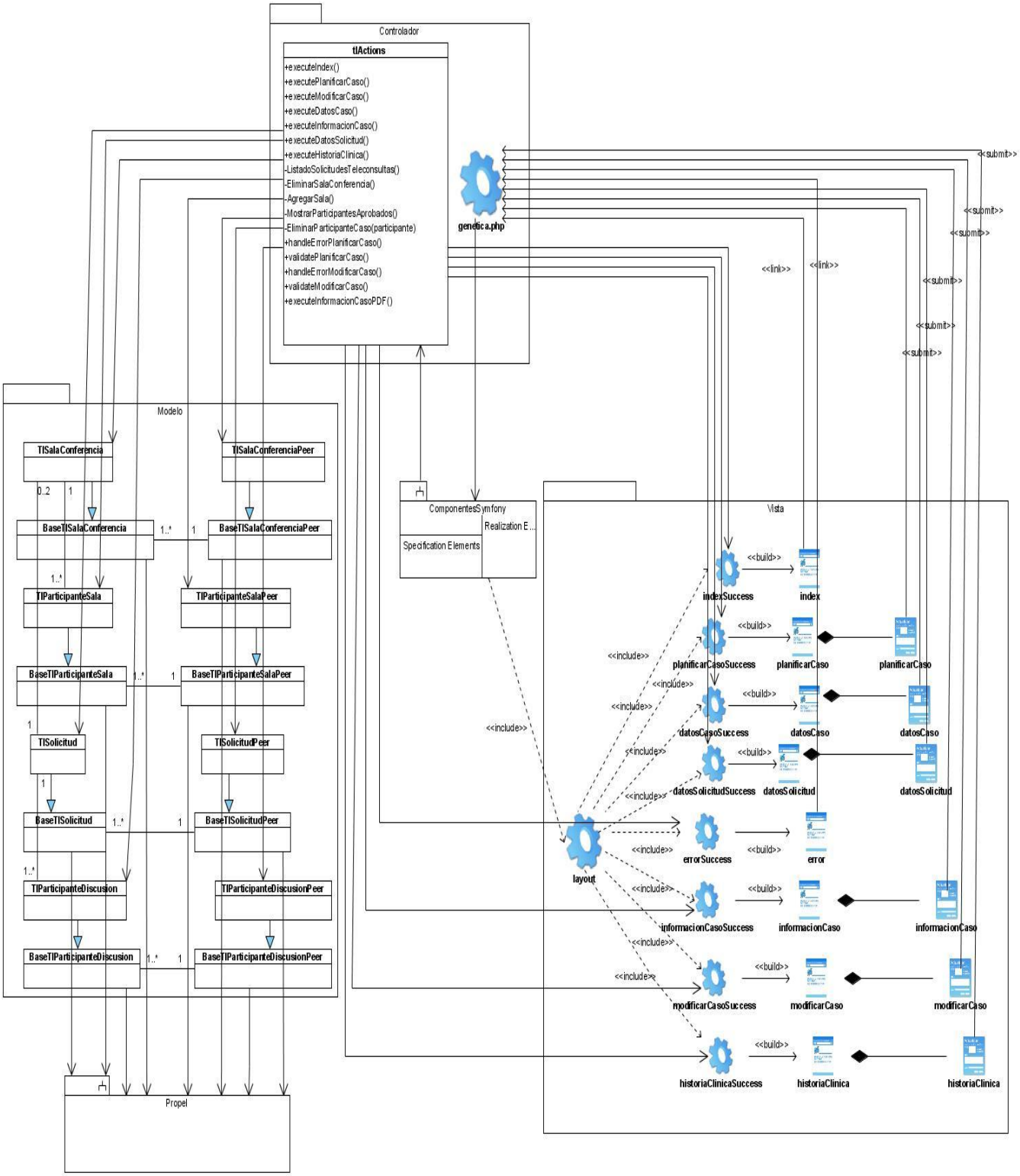


Figura 2. 6: Diagrama de clases del diseño CU: "Gestionar caso a discutir".

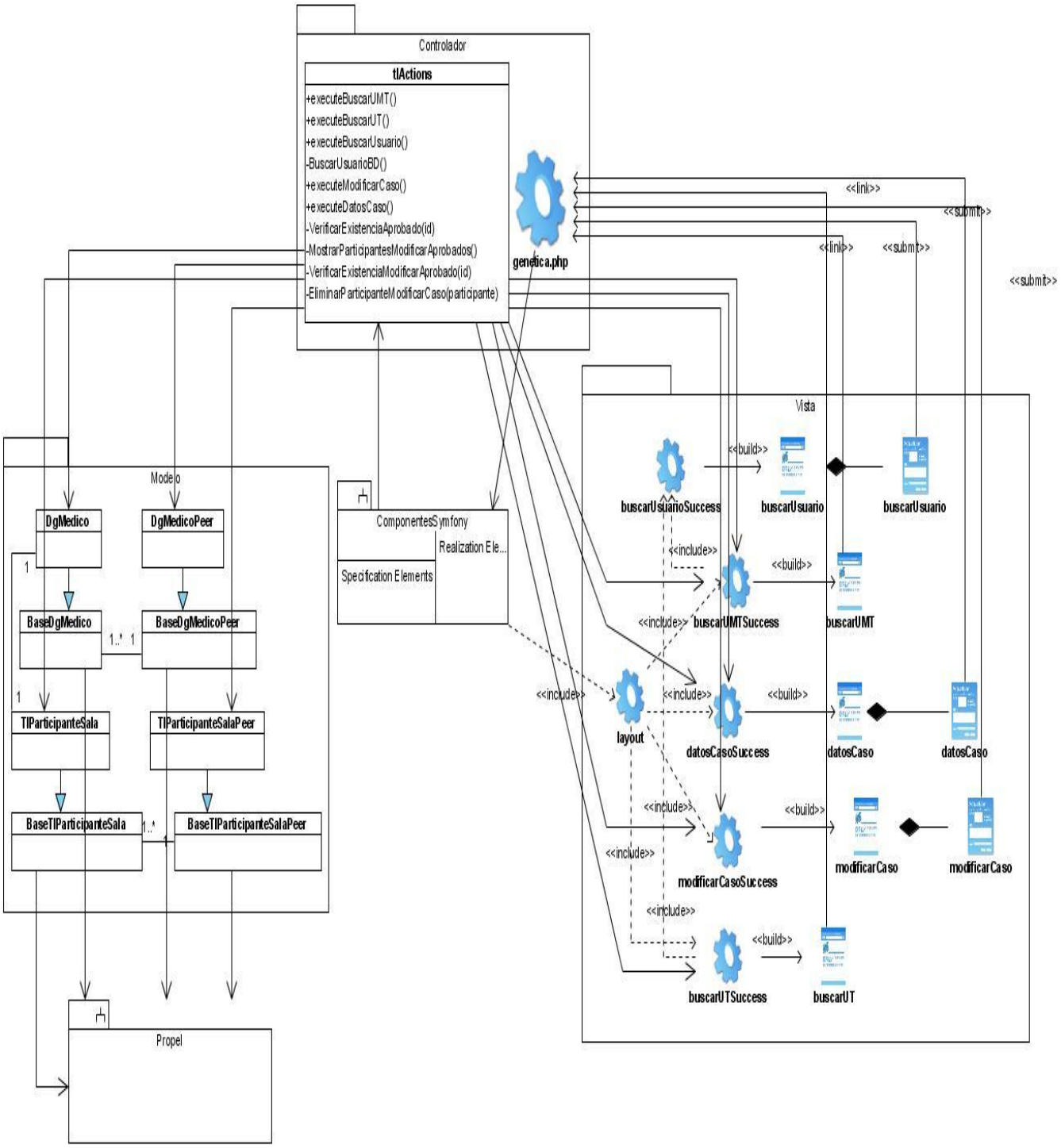


Figura 2. 7: Diagrama de clases del diseño CU: "Gestionar participante a la discusión".

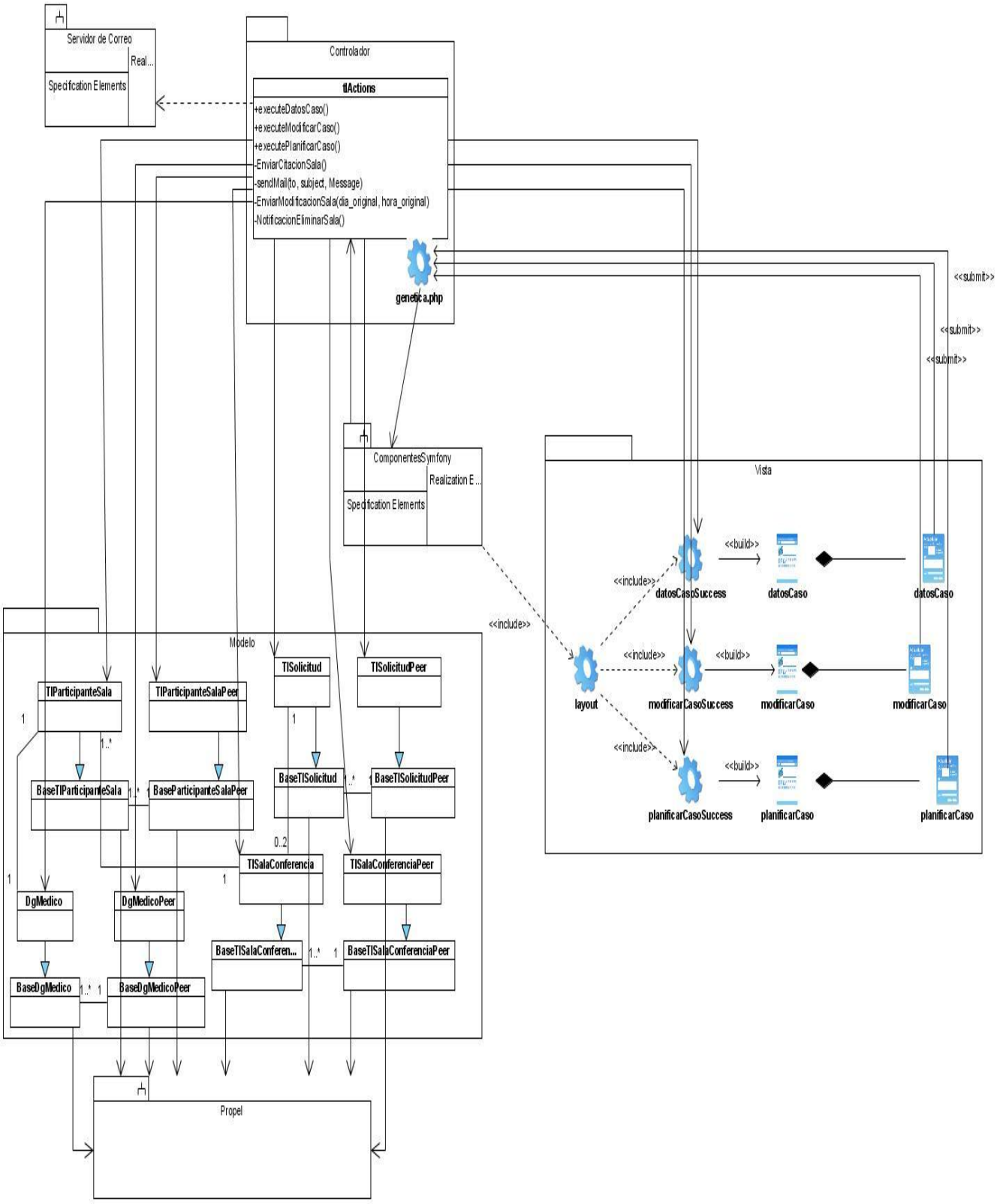


Figura 2. 8: Diagrama de clases del diseño CU: "Citar participante a la discusión".

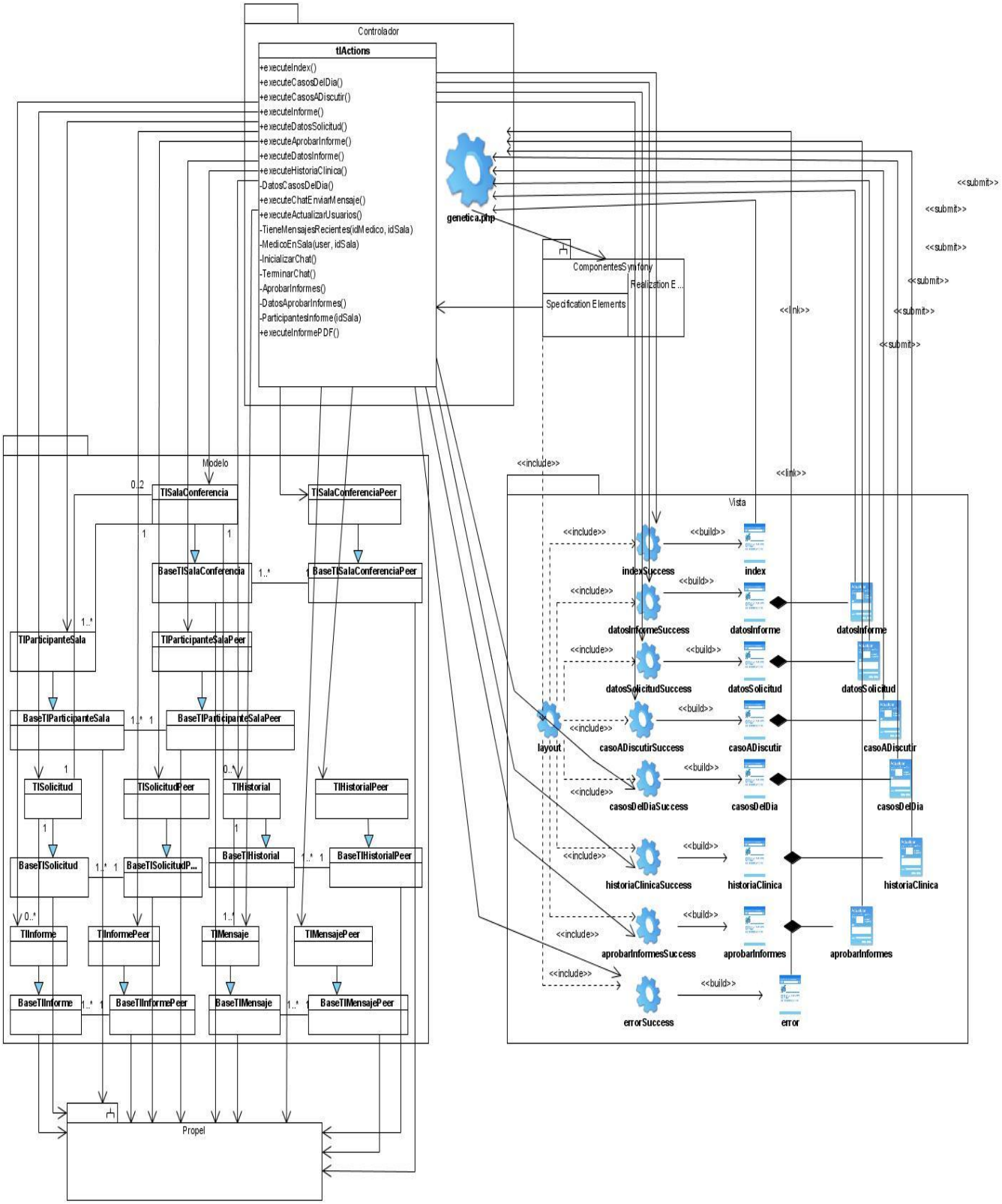


Figura 2. 9: Diagrama de clases del diseño CU: "Administrar caso en discusión".

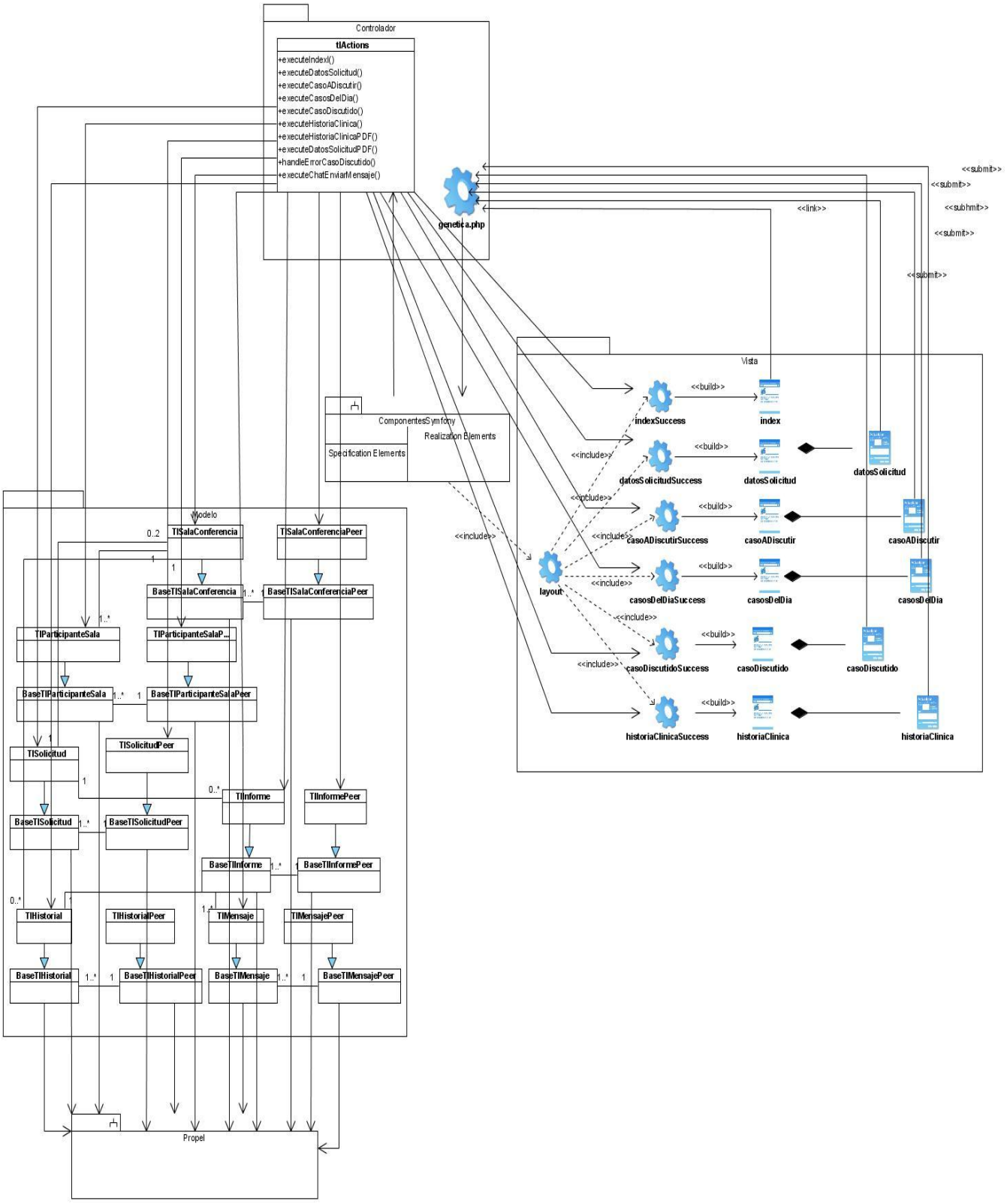


Figura 2. 10: Diagrama de clases del diseño CU: "Participar en discusión de un caso".

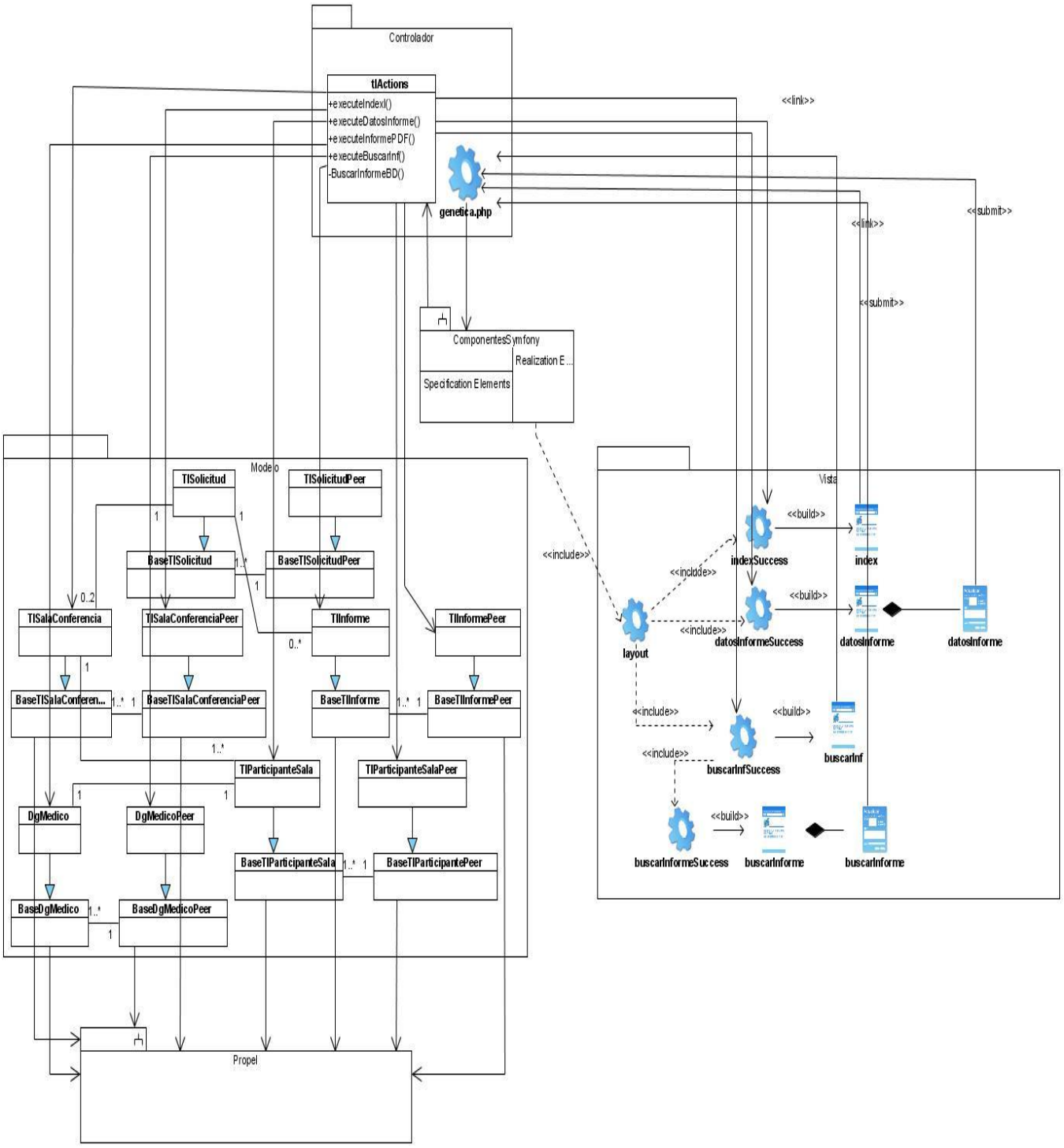


Figura 2. 11: Diagrama de clases del diseño CU: "Visualizar informe".

2.7.4 Diagramas de Interacción (Secuencia). (Ver Anexo #3)

Los Diagramas de Interacción (Secuencia y Colaboración) muestran la relación que se establece mediante el paso de mensajes entre los distintos objetos que participan en un escenario. Modelan también el comportamiento dinámico del sistema, es decir, el flujo de control en una operación. Dentro de estos el diagrama de secuencia destaca la ordenación temporal de los mensajes. Al ser una aplicación web se requerirá del uso de una extensión de UML para el modelado de este tipo de sistemas, con el estereotipo Server Page para representar la página web que tiene código que se ejecuta en el servidor; Client Page, que representa una página web con formato HTML; y Form, como el grupo de elementos de entrada que son parte de una página cliente.

2.8 Modelo de despliegue.

El Modelo de Despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo, se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. A continuación se muestra el diagrama de despliegue correspondiente al módulo Teleconsulta.

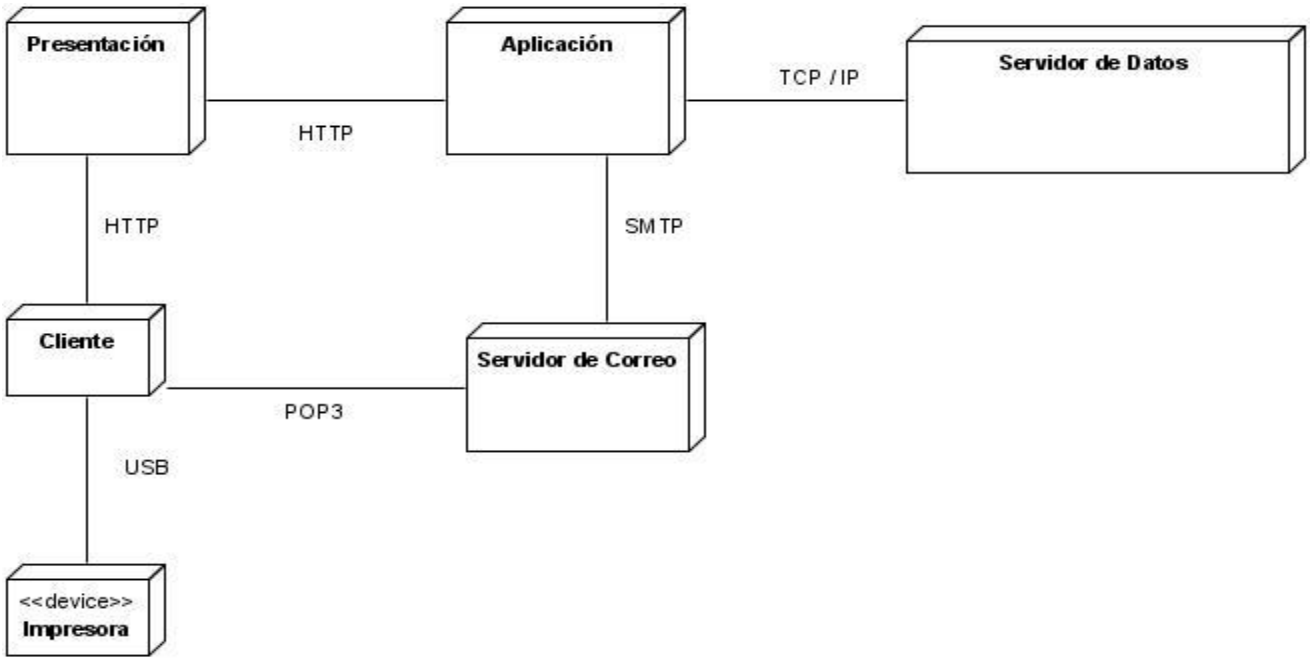


Figura 2. 12: Diagrama de despliegue.

2.9 Validación.

La validación de los datos de la acción, normalmente los parámetros de la petición, es una tarea repetitiva y tediosa. Symfony incluye un sistema de validación, utilizando métodos de la clase acción.
Se ve en primer lugar un ejemplo, cuando un usuario hace una petición a miAccion, Symfony siempre busca primero un método llamado validateMiAccion(). Si lo encuentra, Symfony ejecuta ese método. El valor de retorno de esta validación determina el siguiente método que se ejecuta: si devuelve true, entonces se ejecuta el método executeMiAccion(); en otro caso, se ejecuta handleErrorMiAccion(). En el caso de que handleErrorMiAccion() no exista, Symfony busca un método genérico llamado handleError(). Si tampoco existe, simplemente devuelve el valor sfView::ERROR para producir la plantilla miAccionError.php. La clave para un correcto funcionamiento de la validación es respetar la convención de nombres para los métodos de la acción:

- validateNombreAccion es el método de validación, que devuelve true o false. Se trata del primer método buscado cuando se solicita la acción NombreAccion. Si no existe, la acción se ejecuta directamente.
- handleErrorNombreAccion es el método llamado cuando el método de validación falla. Si no existe, entonces se muestra la plantilla Error.
- executeNombreAccion es el método de la acción. Debe existir para todas las acciones.

Se puede utilizar este mecanismo para implementar la validación de los formularios (esto es, controlar los valores introducidos por el usuario en un formulario antes de procesarlo). [27]

Symfony incluye la posibilidad de validación del lado del servidor, para ello utiliza archivos .yml los cuales se vinculan a las acciones. Para la validación con ficheros YML, Symfony incluye un conjunto de clases tales como sfStringValidator, sfMailValidator, sfNumberValidator, sfURLValidator, sfFileValidator entre otras; y además la posibilidad de implementación de nuevas clases en los casos que las existentes no resuelvan el problema en cuestión. Específicamente en este caso se añadieron las clases genéticaDateValidator y genéticaHorasValidator implementadas en los ficheros genéticaDateValidator.class.php y genéticaHorasValidator.class.php respectivamente, los cuales se encuentran dentro del directorio lib de la aplicación genética. Ambas clases extienden de la clase del symfony sfValidator. La clase genéticaDateValidator se creó con el objetivo de validar rangos de horas que los usuarios deben seleccionar y que deben cumplir con ciertos criterios que establecen las reglas

del negocio. Específicamente se utilizan en la implementación del CU Planificar Caso. La clase genéticaHorasValidator al igual que la anterior, se crea con objetivo similar, pero esta vez para validar rangos de horas que deben cumplir con ciertas reglas del negocio. A continuación se muestra el código del archivo códigoÉtica.yml.

```
fillin:
  enabled: true #Habilita volver a mostrar los datos

  param:
    #name: prueba # Nombre del formulario (no es necesario indicarlo si solo
hay 1 formulario en la pÃ¡gina)
    #skip_fields: [email] # No mostrar los datos introducidos en estos campos
    exclude_types: [hidden, password] # No mostrar los campos de estos tipos
    check_types: [text, checkbox, radio, password, hidden] # Muestra los datos
de estos tipos de campos

fields:
  urgencia:
    required:
      msg: El campo urgencia no debe estar vacio
                                tipo_urgencia:
    required:
      msg: El campo nombre no se puede dejar vacio
  fundamentacion:
    required:
      msg: Debe exponer una fundamentaci&oacute;n
  motivo:
    required:
      msg: Debe exponer un motivo
```

2.10 Seguridad.

El framework Symfony, que es el utilizado en el presente trabajo de diploma, garantiza la seguridad al ser ejecutada cada acción a través de un filtro verificando si el usuario logueado tiene los privilegios necesarios a la acción en cuestión, como por ejemplo:

- Las acciones requieren que los usuarios estén autenticados.
- La autenticación y las credenciales son privilegios agrupados bajo el nombre y simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario en grupos.

Para garantizar la seguridad en el sistema, este contará con un módulo para la gestión de la autenticación. Cuando un usuario se autentica en el sistema se crean automáticamente un grupo de

variables temporales que permiten al programador comprobar en cada momento que nivel y que permisos tiene cada usuario. Para el control de estos permisos se crearon permisos y grupos de usuarios. Específicamente en el ámbito de la Teleconsulta todas las acciones requieren que los usuarios estén autenticados en el sistema para acceder a ella, además las credenciales utilizadas en este módulo son las definidas por el módulo de administración del SIGM, existen los grupos nacAdminTl y provAdminTl que cuentan con los permisos Nacionales y Provinciales respectivamente, el resto de los usuarios se asume por defecto que son genetistas de nivel municipal. Los usuarios que no pertenezcan a los grupos de administradores de teleconsultas nacionales o provinciales (nacAdminTl, provAdminTl) sin los permisos Provincial o Nacional sólo tienen posibilidad de realizar nuevas solicitudes, participar en los casos que han solicitado o en los que han sido convocados y visualizar los informes de los casos en que han participado. Los usuarios del grupo provAdminTl tienen la posibilidad de realizar solicitudes, autorizar solicitudes, planificar, modificar y cancelar casos, participar en discusiones y aprobar informes, teniendo en cuenta que solamente podrá realizar todas estas operaciones sobre solicitudes de su provincia. Los usuarios del grupo nacAdminTl tienen igual posibilidades que los del grupo provAdminTl, solamente que el estado de las solicitudes que ellos atenderán será remitido, el resto de las posibilidades son las mismas.

Para restringir el acceso a páginas no autorizadas, según el grupo al que pertenece cada usuario se definió el fichero security.yml dentro del directorio config del módulo tl de la aplicación. En este fichero se especifica cuáles grupos de usuarios pueden tener acceso a cada página. Si el usuario autenticado no pertenece al grupo indicado el navegador mostrará un mensaje de error. A continuación se muestra el fichero security.yml perteneciente a la carpeta config del módulo Teleconsulta.

```
#autorizarSolicitud:
  #credentials: [[ nacAdminTl, provAdminTl ]]

#planificarCaso:
  #credentials: [[ nacAdminTl, provAdminTl ]]

#aprobarInformes:
  #credentials: [[ nacAdminTl, provAdminTl ]]

all:
  is_secure: on
```

2.11 Conclusiones.

El principal resultado del diseño es el Modelo de Diseño que conserva la estructura del sistema propuesto por el Modelo de Análisis. Se realizó la distribución física del sistema a través de un modelo de despliegue, que describe todas las configuraciones sobre las cuales debe implementarse el sistema. Se pudo definir sobre la base de la modelación correspondiente a este capítulo, la estructura que debe seguir la implementación para una correcta orientación a objetos en cada uno de sus escenarios. Los diagramas de diseño y despliegue se consideran la entrada principal para el Flujo de Trabajo de Implementación. Además se explicó la definición del diseño que se aplicó, la forma de tratar los errores y la seguridad del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN DEL SISTEMA

3.1 Introducción.

La implementación comienza precisamente con los resultados arrojados por el modelo de diseño, pero esta vez esa concepción del sistema se llevará al nivel más bajo posible, o sea al nivel de componentes. Además se realiza la codificación de los métodos más importantes y se reflejan algunas de las interfaces de la aplicación. Finalmente se realizan pruebas a nivel de desarrollador para asegurar la calidad del software. Como resultado de este capítulo se genera el artefacto diagrama de componentes.

3.2 Modelo de Implementación.

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. [28]

Diagrama de Componentes.

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo archivos, cabeceras, módulos, paquetes.) y muestra las dependencias entre estos componentes. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema.

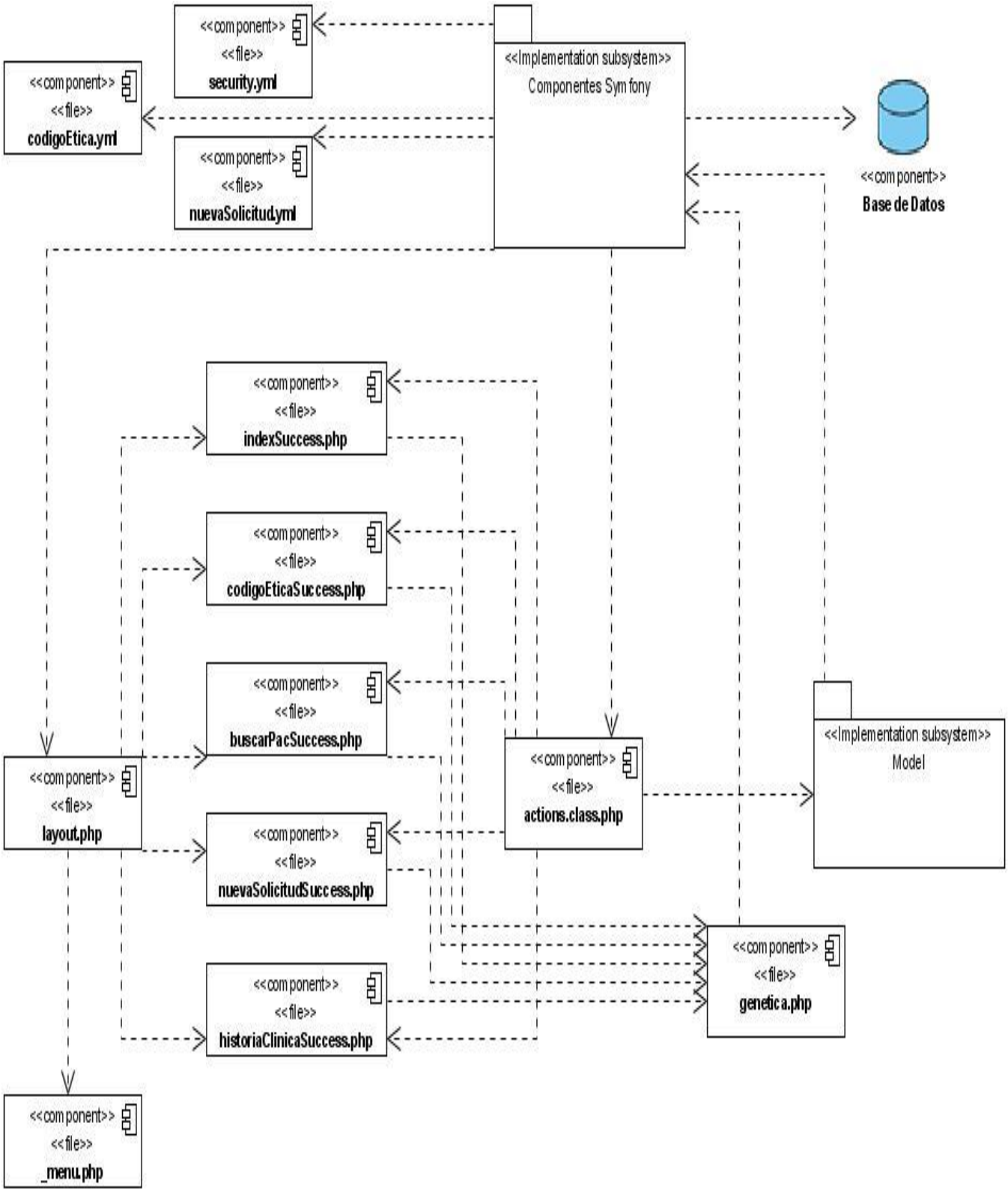


Figura 3. 1: Diagrama de componentes del CU: "Solicitar discusión de un caso".

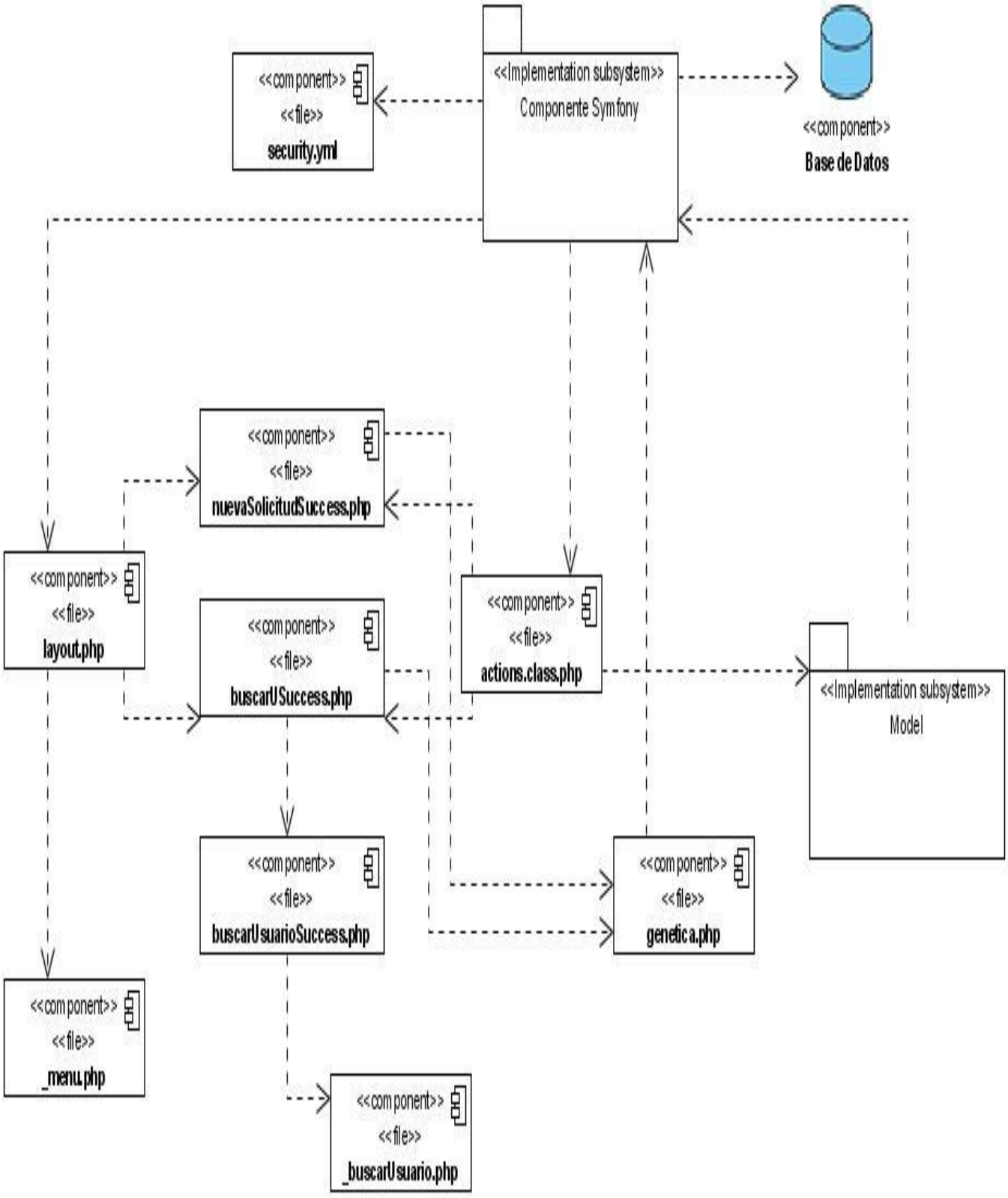


Figura 3. 2: Diagrama de componentes del CU: "Proponer participantes a la discusión".

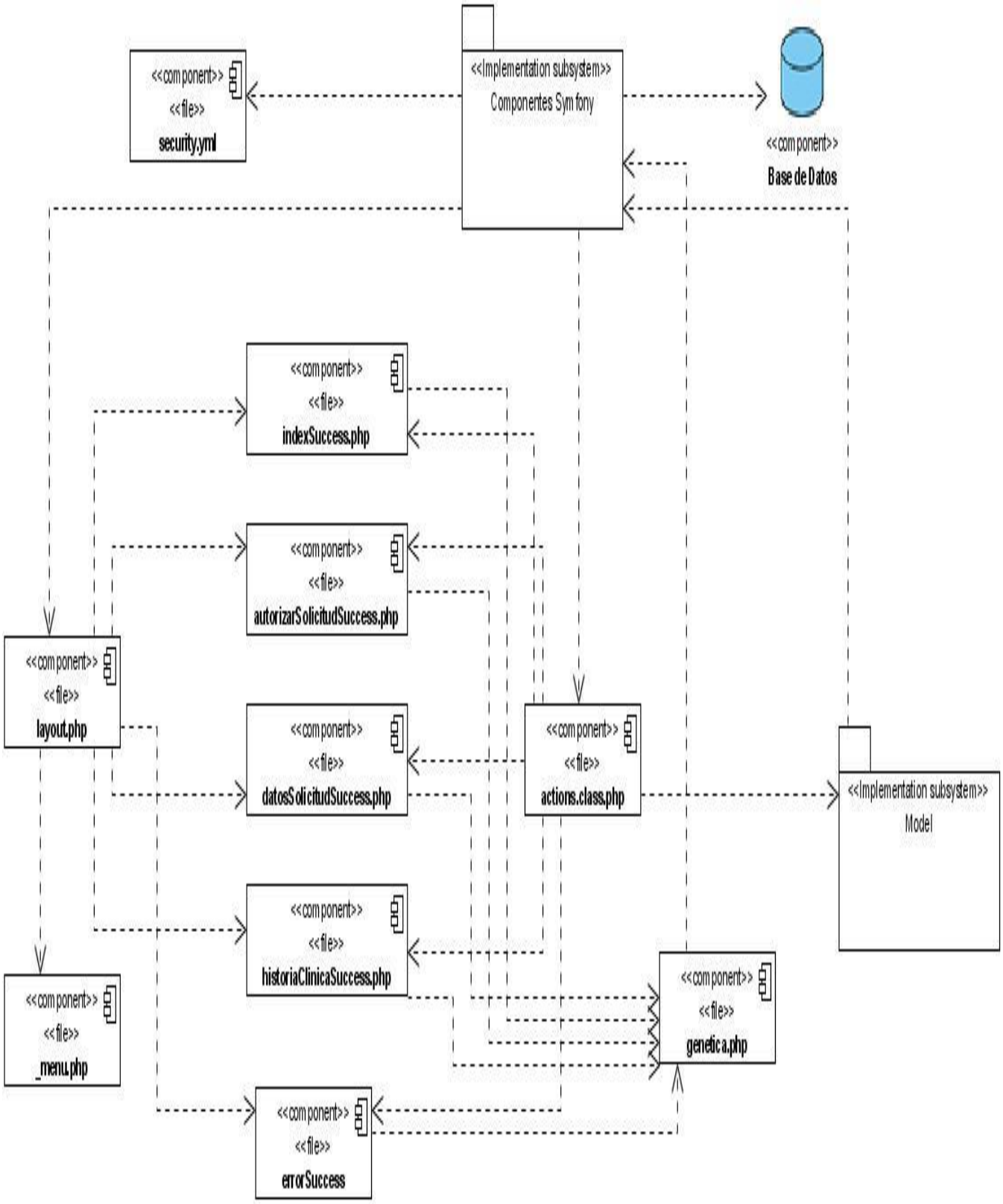


Figura 3. 3: Diagrama de componentes del CU: "Autorizar solicitud".

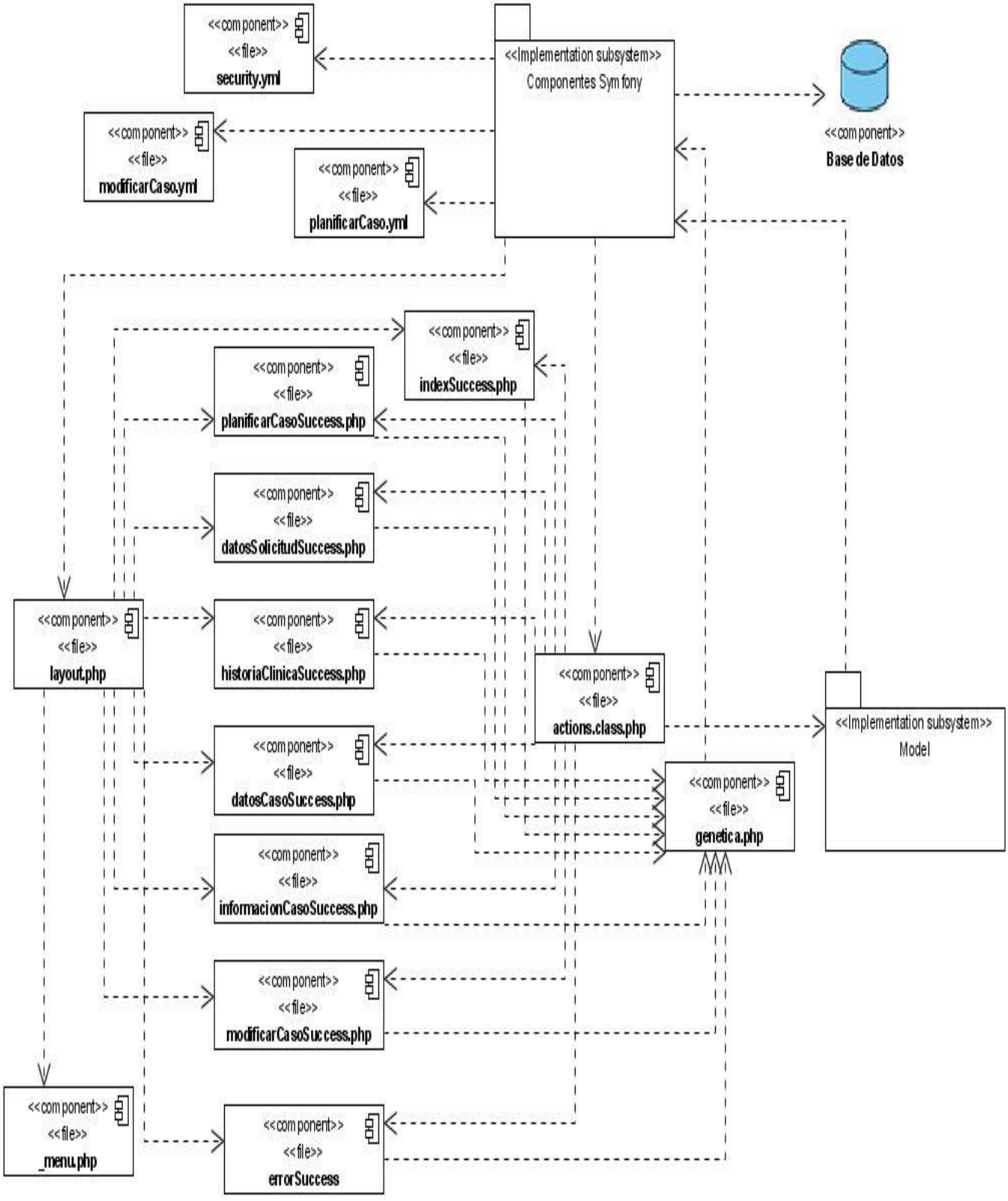


Figura 3. 4: Diagrama de componentes del CU: "Gestionar caso a discutir".

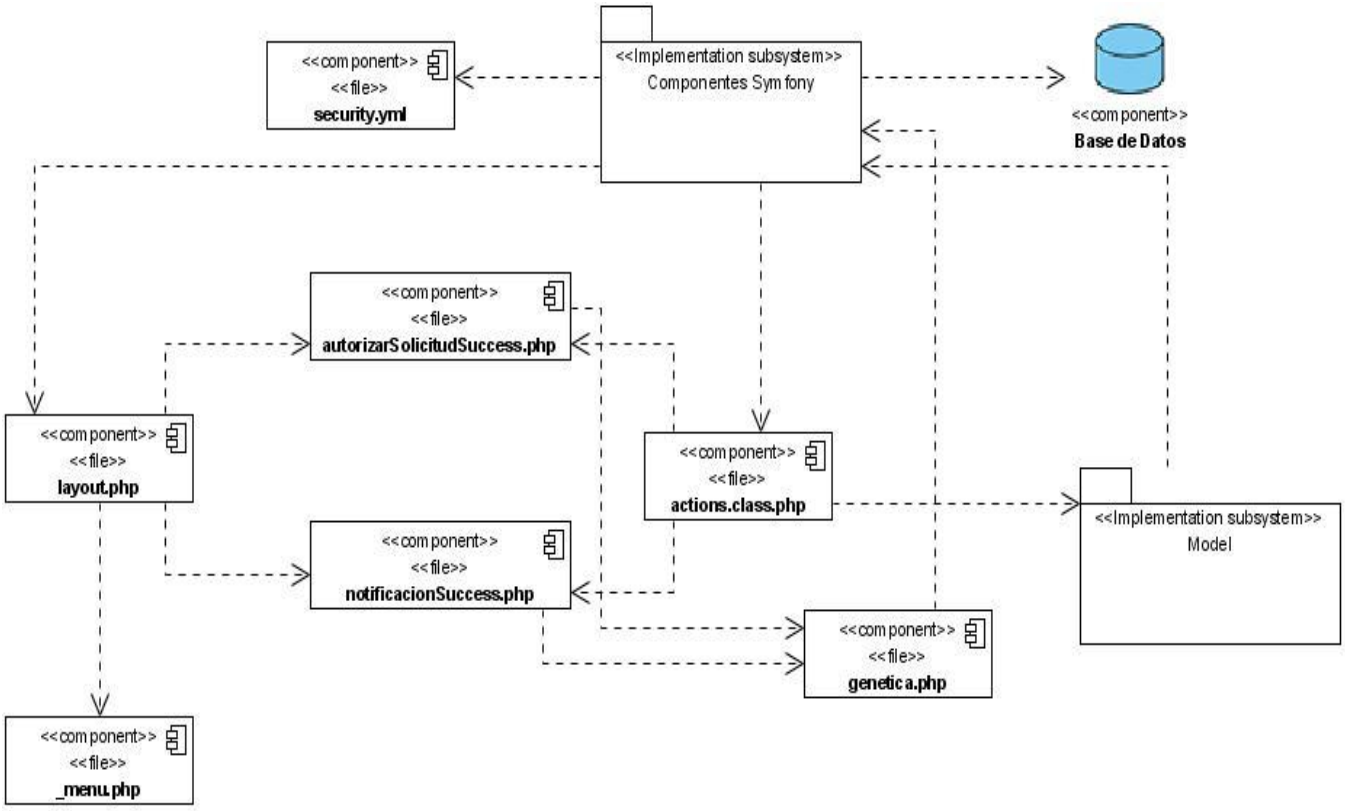


Figura 3. 5: Diagrama de componentes del CU: "Notificar negación".

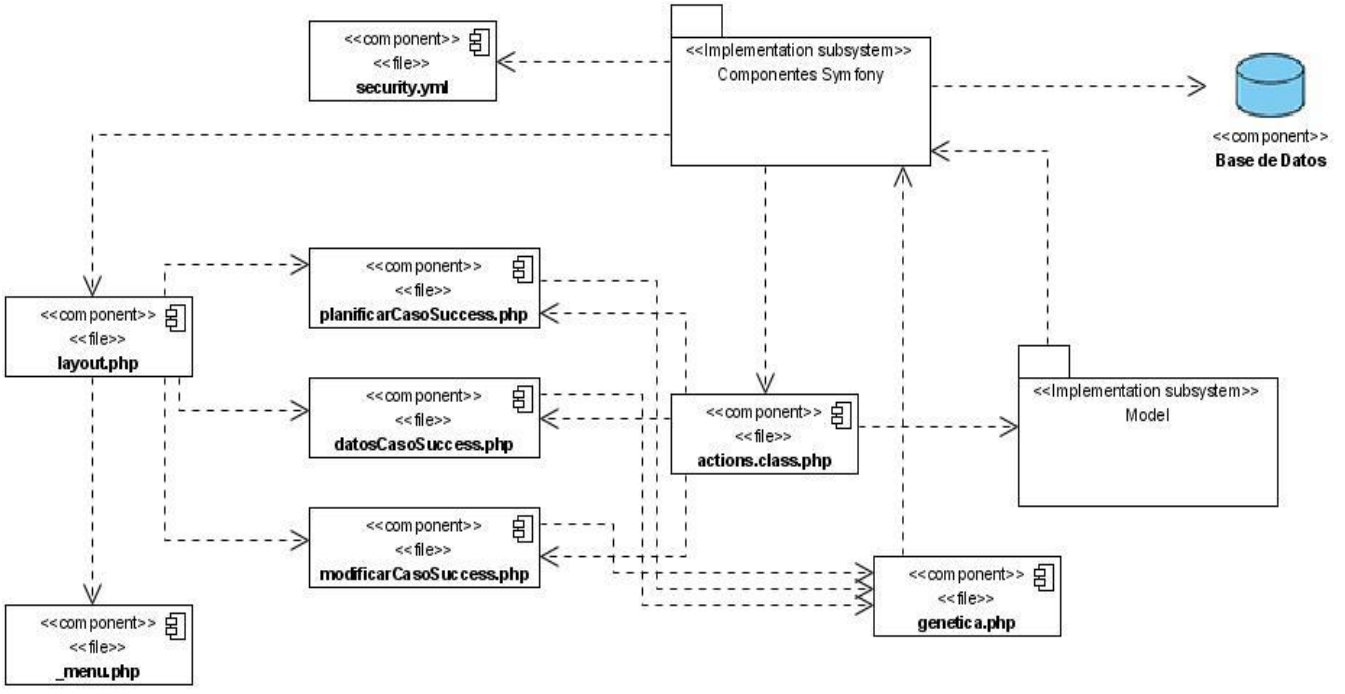


Figura 3. 6: Diagrama de componentes del CU: "Citar participante a la discusión".

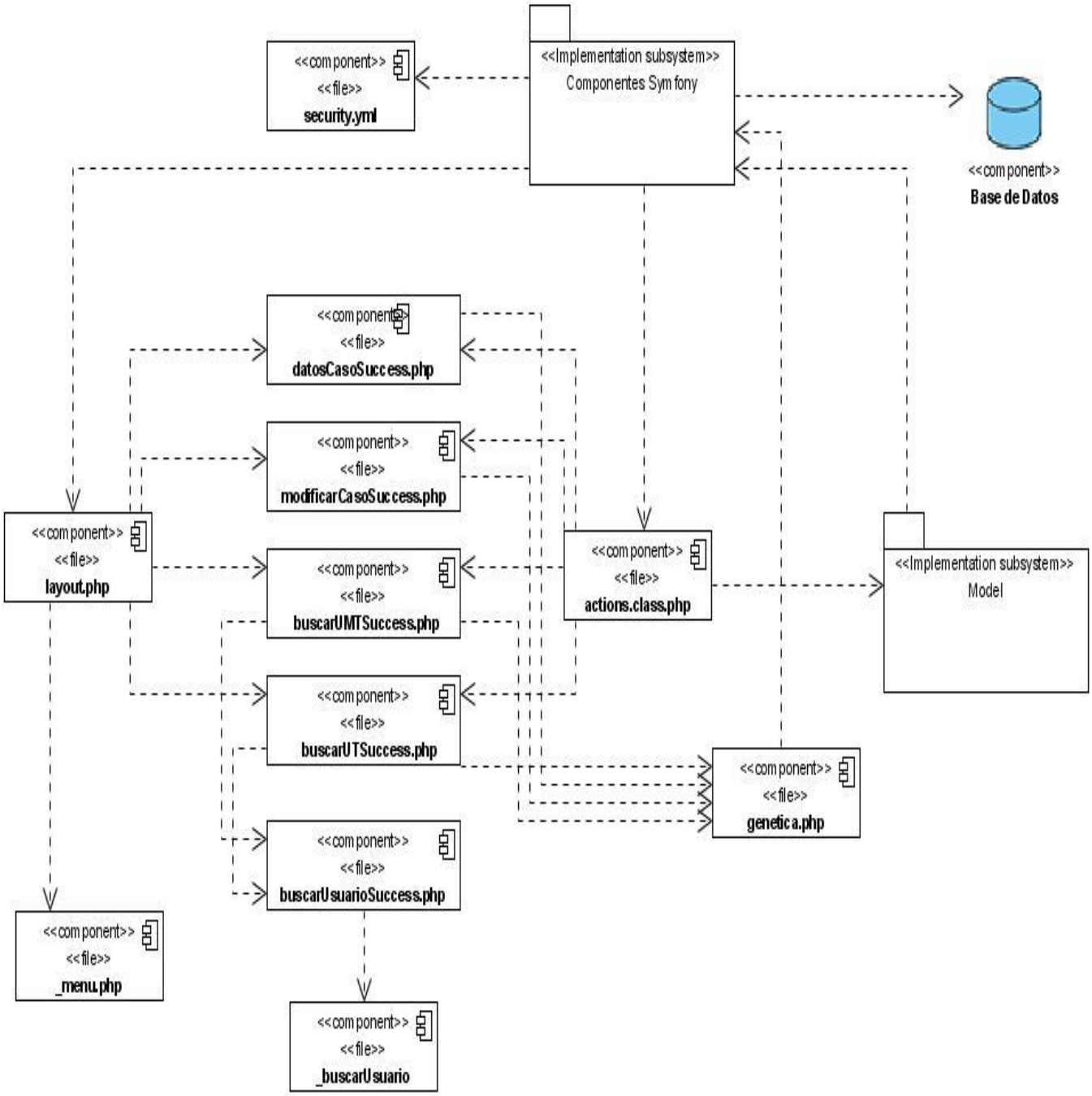


Figura 3. 7: Diagrama de componentes del CU: "Gestionar participante a la discusión".

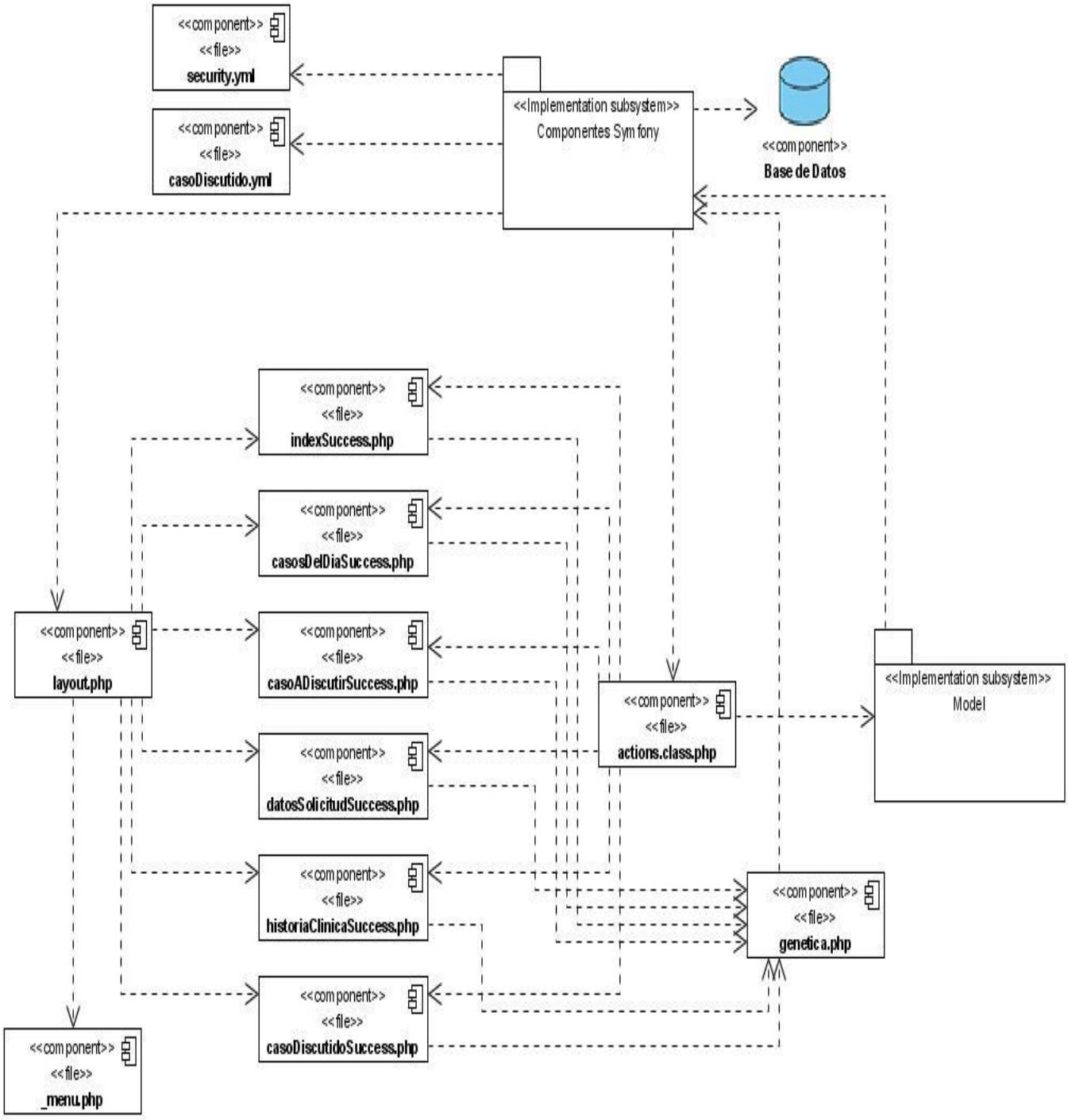


Figura 3. 8: Diagrama de componentes del CU: "Participar en discusión de un caso".

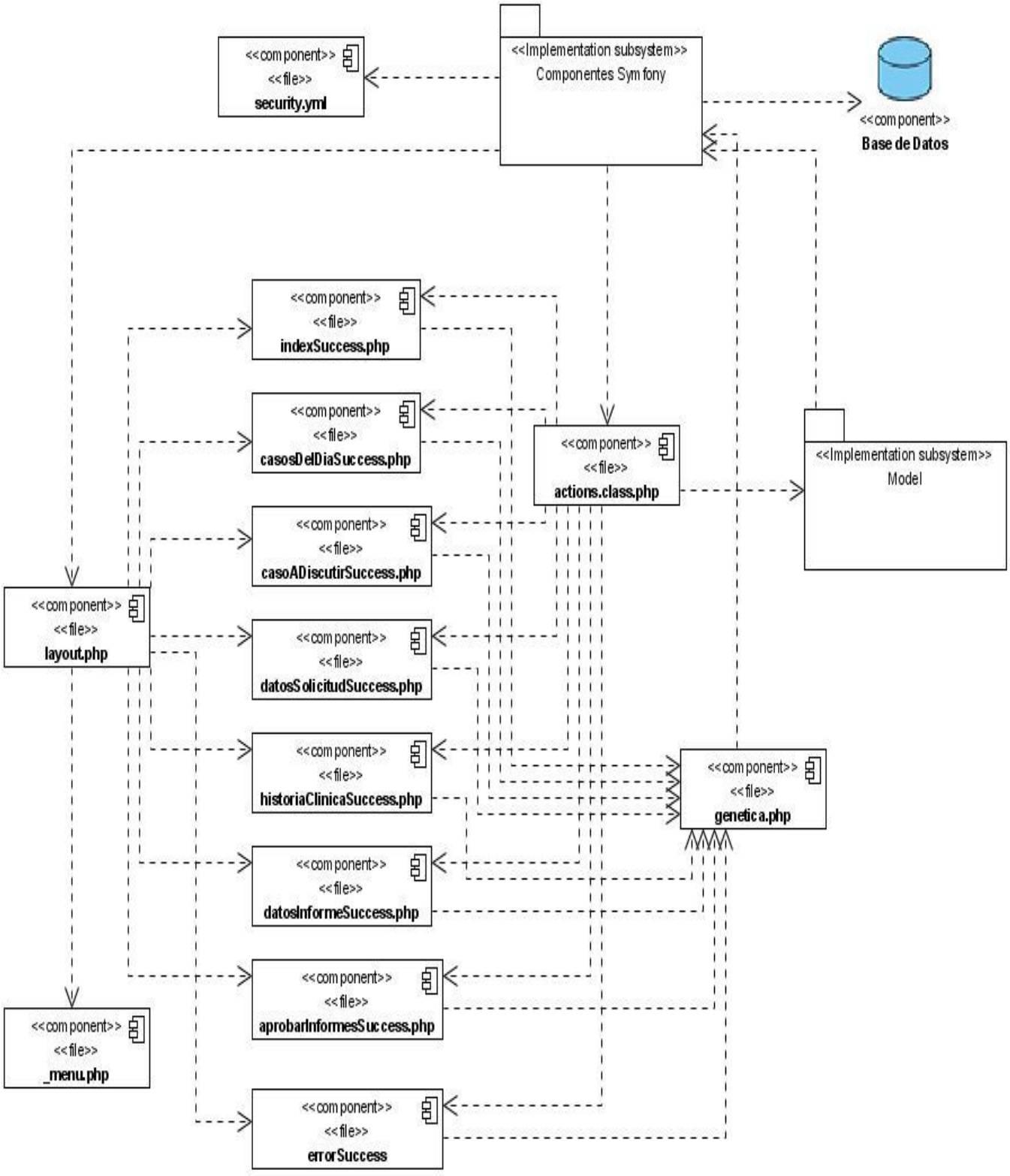


Figura 3. 6: Diagrama de componentes del CU: "Administrar caso en discusión".

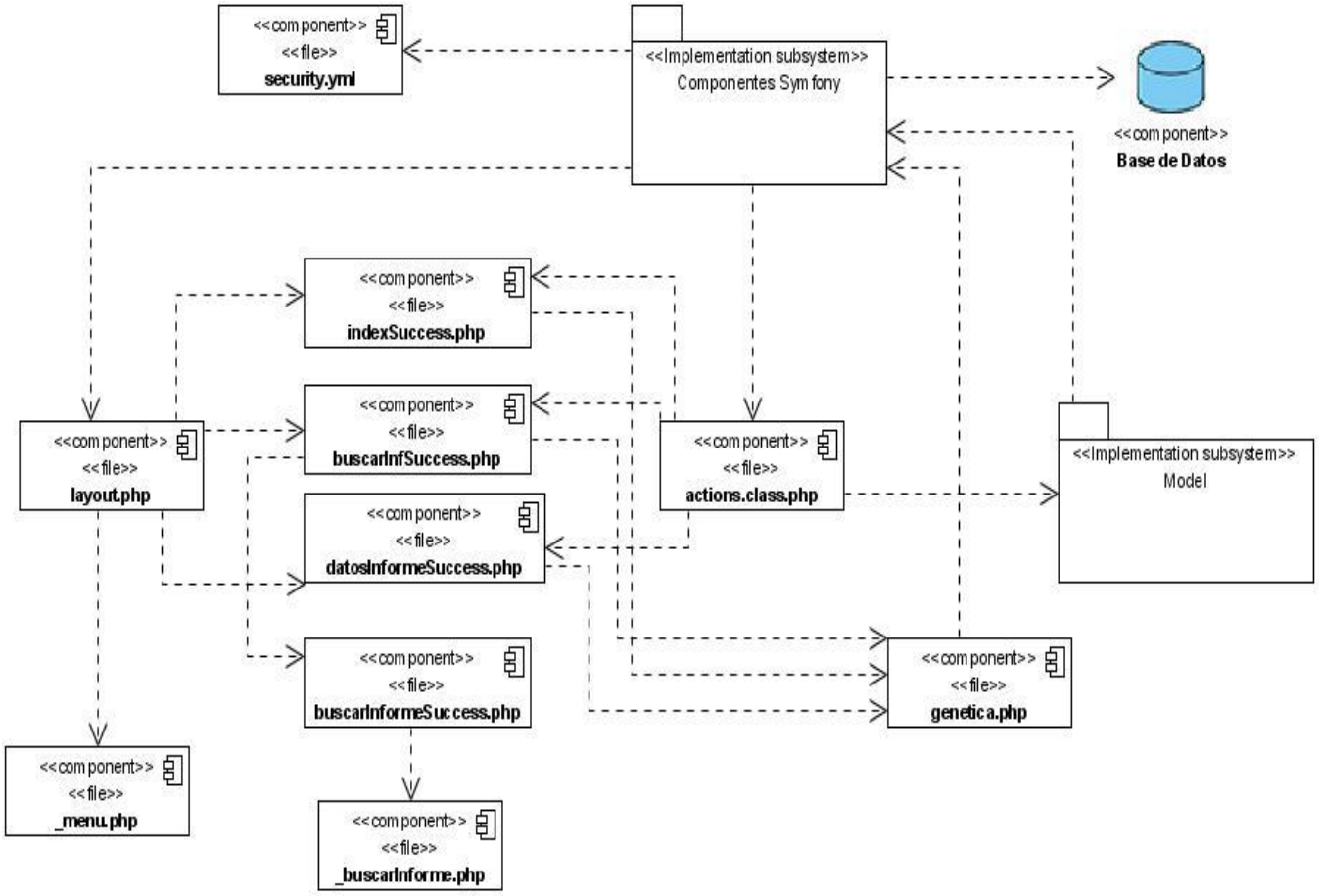


Figura 3. 7: Diagrama de componentes del CU: "Visualizar informe".

3.3 Estándar de codificación.

Los lenguajes de programación son herramientas que permiten crear programas y software. La manera de escribir código es variable por los programadores, especialmente en PHP, lo que provoca que la próxima persona que trabaje con los módulos, opinará que no es ordenado y de esta forma se atrasará el trabajo. Un estándar de codificación comprende los aspectos de la generación de código y repercute directamente en la legibilidad y la extensibilidad de cualquier proyecto de software, haciendo que nuevos desarrolladores se acoplen rápidamente al proceso de desarrollo. Además son reglas específicas a una lengua que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Los estándares de codificación no destapan problemas existentes, evitan más bien que los errores ocurran. Si bien los programadores deben implementar un estándar de forma

prudente, éste debe tender siempre a lo práctico, por estas razones siempre un proyecto debe definir los estilos de codificación, para así tener una estandarización en el proyecto.

Estilo de codificación para el SIGM

1. Todas las etiquetas php deben ser completas (<?php?>)... no reducidas (<? ?>).
2. Todas las variables deberían ser inicializadas o, al menos, comprobada su existencia utilizando isset () antes de ser utilizadas.
3. La sangría del texto debe ser siempre de 4 espacios. No utilices el tabulador (todos los editores no interpretan el Tab de la misma manera).
4. Los nombres de las variables y funciones tienen que ser siempre fáciles de leer, procurando que sean palabras en minúsculas con significado claro. Si realmente necesita más de una palabra, póngalas juntas, poniendo la inicial de cada palabra en mayúscula siempre que no sea la primera, pero procure mantenerlas tan breves como sea posible. Utilice nombres en plural para arreglos o matrices de objetos. Ejemplos: $edad, $fechaNacimiento, $personas.
5. Los bloques de código siempre deben estar encerrados por llaves (incluso si sólo constan de una línea) y correctamente alineados (a 4 espacios).
6. Las cadenas tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.
7. Todas las funciones y clases deben estar comentariadas. Los comentarios deben ser añadidos de forma que resulten prácticos, para explicar el flujo del código y el propósito de las funciones o variables. [29]

3.4 Código fuente de los principales métodos y su descripción.

No se explicarán aquí los métodos de gestión usualmente utilizados, entiéndase por estos, insertar, modificar, eliminar y recuperar registros de una BD, puesto que ello constituye sólo la base para la implementación de cualquier sistema de gestión de datos y se entiende que el lector debe tener conocimientos básicos sobre programación. Se hará referencia por tanto, a los métodos que por su singularidad sobresalen entre el resto, y entre ellos se encuentran los relacionados con los que implementan el caso de uso Caso a Discutir.

Para ello se ha seleccionado el código que compone el cuerpo de la acción de la página CasoADiscutir, que es donde precisamente se lleva a cabo la sala de conferencia. Para una mejor comprensión se ha separado el código en secciones lógicas donde se realizan distintas operaciones que se explican a continuación:

```
public function executeCasosADiscutir()
    {
        SECCIÓN 1
        $this->idCaso = $this->getRequestParameter('idCaso');
        $this->SalaConferencia = TlSalaConferenciaPeer::retrieveByPK($this->idCaso);
        $this->Solicitud = TlSolicitudPeer::retrieveByPK($this->SalaConferencia->getIdSolicitud());
        $this->discusion = $this->Solicitud->getDiscusion();
        $this->Medico = DgMedicoPeer::retrieveByPK($this->Solicitud->getIdMedico());
        $this->municipio = DgNmunicipioPeer::retrieveByPK($this->Medico->getIdMunicipio());
        $this->provincia = DgNprovinciaPeer::retrieveByPK($this->municipio->getIdProvincia());

        SECCIÓN 2
        if($this->getRequestParameter('operator')=='close')          //SE PULSADO EL BOTON TERMINAR
            {
                $this->Solicitud->setIdEstado($this->getRequestParameter('estado'));
                $this->Solicitud->setDiscusion(2);
                $this->Solicitud->save();


                if($this->getRequestParameter('estado') == 3)        //HA SIDO POSPUESTO
                    $this->redirect('tl/modificarCaso?id='.$this->idCaso.'&pgm=planificar');
                else
                    $this->redirect('tl/casosDelDia');               //SE REDIRECCIONA PARA OTRA PAGINA
            }

        if($this->getRequestParameter('pgm') == 'resumen')           //EL SOLICITANTE HA PULSADO EL VINCULO DE RESUMEN
            {
                $this->redirect('tl/casoDiscutido?idCaso='.$this->idCaso);
            }
```

Figura 3. 8: Sección 1-2 del método executeDatosADiscutir()

En la sección 1 del código que se muestra en la figura 3.10 se realiza la captura del identificador de la sala de conferencia enviado a la página mediante la función getRequestParameter(´Parametro´), a continuación con este dato se recuperan de la BD la sala de conferencia correspondiente, la solicitud y demás datos necesarios para la correcta ejecución de la página. Por su parte la sección 2 verifica de que manera fue llamada o abierta la página. Esto es muy interesante porque en cualquier otro caso lo natural es verificar simplemente el método que se envió, si fue por POST (en el caso que se envíe un formulario) o no, pero en este caso no puede ser así, porque al tratarse de un chat, que además se

desea alcance tiempos razonables de respuesta, se decidió utilizar formularios ‘ajax’ que permiten el envío de formularios sin refrescar la vista y respuestas muy rápidas.

Al inicio de la sección se verifica si el parámetro operador (campo oculto en la vista) tiene valor “close”, en dicho caso se cambia el estado de la solicitud según el valor del parámetro estado seleccionado por el moderador del caso, igualmente se cambia el campo discusión de la solicitud a 2, lo que indica que esa solicitud ha sido discutida. Más adelante se verifica si el estado seleccionado ha sido “posponer”, en cuyo caso se redirecciona el navegador hacia la página “ModificarCaso”.

Por último, si se trata del solicitante el que está autenticado en el sistema, tiene la posibilidad, si aún no lo ha hecho de realizar el resumen del caso, para ello se debe verificar si el parámetro “pgn” fue pasado con valor “resumen”, en cuyo caso se redirecciona el navegador hacia la página “Resumen”.

SECCIÓN 3

```
        $this->cartel = "";
        if(strtotime($this->SalaConferencia->getHoraFinSala()) > time('h:mm') && ($this->discusion == 0)){
            $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td>Este caso estar&aacute;
disponible el d&iacute;a '.$this->SalaConferencia->getFechaSala().' desde las '.$this->SalaConferencia->getHoraInicioSala().'
horas hasta las '.$this->SalaConferencia->getHoraFinSala().' horas</td></tr>';
            }

        if(($this->discusion == 0) && strtotime($this->SalaConferencia->getHoraFinSala()) < time('h:mm'))
            $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td>Este caso ha expirado</td></tr>';

        if(($this->discusion == 2)){
            if($this->esSolicitante($this->getUser()->getIdMedico(),$this->idCaso) && !($this->existeInforme($this->
SalaConferencia->getIdSolicitud())))
                $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td>Usted como solicitante debe
realizar el resumen de la discusi&oacute;n <a href='.$this->idCaso.'/pgn/resumen>Aqu&iacute;</a></td></tr>';;

                if($this->getUser()->hasGroup('tl_admin_n') && ($this->Solicitud->getIdEstado() == 2)){
                    $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td><input name="estado"
type="radio" value="1" checked> Resuelto <input name="estado" type="radio" value="3"> Posponer <input
type="submit" value="Aceptar" class="sbttn"></td></tr>';;
                }
                else if((($this->getUser()->hasGroup('tl_admin_p') || $this->esModerador($this->getUser()->getIdMedico(),$this->
idCaso))) && ($this->Solicitud->getIdEstado() == 0)){
                    $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td><input name="estado"
type="radio" value="1" checked> Resuelto <input name="estado" type="radio" value="2"> Remitir <input
name="estado" type="radio" value="3"> Posponer <input type="submit" value="Aceptar" class="sbttn"></td></tr>';
                }

                if(($this->Solicitud->getIdEstado() > 0 ))
                {
                    if($this->esSolicitante($this->getUser()->getIdMedico(),$this->idCaso)){
                        $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td>Usted como
solicitante debe realizar el resumen de la discusi&oacute;n <a href='.$this->idCaso.'/pgn/resumen>Aqu&iacute;</a></td></tr>';
                    }
                    else{
                        $this->cartel = '<tr id="cartel" align="center" class="nombrecampo" height="30"><td>Este caso ha sido
cerrado</td></tr>';
                    }
                }

            }
```

Figura 3. 9: Sección 3 del método executeDatosADiscutir()

El propósito de la sección número 3 es mostrar botones y mensajes en dependencia del rol del usuario que esté autenticado en el sistema. Estos roles pueden ser los de administradores provinciales o nacionales, moderadores (dentro de los que se encuentran los administradores), solicitante y participante. Tanto moderador, solicitante y participante son roles temporales, ya que para un caso usted puede ser un moderador pero para otro puede ser un participante. Por su parte los administradores son roles permanentes, ellos siempre mantienen su condición, y tienen la posibilidad de participar en todos los casos de sus provincias como moderadores siempre que lo deseen. Al inicio de esta sección se pretende construir un mensaje informativo para el usuario. Inicialmente se verifica

que la hora planificada para que el caso concluya no haya pasado y que la discusión no haya comenzado. Si se verifican estas condiciones el mensaje será informando al usuario la fecha y hora en que comenzará la discusión.

La siguiente condición verifica si el caso no ha sido discutido y la hora planificada ha pasado, en cuyo caso se muestra un mensaje de que el caso ha expirado. La próxima condición verifica varias opciones, todas incluidas en el caso que la discusión ya haya concluido, lo cual lo indica si la variable "discusión" tiene el valor 2. Si el usuario autenticado es el Solicitante (lo cual se verifica con el método esSolicitante(idMédico,idSala)) y no ha realizado el informe del caso (existeInforme(idSolicitud)) se le muestra un mensaje explicándole la necesidad de realizar dicho informe y un vínculo para dicha página.

El siguiente caso es si el que está autenticado es un administrador nacional ($this->getUser->hasGroup('nacAdminTl')), en cuyo caso se muestran las opciones de posponer y terminar el caso. La siguiente variante es que el administrador sea provincial o se trate de un moderador (esModerador(idMédico,idCaso)), en este caso se le muestran las mismas opciones del administrador nacional adicionando la posibilidad de remitir el caso.

Por último, si se ha pasado por todas las verificaciones anteriores con resultado negativo es porque el caso ha sido cerrado y no se puede realizar más opciones sobre él, emitiéndose el mensaje correspondiente.

SECCIÓN 4

```
            if((strtotime($this->SalaConferencia->getHoraInicioSala()) > time('h:mm') || strtotime($this->SalaConferencia->
getHoraFinSala()) < time('h:mm')) && ($this->discusion == 2))
                $this->fuera_de_tiempo = true;
                else $this->fuera_de_tiempo = false;

            if(($this->getUser()->hasGroup('tl_admin_n')) || ($this->getUser()->hasGroup('tl_admin_p')) || (($this->esModerador(
$this->getUser()->getIdMedico(),$this->idCaso)) == 1) && ($this->discusion != 2)){

                if((strtotime($this->SalaConferencia->getHoraInicioSala()) > time('h:mm')) || (strtotime($this->SalaConferencia->
getHoraFinSala()) < time('h:mm')))
                {
                    $disabled = 'disabled';
                    $disabled2 = '';
                }
                else
                    {
                        $disabled = '';
                        $disabled2 = 'disabled';
                    }
                if($this->discusion != 0){
                    $disabled = 'disabled';
                    $disabled2 = '';
                }

                $this->iniciar = '<input type="submit" name="iniciar" id="iniciar" class="sbttn"
onClick="javascript:IniciarChat()" value="Iniciar Discusion"'.$disabled.'>';

                $this->terminar = '<input type="submit" name="terminar" id="terminar" class="sbttn"
onClick="javascript:TerminarChat()" value="Terminar Discusion"'.$disabled2.'>';

            }
            else{
                $this->iniciar = "";
                $this->terminar = "";
            }

    }
```

Figura 3. 10: Sección 4 del método executeDatosADiscutir()

La sección 4 y última de esta función realiza una tarea no menos importante que las secciones anteriores. Al igual que en la sección 3 se verificaban los datos del usuario autenticado y el estado de la solicitud y la discusión, en esta sección se realiza algo similar, pero esta vez para garantizar que a cada usuario se le dé la posibilidad de hacer lo que a cada cual le corresponde. Inicialmente se hace una verificación trivial que consiste en verificar que la discusión aún está en tiempo, en caso de no estarlo, en la vista se mostrará el mensaje apropiado indicándoles a los usuarios que este caso ha expirado por tiempo. Hacer la salvedad que se utiliza la función strtotime() que recibe una cadena, ya que le función time() con la que se compara devuelve un elemento de tipo 'timestamp' incompatible con una cadena de texto como por ejemplo la hora de inicio de la discusión.

Seguidamente se procede a verificar el rol del usuario que está autenticado mediante la sesión de usuario como ya se explicó anteriormente. Puede suceder que el usuario sea administrador nacional,

provincial o simplemente estar desempeñando el rol de moderador. En cada uno de estos casos tendrán la posibilidad de contar con 2 botones, uno de iniciar la discusión y otro de terminar la discusión. Para el resto de los participantes en la discusión estos elementos no existirán en sus vistas.

Adicionalmente se tiene en cuenta si la discusión ha sido iniciada, en cuyo caso el botón iniciar debe estar deshabilitado. Igualmente ocurre con el botón terminar que debe estar deshabilitado cuando aún no se ha seleccionado el botón iniciar. En resumen el objetivo de esta acción es preparar a su plantilla correspondiente para visualizarla al usuario en dependencia del rol que este tenga. El resto de las acciones que realiza son de enrutamiento básico. La otra acción, estrechamente relacionada con la anterior, es executeChatEnviarMensaje(). Se puede incluso decir que esta, en conjunto con los 'helpers' de 'ajax' utilizados constituyen el pilar fundamental de la implementación del chat. Para poder comprender como funciona el sistema en torno a esta función es necesario definir algunos elementos utilizados en el template CasosADiscutir.

Integración con Ajax

'Ajax' se ha convertido en la forma más elegante y rápida de producir aplicaciones interactivas. Permite a las páginas hacer una pequeña petición de datos al servidor y recibirla sin necesidad de cargar la página entera. Por si fuera poco 'ajax' funciona en cualquier navegador, y es perfectamente compatible con cualquier tipo de servidor estándar y lenguaje de programación Web: PHP, ASP. ASP.Net, Perl, JSP, Cold Fusion. Su implementación por lo general no es demasiada complicada. Para el trabajo con 'ajax', Symfony ha incluido en sus librerías varios 'helpers' para el manejo de peticiones 'ajax', entre los más significativos, y además utilizados en este sistema se encuentran:

- form_remote_tag(): Este 'helper' crea una etiqueta igual que como la creada por el 'helper' form_tag(), sin embargo el envío del formulario se realiza en segundo plano, de manera que se construya invisible para el usuario. Su declaración general tiene el aspecto siguiente:

```
<?php echo form_remote_tag(array('update' => 'ventana',
'url' => 'tl/chatEnviarMensaje',
)) ?>
```

El envío consiste en realizar una petición de tipo POST a la acción localizada en tl/ChatEnviarMensaje, la respuesta del servidor reemplazará el contenido del elemento del formulario cuyo id sea ventana.

- periodically_call_remote(): este 'helper' realiza un llamado a una función remota cada cierto tiempo que se le especifique. Si se piensa un momento en esto, es fácil darse cuenta que en este 'helper' radica la lógica del chat.

```
<?php echo periodically_call_remote(array( 'frequency' => 2, 'update' => 'ventana',
    'url' => 'tl/chatEnviarMensaje', 'with' => '"idCaso=' + document.getElementById('idCaso')"
)) ?>
```

Al igual que el 'helper' form_remote_tag() este realiza una petición 'ajax' en segundo plano, con la diferencia que no envía el formulario y además permite especificar el parámetro 'frecuency' que será la frecuencia con la que se ejecutará la función. Como este 'helper' no envía el formulario puede ser necesario pasarle a la función remota algún parámetro que sea útil para su ejecución para ello se especifica el parámetro 'with'. Una vez presentados los 'helper' utilizados en la vista para llamar funciones remotas en segundo plano, se pasará a la explicación de la función executeChatEnviarMensaje(). Como ya se puede deducir la tarea de esta función consiste en devolver una respuesta a los que realicen peticiones a ella, de manera que esta respuesta pueda ser incluida en dicha plantilla sin que la descomponga, por lo tanto una de las primeras observaciones es que esta acción no posee 'template', puesto que lo único que devuelve es un pequeño fragmento de código .html que se incluye dentro de otra página.

SECCIÓN 1

```
public function executeChatEnviarMensaje()
    {
        $this->idCaso = $this->getRequestParameter('idCaso');
        $color = $this->getRequestParameter('color');

        if($this->getRequestParameter('operator')=='init')          //APRETARON EL BOTON INICIAR DISCUSION
            {
                $this->InicializarChat();                           //INICIALIZAMOS EL HISTORIAL
            }

        if($this->getRequestParameter('operator')=='finish')        //APRETARON EL BOTON TERMINAR DISCUSION
            {
                $this->TerminarChat();                              //ACTUALIZAMOS LOS DATOS DE LA SOLICITUD
            }

        $this->SalaConferencia = TlSalaConferenciaPeer::retrieveByPK($this->idCaso);
        $this->Solicitud = TlSolicitudPeer::retrieveByPK($this->SalaConferencia->getIdSolicitud());
```

Figura 3. 11: Sección 1 del método executeChatEnviarMensaje()

En la sección 1, como generalmente ocurre al inicio de las acciones, se procede a la verificación del origen de la petición. Este caso tiene la particularidad que dicho origen varía en dependencia de

parámetros que son enviados a la acción y no precisamente del origen de la petición (POST, GET, XMLHttpRequest). Por lo tanto en dependencia de los valores que tomen estos parámetros, en este caso el parámetro 'operator' será la acción a realizar por la función. Luego se procede a la recuperación en la BD de los datos de la sala de conferencia y de su solicitud.

SECCIÓN 2

```
if($this->getRequestParameter('operator')=='add' && $this->Solicitud->getDiscusion() != 2){
//SE HA ENVIADO UN NUEVO MENSAJE Y NO HA TERMINADO LA DISCUSION

    $usuario = $this->getUser()->getUsername();  //OBTIENE EL NOMBRE USUARIO QUE ESTA LOGUEADO
    $medico = $this->getMedicoByUser($usuario);  //DVUELVE UN MEDICO QUE PERTECE AL UN USUARIO

    //LOCALIZAMOS AL PARTICIPANTE QUE ESTA LOGUEADOY ESTA EN EL LISTADO DE PARTICIPANTES DE ESTA SALA
    $participante = $this->getParticipanteSalaByIdMedicoIdSala($this->getMedicoByUser
    ($this->getUser()->getUsername())->getIdMedico(),$this->idCaso);

    //PUEDE SER QUE SEA UN ADMINISTRADOR Y NO ESTE ADICIONADO A LA DISCUSION
    if(!$participante && (($this->getUser()->hasGroup('nacAdminTl')) || ($this->getUser()->hasGroup
         ('provAdminTl')))){
        //LO CREAMOS Y LO ADICIONAMOS
        $nuevo_participante = new TlParticipanteSala();
        $nuevo_participante->setIdSalaConferencia($this->idCaso);
        $nuevo_participante->setIdMedico($this->getMedicoByUser($this->getUser()->getUsername())->
          getIdMedico());
        $nuevo_participante->save();
        $participante = $this->getParticipanteSalaByIdMedicoIdSala($this->getMedicoByUser($this->
       getUser()->getUsername())->getIdMedico(),$this->idCaso);
    }


    if($participante->getColor() != $color){  //SI SU COLOR EN LA BD ES DISTINTO DEL COLOR PASADO
    $participante->setColor($color);  //CAMBIAMOS EL COLOR EN LA BD
    $participante->save(); //SALVAMOS LOS CAMBIOS
}


    $mensaje = $this->getRequestParameter('mensaje');
    if($mensaje != ''){
        $new_message = new TlMensaje();
        $new_message->setIdHistorial($historial->getIdHistorial());
        $new_message->setIdMedico($medico->getIdMedico());
        $new_message->setMensaje($mensaje);
        $new_message->setCreateAt( time() ); //TIME() DA LA HORA DE LA PC
        $new_message->save();  //SALVA EL MENSAJE EN LA BD
    }
}
```

Figura 3. 12: Sección 1 del método executeChatEnviarMensaje()

La sección 2 contempla el caso en que el dato enviado mediante el parámetro 'operator' sea 'add', esto indica a la acción que debe esperar recibir un mensaje y que debe guardarlo en la BD. Para que este flujo de eventos se lleve a cabo inicialmente se deben obtener los datos del usuario que está logueado, esto se logra a través de los datos de la sesión de usuario ($this->getUser()->getUserName()).

Adicionalmente se comprueba la posibilidad que dicho usuario no esté dentro de la planificación del caso y se trate de un administrador. Estos administradores tienen la posibilidad de intervenir en todos los casos que estimen conveniente. En este caso es necesario crear un nuevo participante y adicionarlo seguidamente. A continuación, independientemente de si se verifica la condición anterior, se verifica si el color pasado por el navegador es el mismo que el participante posee en la BD, en caso de ser distintos se procede a cambiar el color en la misma.

Finalmente, cuando ya se tienen los datos necesarios para crear un mensaje se captura el parámetro 'mensaje' enviado por el formulario 'ajax' y se procede a su construcción y almacenamiento a la BD.

SECCIÓN 3

```
//ESTO SE EJECUTA INDEPENDIENTEMENTE DEL OPERATOR
if($historial)
    $IdHistorial = $historial->getIdHistorial();
else
    $IdHistorial = -1;

$c = new Criteria();  //LOS MENSAJES DE LA BD CON EL ID DE HISTORIAL CORRECTO
$c->addAscendingOrderByColumn(TlMensajePeer::ID_MENSAJE);
$c->add(TlMensajePeer::ID_HISTORIAL,$IdHistorial);
$messages = TlMensajePeer::doSelect($c);
```

Figura 3. 13: Sección 1 del método executeChatEnviarMensaje()

La sección 3 realiza una operación elemental que se ejecuta sin importar el valor del parámetro ´operator´. La relevancia del código de esta sección es realizar una consulta a la BD que devuelva sólo los mensajes que pertenecen al historial abierto en cada momento. Un error en este fragmento puede ocasionar que no se muestren los mensajes correctamente en el navegador. Aclarar que los mensajes son ordenados ascendentemente por el identificador del mensaje (addAscendignOrderByColum(TLMensajePeer::ID_MENSAJE)) lo cual garantiza que estos se muestren en el orden correcto.

SECCIÓN 4

```
//CONSTRUIMOS LA RESPUESTA
$response = "";
$response.= "<table width=450>";
foreach($messages as $message){
    $response.= "<p>";
    $participante = $this->getParticipanteSalaByIdMedicoIdSala($message->getIdMedico()
    ,$this->idCaso);
    $response.="<tr><td class=grisoscuro align=left><strong><span style=color:".$participante->
     getColor().">".$message->getCreateAt('h:i:s')." ".$this->getMedico($message->getIdMedico())
    ->getUsuario()."</span></strong></td></tr>";
    $response.= "<tr><td class=valorcampo align=left><span style=color:".$participante->getColor()
    .">".$message->getMensaje()."</span></td></tr>";
    $response.= "</p>";
}

if($this->Solicitud->getDiscusion() == 2)  //SI SE ACABO LA DISCUSION
    {
        $txt = "La discusi&oacute;n de este caso ha concluido";
        //MOSTRAREMOS UN MENSAJE EN EL CHAT DE QUE SE ACABO LA DISCUSION
        if($this->esSolicitante($this->getMedicoByUser($this->getUser()->getUsername())
        ->getIdMedico(),$this->idCaso)){
            if($this->Solicitud->getIdEstado() != 3)
                $txt = "Usted como solicitante de este caso tiene la responsabilidad de realizar
                el resumen de la discusi&oacute;n, para ello de click <a href=".$this->idCaso."/pgn/
                resumen>Aqu&iacute;</a>";
            }
            else if(!$this->esModerador($this->getMedicoByUser($this->getUser()->getUsername())->
           getIdMedico(),$this->idCaso)){
                $txt = "El moderador ha cerrado la sala del chat, los mensajes ya no se
                guardar&aacute;n a partir de este momento. Se le agradece su participaci&oacute;n
                en esta discusi&oacute;n";

                }
                    $response.= "<p>";
                    $response.="<tr><td class=grisclaro align=center height=2></td></tr>";
                    $response.="<tr><td align=left>".$txt."</td></tr>";
                    $response.= "</p>";
            }

                $response.= "</table>";

            return $this->renderText($response); //RENDERTEXT CONVIARTE UN CODIGO A HTML PURO
             }
```

Figura 3. 14: Sección 1 del método executeChatEnviarMensaje()

La sección 4 es la encargada de procesar todos los mensajes recuperados por la sección 3 y enviarlos como respuesta de la petición ‘ajax’. De esta sección se puede reparar en la forma de construir esta respuesta, puesto que como lo indica la variable ‘$response’ el resultado no es más que una cadena de texto.

Por último si se verifica que la discusión ha concluido se incluyen en la respuesta mensajes para cada usuario, en dependencia de su rol, que aparecerán al final de la ventana del chat.

La respuesta final no se devuelve como cadena de texto, sino que se le aplica la función renderText() que no necesita plantilla y acelera las respuestas 'ajax' contribuyendo a disminuir el tiempo de respuesta del chat.

3.5 Interfaces de la aplicación.

Para la realización de esta aplicación se desarrolló una interfaz amigable y segura, acorde con el personal que trabajará con la misma, y para hacer más fácil la interacción de los usuarios se llevó el mismo orden de los elementos que aparecen en la planilla de la solicitud.

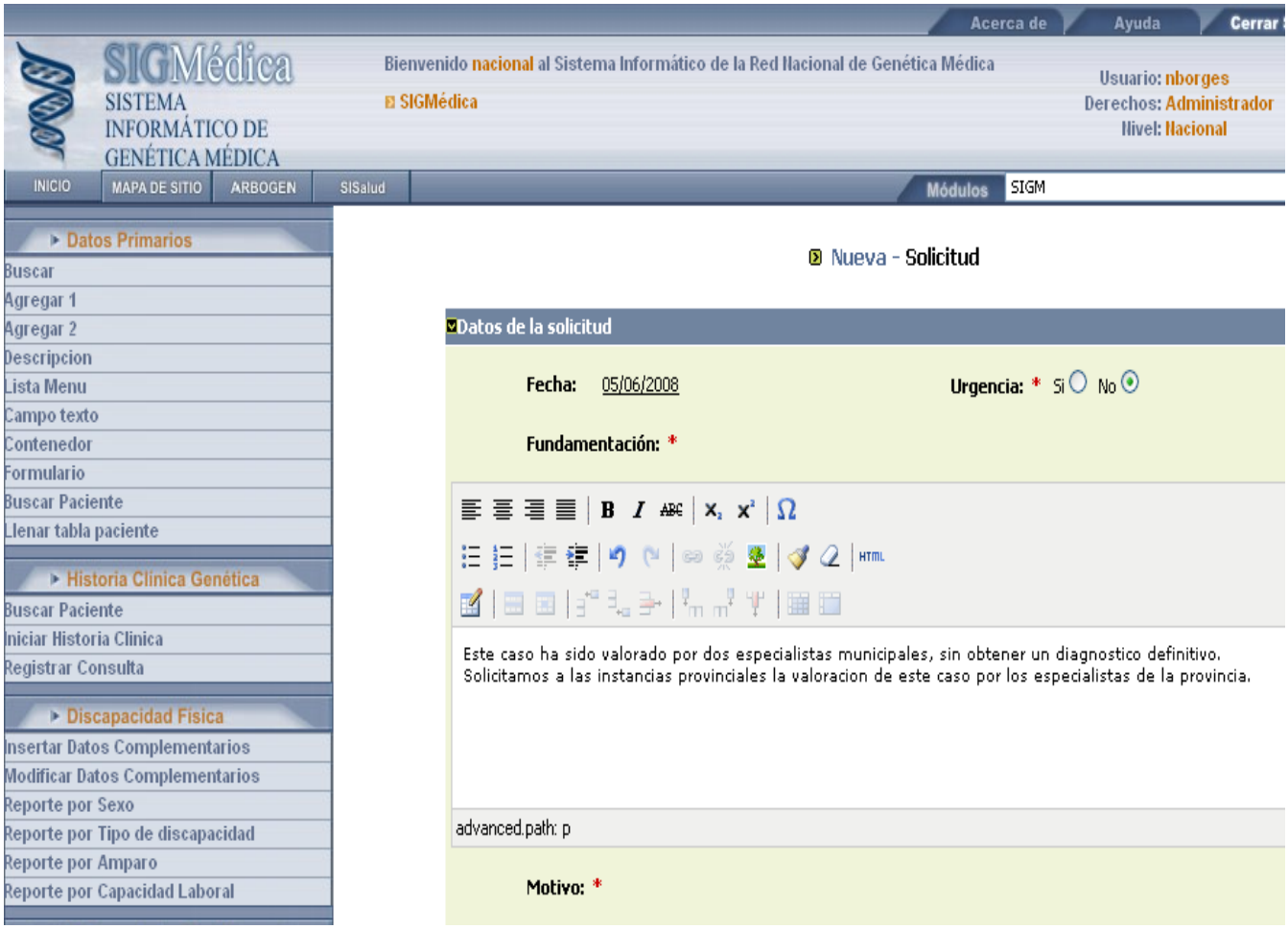


Figura 3. 15: Interfaz del CU: "Solicitar discusión de un caso"

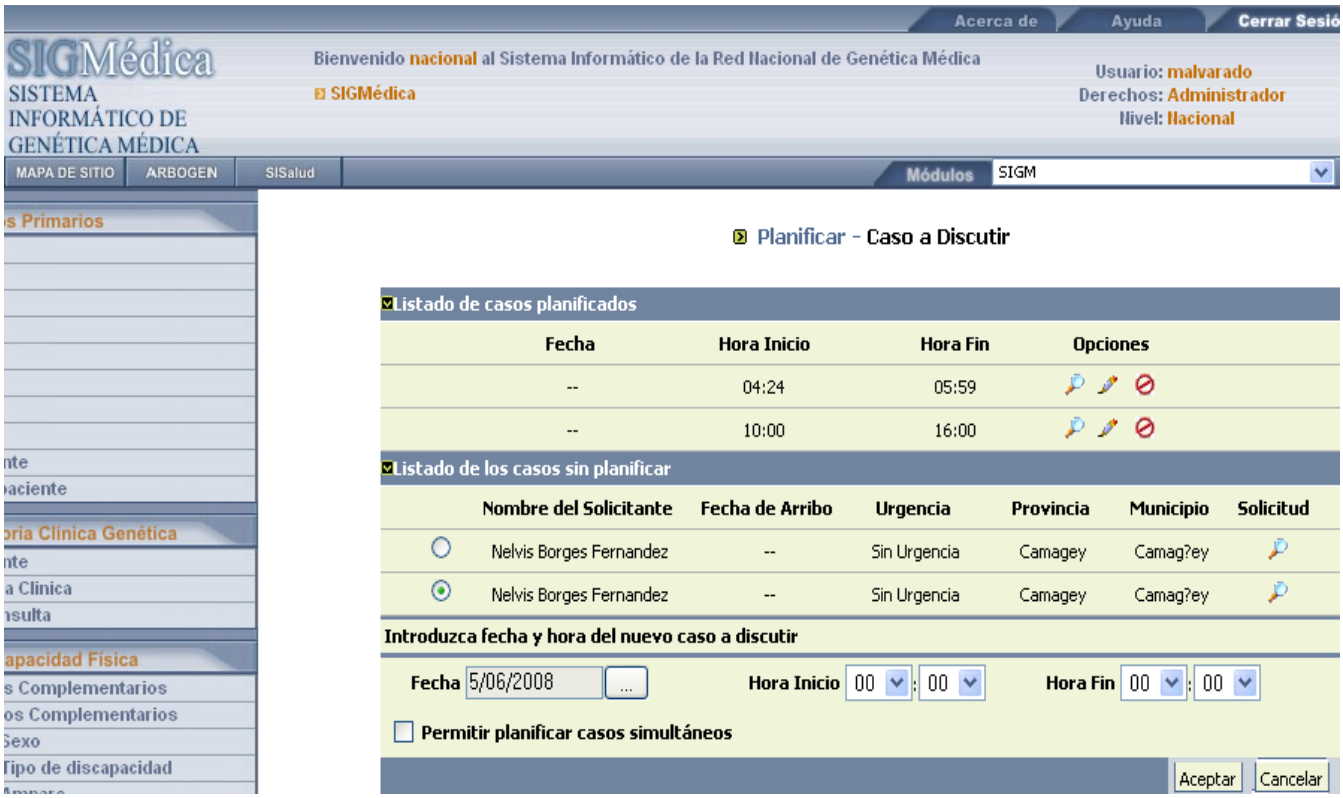


Figura 3. 16: Interfaz del CU: "Gestionar caso a discutir"

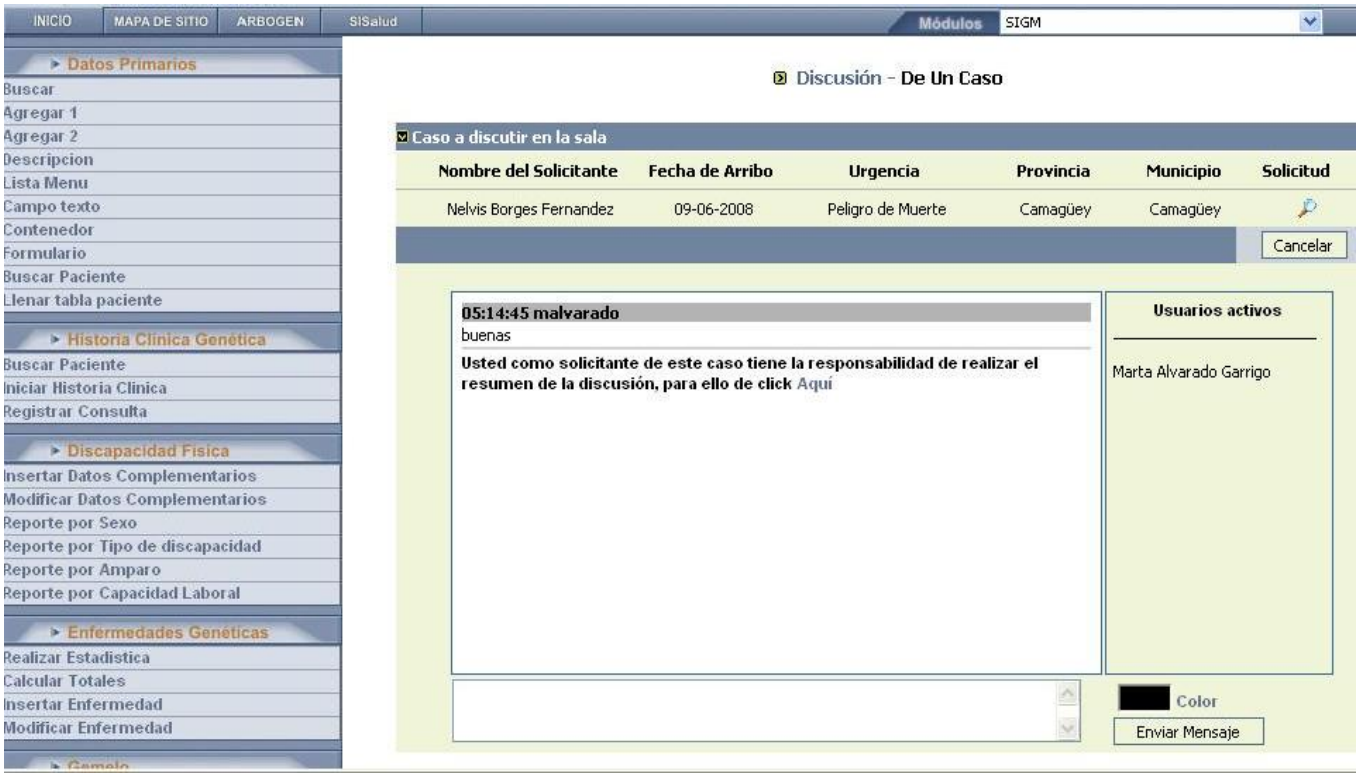


Figura 3. 17: Interfaz del CU: "Participar en discusión de un caso"

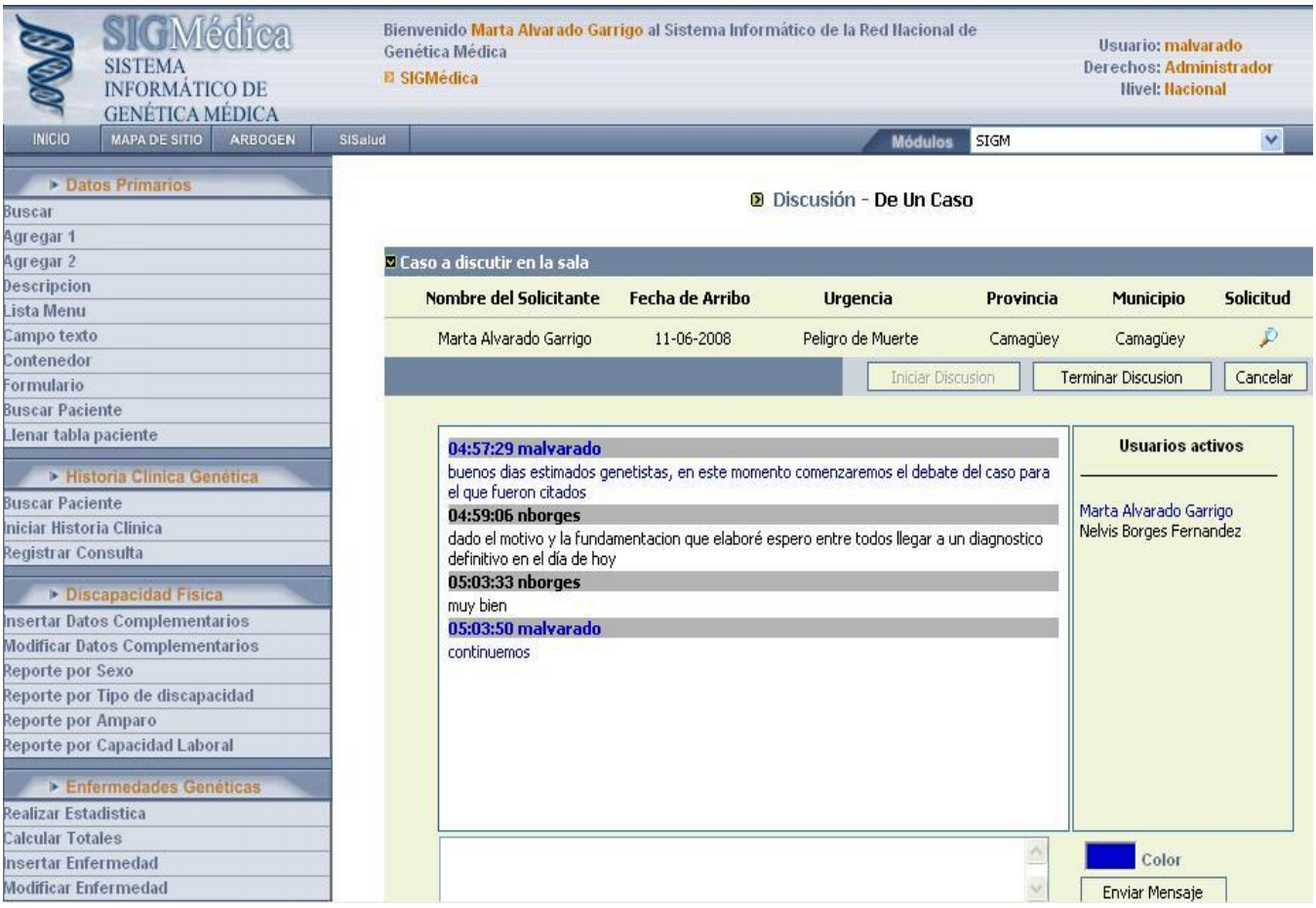


Figura 3. 18: Interfaz del CU: "Participar en discusión de un caso"

Figura 3. 19: Interfaz del CU: "Participar en discusión de un caso"

3.6 Validación de la Aplicación.

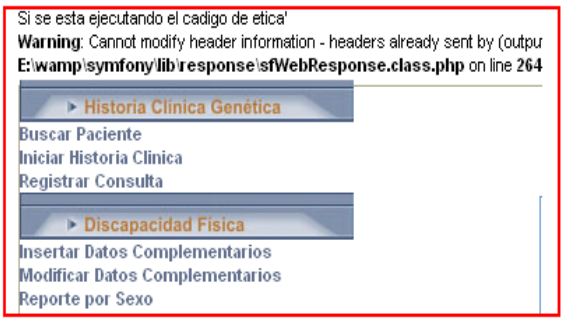
Sin darse cuenta cada desarrollador es un probador en potencia, un especialista en calidad, y lo que logra con ello es validar la aplicación sobre la marcha. Similar a la forma de redactar un pseudocódigo, cada programador tiene su estilo propio para validar sus aplicaciones y comprobar que el camino por el que se conduce lo llevará a una solución satisfactoria.

Teniendo en cuenta el lenguaje de programación utilizado y el framework de desarrollo, los cuales no cuentan con robustos compiladores en tiempo de diseño que permitan la rápida detección y mitigación de los errores se hace necesaria la aplicación de los conocimientos, la experiencia y la perspicacia del programador para detectar estos errores de la forma más rápida.

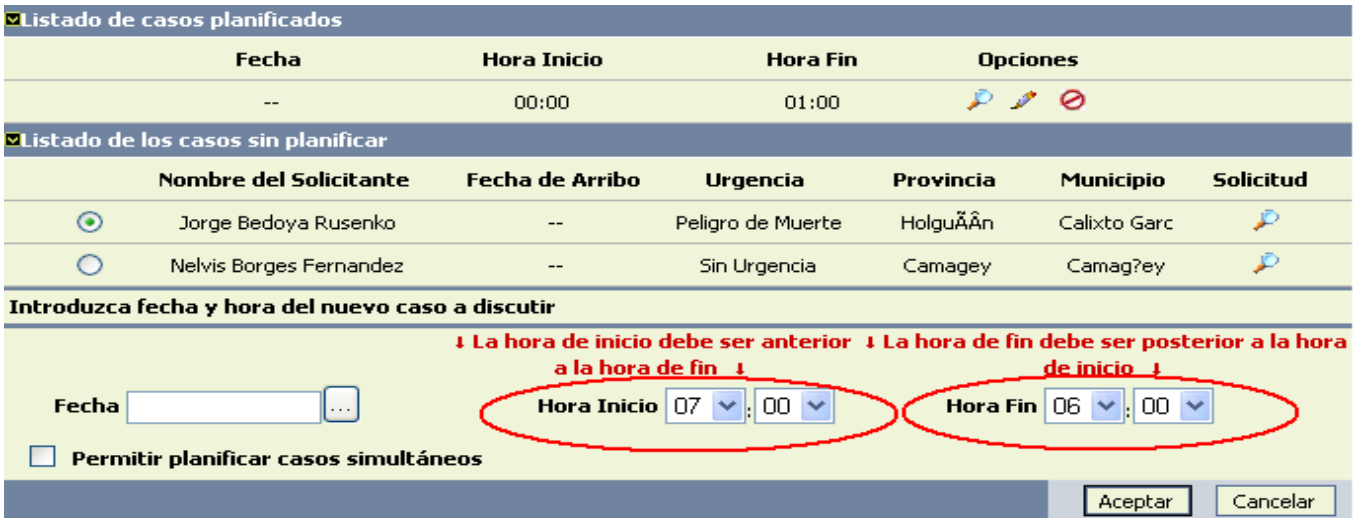
PHP es un lenguaje interpretado, esto significa que las instrucciones son interpretadas y ejecutadas en tiempo de ejecución, no media un lenguaje de objeto como en los lenguajes C++ y C#. Por tanto de más está decir que debuguear una aplicación web es tarea más que compleja. En ocasiones se hace necesario comprobar si determinada función se está ejecutando, o si cierto algoritmo está alcanzando un punto determinado, en estos casos se puede auxiliar el programador de la función ´echo´ de php que muestra un mensaje en el navegador tal como la consola en Java, C# o C++.

```
public function executeCodigoEtica(){
  if($this->getRequest()->getMethod() != sfRequest::POST)
    {
        echo "Si se está ejecutando el código de ética'";
        return sfView::SUCCESS;
    }
    else
    {
        $this->redirect('tl/buscarPac?newuser=true');
    }
}
```

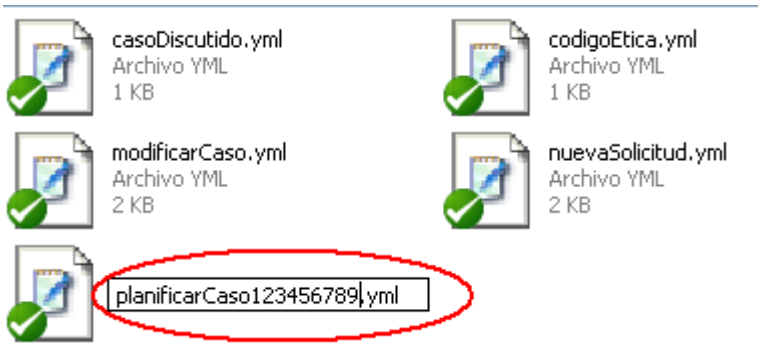
=>



La introducción inesperada de un 'echo' en una función puede producir efectos no deseados en la página como muestra la imagen en la parte derecha, pero la verificación de que la ejecución está alcanzando el punto que se deseaba ya está resuelta.
La introducción de datos incorrectos puede ser otra manera de verificar la consistencia de la aplicación que se está desarrollando, un adelanto y ayuda para la etapa de pruebas.



Desconfigurar un fichero, cambiarle el nombre, incluso eliminar un registro pueden constituir métodos, que aunque poco recomendados pueden aclarar y facilitar la solución de ciertos problemas.



Cambiar el nombre de un fichero .yml para desvincularlo de su 'template' puede esclarecer el origen de muchos problemas que se dan a la hora de las validaciones.

3.7 Conclusiones.

En este capítulo se definieron los elementos de implementación, representándose los diagramas de componentes de cada uno de los casos de usos, guiados por las definiciones realizadas durante el diseño. Para un mejor entendimiento del código de la aplicación se reflejaron los estándares de codificación. Se mostraron algunas de las interfaces y se expusieron los principales métodos así como algunos ejemplos de las validaciones realizadas a nivel de desarrollador.

Conclusiones Generales

El módulo Teleconsulta constituye un importante aporte al desarrollo de la medicina de este país, así lo demuestran las bondades que a continuación se reflejan:

Con la aplicación de patrones en el diseño donde las clases tenían sus responsabilidades bien definidas se logró realizar una codificación ágil y sencilla para la gestión de teleconsultas entre genetistas.

El producto funcional obtenido a partir de la implementación de las clases del diseño permitió integrar el módulo Teleconsulta al SIGM, dando solución al problema científico, proporcionando a los genetistas una herramienta para el intercambio de la información genética de los casos cuyos diagnósticos no estén claros, facilitando de esta forma la obtención de segundas opiniones de especialistas, tomando así decisiones multidisciplinarias en pacientes de difícil manejo. Posibilitando además la gestión de todo el proceso de solicitud, discusión e informe de los casos.

Recomendaciones

Teniendo en cuenta el estado que alcanzó el sistema implementado se recomienda:

- Realizar las pruebas exploratorias al sistema implementado.
- Extender el sistema desarrollado incluyéndole nuevas funcionalidades que permitan la realización de videoconferencias.

Referencias Bibliográficas

1. **Delgado Ramos, Ariel y Vidal Ledo, María.** Informática en la salud pública cubana. Rev Cubana Salud Pública. [En línea] 2006. [Citado el: 13 de Enero de 2008.] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm.

2. **Barroso Rodríguez, Yadira y Rodríguez Valenzuela, Annabell de la Caridad.** Sistema Informatizado para la Gestión de Teleconsultas entre Genetistas. 2007. Tesis de Diploma. UCI. La Habana.

3. [En línea] Junio 27, 2006. http://www.voxnet.it/home.cfm?ID=1047&ID2=t&espandi=1020.

4. Sala de Seminarios Clínicos-Patológicos. [En línea] 01 de Noviembre de 2000. http://www.conganat.org/IICONGRESO/seminarios/index.htm.

5. **Castillo T, Dra. Silvia y Pardo V., Dra. Andrea.** Proyecto Red de Teledismorfología de diagnóstico y consejo genético. [En línea] Septiembre de 2003. http://www.prematuros.cl/ramaneonatologia/reunionramaseptiembre.htm.

6. **Hispanoamericano, IV Congreso Virtual.** Telepatología en las comunidades virtuales de usuarios. [En línea] 04 de Noviembre de 2002. http://conganat.uninet.edu/IVCVHAP/CONFERENCIAS/Coma/.

7. Pontificia Universidad Católica de Chile. [En línea] Santiago - Chile. http://www.puc.cl/.

8. Universidad Virtual de Salud. [En línea] http://www.uvirtual.sld.cu/.

9. **REYNOX.** Metodología de Desarrollo de Software (MDS). [En línea] 2005. [Citado el: 20 de Marzo de 2007.] http://www.reynox.com.ar/sap/metodologia.php.

10. **Zamitiz, Roman y Alberto, Ing.Carlos.** [En línea] 2006. [Citado el: 10 de Diciembre de 2007.] http://www.fi-b.unam.mx/pp/profesores/carlos/aydoo/uml.html.).

11. **Conallen, Jim.** Building Web Applications with UML. [En línea] 2002.

12. **Hernández, Jose Alberto.** [En línea] 2005. [Citado el: 14 de Diciembre de 2007.] Versión Cero. http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.

13. Página oficial de MySQL. [En línea]. http://www.mysql.com/. [Citado el: 5 de Febrero de 2008.] http://www.mysql.com/common/pages/download_access_denied.html.

14. Apache http server project. [En línea]. http://httpd.apache.org. [Citado el: 9 de Febrero de 2008.] http://httpd.apache.org/docs/2.0/es/new_features_2_0.html.

15. **Sanz Bermejo, Laura y Monreal Gómez, Enrique.** Eclipse como IDE. [En línea] [Citado el: 12 de Febrero de 2008.] http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf.

16. **Zaninotto, Fabien Potencier y Francois.** [En línea] 20 de Diciembre de 2007. [Citado el: 12 de Enero de 2008.] http://www.librosweb.es/symfony/.

17. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. [En línea] La Habana, Cuba.

18. **Cabezas Granado, Luis Miguel.** PHP 5. [En línea] 2004. España. ISBN: 8441517851.

19. Patrones de diseño. [En línea] Colectivo de. 2007-2008. UCI.

20. **Soriano, Ing. Fernando.** Diseño Orientado a Objetos Patrones GoF. [En línea] 2008.

21. Conferencia 7 IS1 Patrones de diseño. [En línea] 2005-2006. Universidad de las Ciencias Informáticas. La Habana, Cuba.

22. **Oktaba, Hanna.** Introducción a Patrones. [En línea] Facultad de Ciencias, UNAM. http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

23. Introducción a la ingeniería de software. [En línea] 2007-2008. Colectivo de la UCI.

24. FT Análisis y Diseño. Modelo de Diseño. [En línea] 2007-2008. Cuba, Ciudad Habana.

25. Conferencia 3 IS1 Fase de Inicio. Flujo de trabajo de requerimientos. [En línea] 2005-2006. Universidad de las Ciencias Informáticas. La Habana, Cuba.

26. **Orosco, Eyleen, Molina, Elvismary y Sánchez, Yusdenis.** Arquitectura de Información. [En línea] http://10.7.9.200:5901/svn/genetica/Ingenieria/Arquitectura/.

27. **Fabien Potencier, Francois Zaninotto.** Symfony la guía definitiva. [En línea] 2008. ISBN-13.

28. **Marrero, Carlos David, Muro, Fernando y Rivero, Henry.** Merinde. [En línea] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=96&Itemid=297.

29. **Orosco, Eyleen, Molina, Elvismary y Sánchez, Yusdenis.** Documento de Arquitectura de Software. [En línea] http://10.7.9.200:5901/svn/genetica/Ingenieria/Arquitectura/.

Bibliografía

1. **Delgado Ramos, Ariel y Vidal Ledo, María.** Informática en la salud pública cubana. Rev Cubana Salud Pública. [En línea] 2006. [Citado el: 13 de Enero de 2008.] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm.

2. **Barroso Rodríguez, Yadira y Rodríguez Valenzuela, Annabell de la Caridad.** Sistema Informatizado para la Gestión de Teleconsultas entre Genetistas. 2007. Tesis de Diploma. UCI. La Habana.

3. **REYNOX.** Metodología de Desarrollo de Software (MDS). [En línea] 2005. [Citado el: 20 de Marzo de 2007.] http://www.reynox.com.ar/sap/metodologia.php.

4. **Conallen, Jim.** Building Web Applications with UML. [En línea] 2002.

5. **Zaninotto, Fabien Potencier y Francois.** [En línea] 20 de Diciembre de 2007. [Citado el: 12 de Enero de 2008.] http://www.librosweb.es/symfony/.

6. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. [En línea] La Habana, Cuba.

7. Patrones de diseño. [En línea] Colectivo de. 2007-2008. UCI.

8. **Soriano, Ing. Fernando.** Diseño Orientado a Objetos Patrones GoF. [En línea] 2008.

9. **Oktaba, Hanna.** Introducción a Patrones. [En línea] Facultad de Ciencias, UNAM. http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

10. Introducción a la ingeniería de software. [En línea] 2007-2008. Colectivo de la UCI.

11. **Orosco, Eyleen, Molina, Elvismary y Sánchez, Yusdenis.** Arquitectura de Información. [En línea] http://10.7.9.200:5901/svn/genetica/Ingenieria/Arquitectura/.

12. **Fabien Potencier, Francois Zaninotto.** Symfony la guía definitiva. [En línea] 2008. ISBN-13.

13. **Larman, Craig.** UML y Patrones. [En línea] 2004. La Habana, Cuba.

14. **Marrero, Carlos David, Muro, Fernando y Rivero, Henry.** Merinde. [En línea] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=96&Itemid=297.

15. [En línea] Junio 27, 2006. http://www.voxnet.it/home.cfm?ID=1047&ID2=t&espandi=1020.

16. **RUMBAUGH, JAMES, BOOCH, GRADY and JACOBSON, IVAR.** El lenguaje unificado de modelado. [En línea] manual de referencia. s.l. : Pearson Prentice Hall, 2003. ISBN: 8478290370.

17. **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.

18. **Presman.** Uml y Patrones. [Online] s.l. : Pearson (2003, 2ª edición), 2003.

19. **Sánchez Camejo, Dr. Alejandro and Lorizio, Dra. Wendy.** INFORMATION & COMMUNICATION TECHNOLOGIES IN HEALTHCARE DEVELOPMENT 3rd VIRTUAL CONGRESS IN INTERNET. [En línea] March 1, 2004. http://www.informaticamedica.org/I04/papers/sanchez_5.pdf.

20. **Orosco, Eyleen, Molina, Elvismary y Sánchez, Yusdenis.** Documento de Arquitectura de Software. [En línea] http://10.7.9.200:5901/svn/genetica/Ingenieria/Arquitectura/.

Anexos

Anexo # 1: Aval del Centro Nacional de Genética Médica sobre el módulo Teleconsulta.

Ciudad de la Habana, 10 de junio de 2008
"Año 50 de la Revolución"

A quien corresponda:

Por este medio dejo constancia de los beneficios que permite el módulo Teleconsulta genética, que se ha desarrollado por la Universidad de las Ciencias Informáticas para la Red de Genética Médica; estos incluyen:

1.-Posibilidad de apoyar el diagnóstico en casos donde es difícil identificar la afección que presenten, sin necesidad de que los pacientes salgan de su comunidad de residencia. Esto es de gran importancia ya que generalmente las personas que sufren enfermedades de causa genética presentan una, o varias, discapacidades.

2.-Obtener una segunda opinión, muchas veces de expertos. Las afecciones genéticas son poco frecuentes y raramente un mismo médico logra vivencias personales amplias.

3.-Tomar decisiones multi y trans disciplinarias en pacientes de difícil manejo, con la ventaja de que pueden consultarse simultáneamente especialistas situados en distintos puntos del país a través de una discusión de casos virtual.

Con todo lo anterior se logra que las nuevas tecnologías de la información sirvan de apoyo a la Genética Clínica y esté a servicio de quienes lo necesitan -nuestros pacientes- como parte de una práctica médica justa.

Saludos cordiales,

Estela Morales Peralta
Especialista en Genética Clínica
Centro Nacional de Genética Médica

Anexo # 2: Breve descripción de los casos de usos del sistema.

Caso de Uso:	Solicitar discusión de un caso.
Actores:	Solicitante (inicia)
Resumen:	El caso de uso se inicia cuando el Solicitante desea solicitar la discusión de un caso, para esto primeramente debe aceptar el código de ética, luego debe buscar por diferentes criterios al paciente, seleccionando al correspondiente para mostrar sus datos y comenzar a introducir la información de la solicitud, puede proponer participantes para la discusión, finalizando el caso de uso con el almacenamiento de los datos en el sistema.
Referencia:	R1, R2, R3, R4
CU asociados:	Proponer participantes a la discusión (extendido). Gestionar datos primarios (incluido).
Precondiciones:	El Solicitante ha realizado correctamente el registro en el sistema introduciendo el nombre de usuario y la contraseña.

Tabla 1: Breve descripción del CU: "Solicitar discusión de un caso".

Caso de Uso:	Proponer participantes a la discusión (extendido).
Actores:	Solicitante.
Resumen:	El caso de uso se ejecuta cuando el Solicitante desea proponer participantes para la discusión, o eliminar alguno de los que están adicionados. Si escoge la opción de proponerlo, debe buscar los que desea adicionar teniendo en cuenta diferentes criterios de búsqueda; si escoge eliminarlo, debe especificar cual desea eliminar y este no podrá participar en la discusión del caso. El caso de uso finaliza cuando el Solicitante especifica al sistema que no desea realizar otra acción sobre los participantes.
Referencia:	R11, R12, R13, R14
CU asociados:	
Precondiciones:	

Tabla 2: Breve descripción del CU: "Proponer participantes a la discusión".

Caso de Uso:	Autorizar solicitud.
Actores:	Administrador (inicia)

Resumen:	El caso de uso se inicia cuando el Administrador desea autorizar la solicitud, para lo cual deberá seleccionar del listado de solicitudes, ordenadas por el nivel de urgencia y la fecha de arribo, en espera de aprobación la(s) que desea(n) aprobar o denegar. Si el administrador decide denegar la solicitud se le notifica al solicitante que su solicitud ha sido denegada con una explicación. Finaliza el caso de uso cuando el Administrador notifica al sistema que ya no desea seguir aprobando o denegando solicitudes.
Referencia:	R5, R6, R7, R24
CU asociados:	Notificar negación de solicitud (extendido).
Precondiciones:	El Administrador ha realizado correctamente el registro en el sistema introduciendo el nombre de usuario y la contraseña. Deben existir solicitudes registradas sin autorizar.

Tabla 3: Breve descripción del CU: "Autorizar solicitud".

Caso de Uso:	Notificar negación de solicitud (extendido).
Actores:	Administrador y Servidor de correo.
Resumen:	El caso de uso se inicia cuando se deniega la solicitud de discusión de un caso pues se necesita enviar un correo electrónico al Solicitante, el sistema le muestra al Administrador una página para que llene con la explicación de la negación y a través del Servidor de correo se envía, finalizando así el caso de uso.
Referencia:	R8
CU asociados:	
Precondiciones:	Debe denegarse una solicitud.

Tabla 4: Breve descripción del CU: "Notificar negación de solicitud".

Caso de Uso:	Gestionar caso a discutir.
Actores:	Administrador (inicia).
Resumen:	El caso de uso se inicia cuando el Administrador necesita planificar un caso a discutir, consultar, modificar o eliminar alguno de los existentes. Para esto se muestran los casos a discutir planificados así como los casos a discutir que no están planificados. Si escoge la opción de adicionar, el caso a discutir queda añadido al listado de los casos a discutir planificados, si escoge modificar

	puede cambiar fecha, hora de inicio y fin de la discusión del caso y sus participantes, si escoge eliminar queda eliminado de la lista de casos a discutir planificados. Si escoge mostrar puede ver los datos del caso planificado que desee. Finaliza el caso de uso enviando un correo a los participantes después de aceptar cualquiera de las opciones de planificar, modificar o eliminar.
Referencia:	R9, R10, R15, R16, R17, R24.
CU asociados:	Citar participante a la discusión (incluido).
Precondiciones:	El Administrador ha realizado correctamente el registro en el sistema introduciendo el nombre de usuario y la contraseña. Deben existir casos sin planificar y planificados.

Tabla 5: Breve descripción del CU: "Gestionar caso a discutir".

Caso de Uso:	Gestionar participante a la discusión.
Actores:	Administrador.
Resumen:	El caso de uso se ejecuta cuando el Administrador necesita adicionar un nuevo participante a la discusión de un caso, o eliminar alguno de la misma. Si escoge la opción de adicionarlo, deberá buscar los participantes que desea adicionar teniendo en cuenta diferentes criterios de búsqueda; si escoge eliminarlo, deberá especificar cual desea eliminar y este no podrá participar en la discusión del caso. El caso de uso finaliza cuando el Administrador especifica al sistema que no desea realizar otra acción sobre los participantes.
Referencia:	R11, R12, R13, R14.
CU asociados:	
Precondiciones:	Deben existir casos planificados.

Tabla 6: Breve descripción del CU: "Gestionar participante a la discusión".

Caso de Uso:	Citar participante a la discusión (incluido).
Actores:	Administrador, Servidor de correo
Resumen:	El caso de uso se inicia cuando el Administrador acepta la planificación del caso, su modificación o eliminación y el sistema envía automáticamente un mensaje citando a los participantes propuestos en la misma, el mensaje contiene la fecha y hora de inicio y fin además de los datos del caso a discutir. En el momento que se elimine un caso a discutir planificado también se les

	notifica a los participantes propuestos, finalizando el caso de uso.
Referencia:	R18.
CU asociados:	
Precondiciones:	Debe aceptarse la planificación de discusión de un caso, su modificación o eliminación.

Tabla 7: Breve descripción del CU: "Citar participante a la discusión".

Caso de Uso:	Administrar caso en discusión.
Actores:	Moderador (inicia).
Resumen:	El caso de uso se inicia cuando el Moderador selecciona el primer caso a discutir planificado para el día de hoy ya que los mismos están ordenados por su hora de inicio. Comienza la discusión cuando el Moderador lo determine. Luego del debate entre los participantes se desea dar por concluida la discusión del caso, pudiendo posponerla si se necesita de otro momento para continuar la discusión, remitirla al nivel superior si no encuentran el diagnóstico definitivo, o darla por concluida si se llega a la solución satisfactoria de la misma. Cuando se termine de debatir el caso se guarda el historial y el informe del caso, estando este último disponible para ser aprobado antes de mostrarse a todos los participantes del debate, finalizando así el caso de uso.
Referencia:	R19, R20, R21, R22, R23, R24, R25, R26.
CU asociados:	
Precondiciones:	El Moderador ha realizado correctamente el registro en el sistema introduciendo el nombre de usuario y la contraseña. Debe existir al menos un caso a discutir planificado para la fecha de la computadora.

Tabla 8: Breve descripción del CU: "Administrar caso en discusión".

Caso de Uso:	Participar en discusión de un caso.
Actores:	Participante (inicia).
Resumen:	El caso de uso se inicia cuando el participante registrado desea entrar a la discusión de un caso, si la misma está habilitada se le muestra el caso que se está discutiendo en ese momento, un enlace a los datos de este y la sala de discusión, en caso contrario se especifica que aún no se ha iniciado la discusión. Cuando se realiza el debate, una vez terminado el mismo al

	Solicitante se le muestra una página para que llene el resumen, recomendaciones y el diagnóstico final o no del caso discutido seleccionando la enfermedad genética de un listado. Finaliza el caso de uso guardándose la información en el sistema
Referencia:	R19, R20, R24, R25, R27.
CU asociados:	
Precondiciones:	El participante ha realizado correctamente el registro en el sistema introduciendo el nombre de usuario y la contraseña. Debe existir al menos un caso a discutir para esa fecha. El participante debe estar entre los propuestos.

Tabla 93: Breve descripción del CU: "Participar en discusión de un caso".

Caso de Uso:	Visualizar informe.
Actores:	Participante (inicia).
Resumen:	El caso de uso se inicia cuando el participante desea ver el informe de un caso discutido, dándole la posibilidad de imprimirlo.
Referencia:	R28.
CU asociados:	
Precondiciones:	El participante ha realizado correctamente el registro en el sistema introduciendo el nombre de usuario y la contraseña. El informe del caso discutido debe estar aprobado.

Tabla 40: Breve descripción del CU: "Visualizar informe".

Anexo # 3: Diagramas de secuencia.

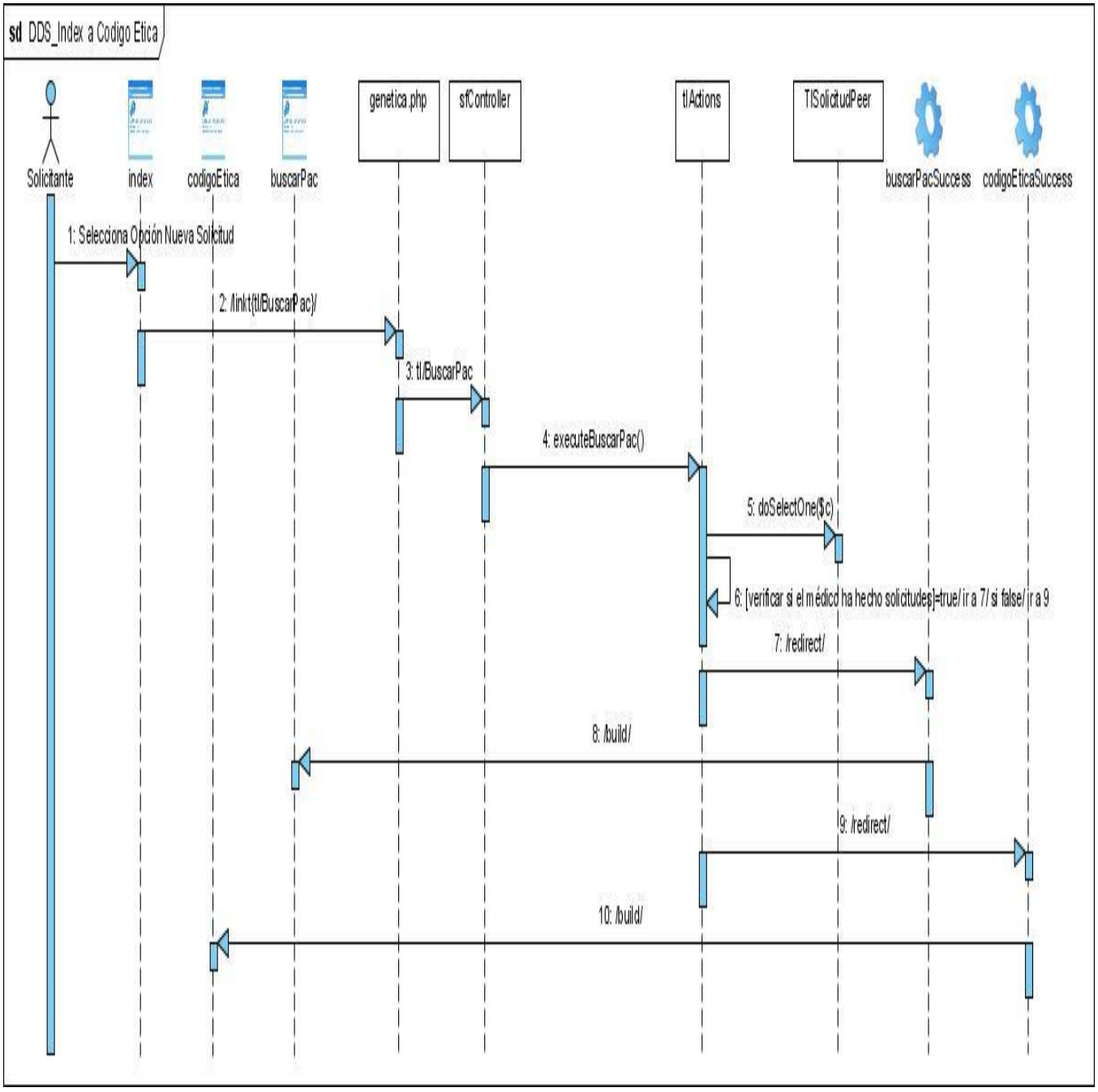


Diagrama de secuencia "Index a Aceptar Código".

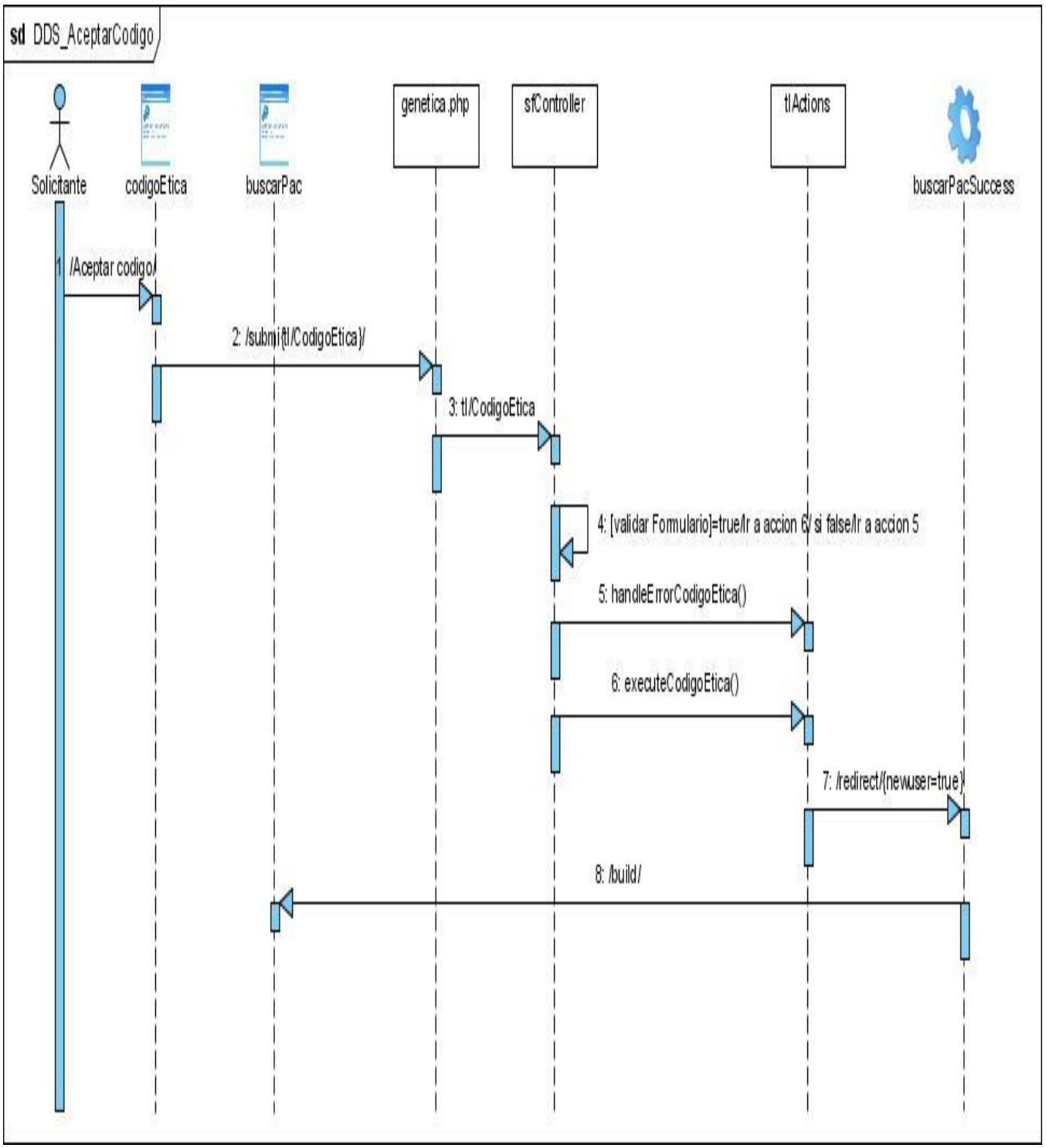


Diagrama de secuencia "Aceptar Código".

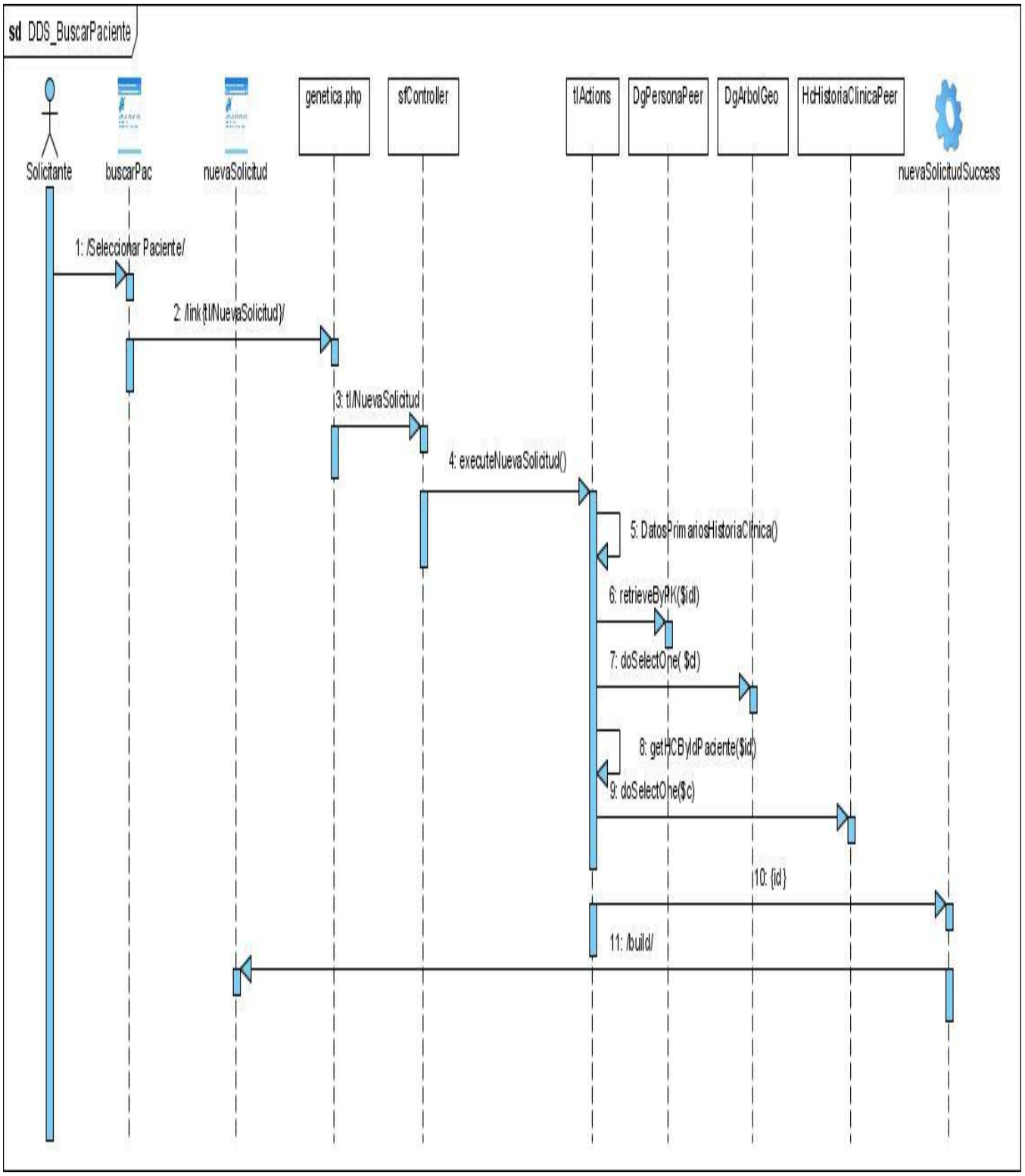


Diagrama de secuencia "Buscar Paciente".

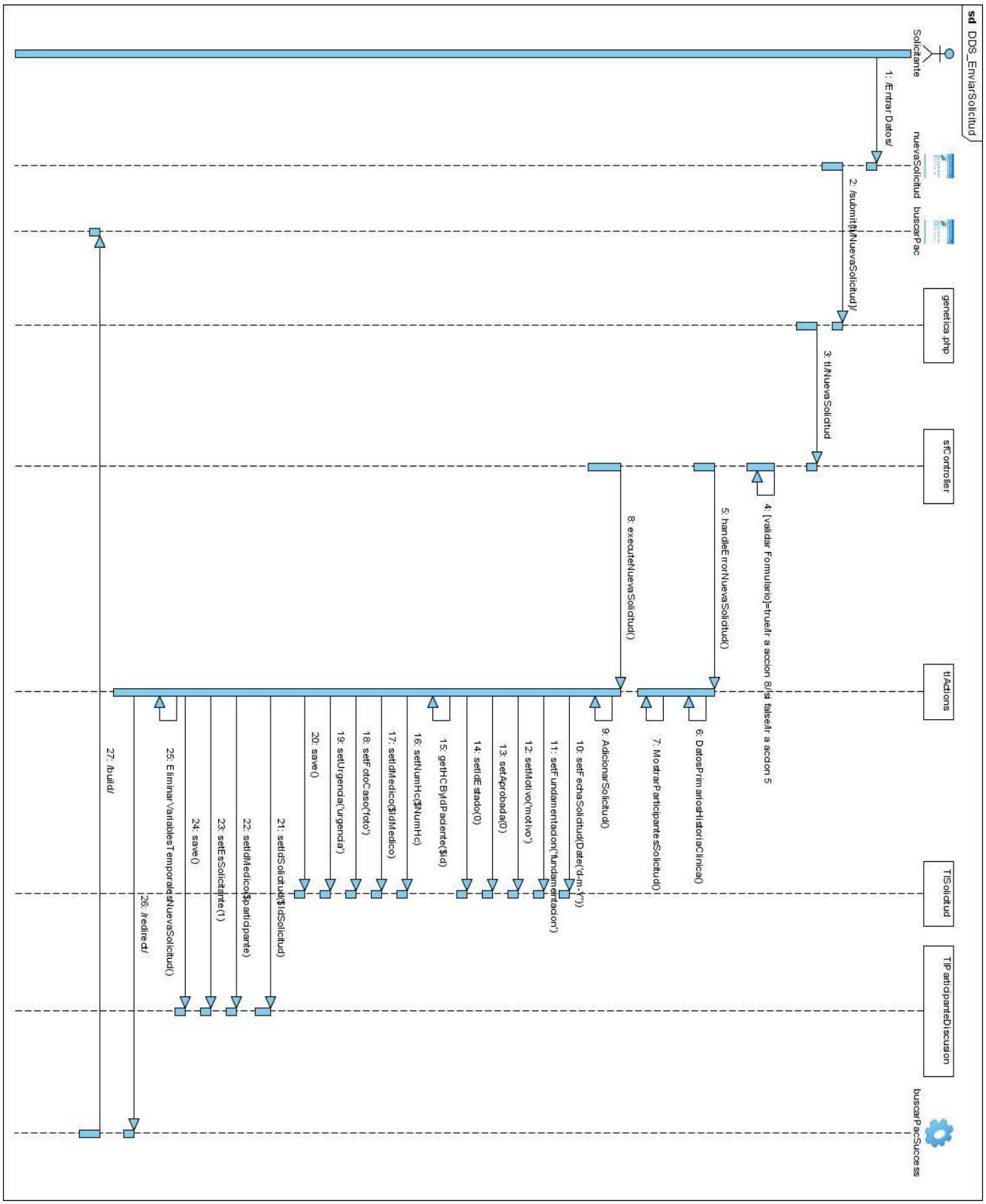


Diagrama de secuencia “Enviar Solicitud”.

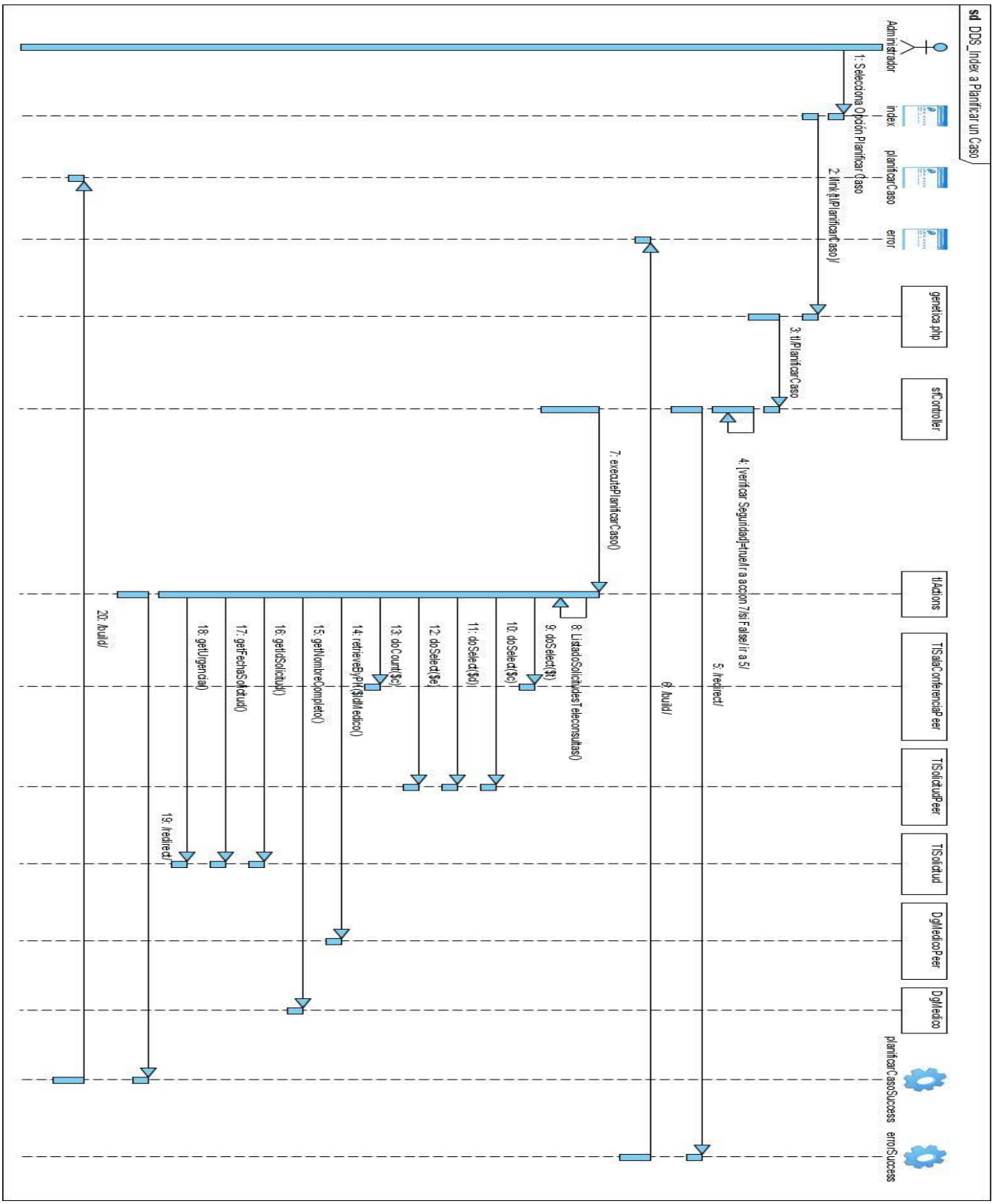


Diagrama de secuencia "Index a Planificar Caso".

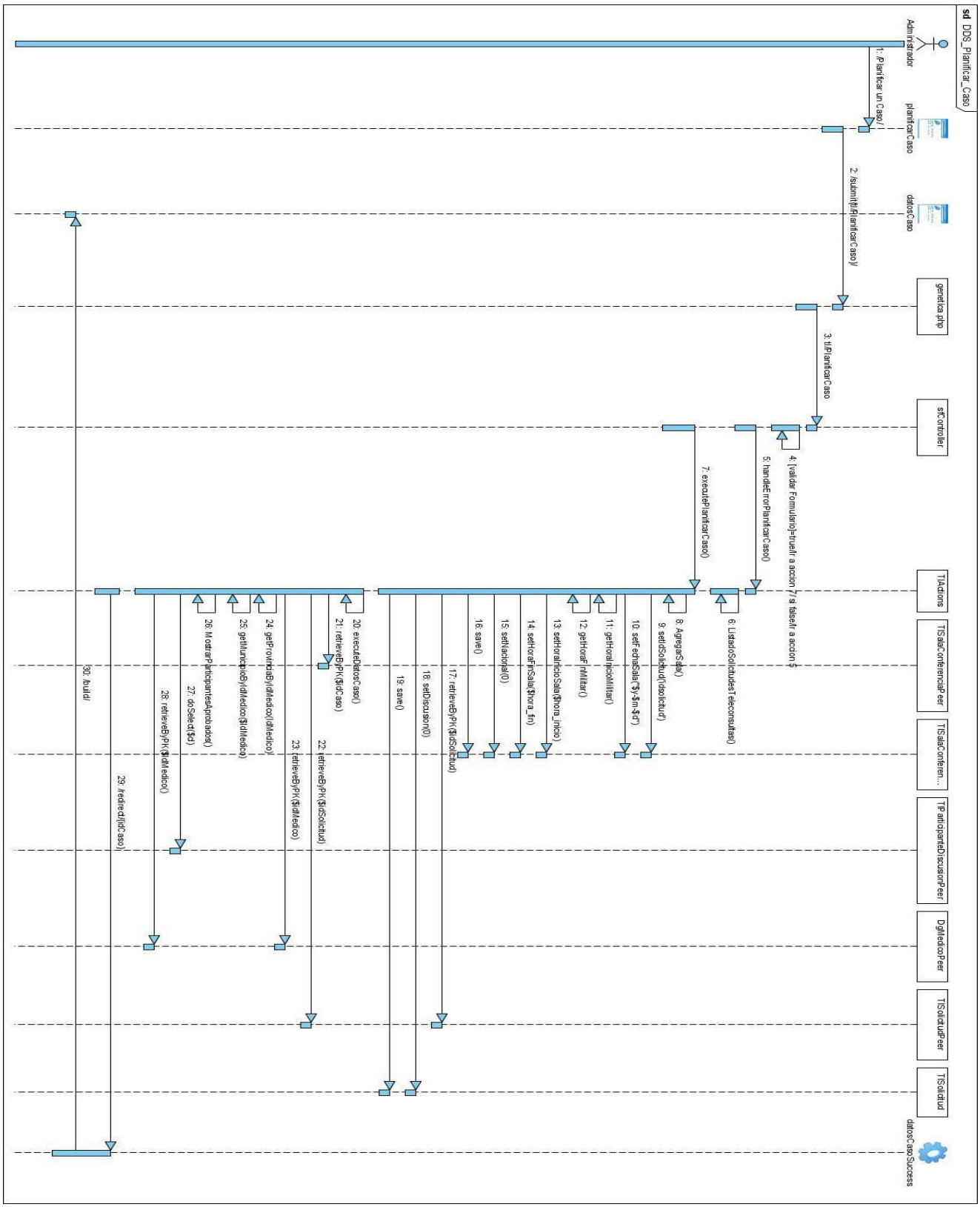


Diagrama de secuencia “Planificar Caso”.

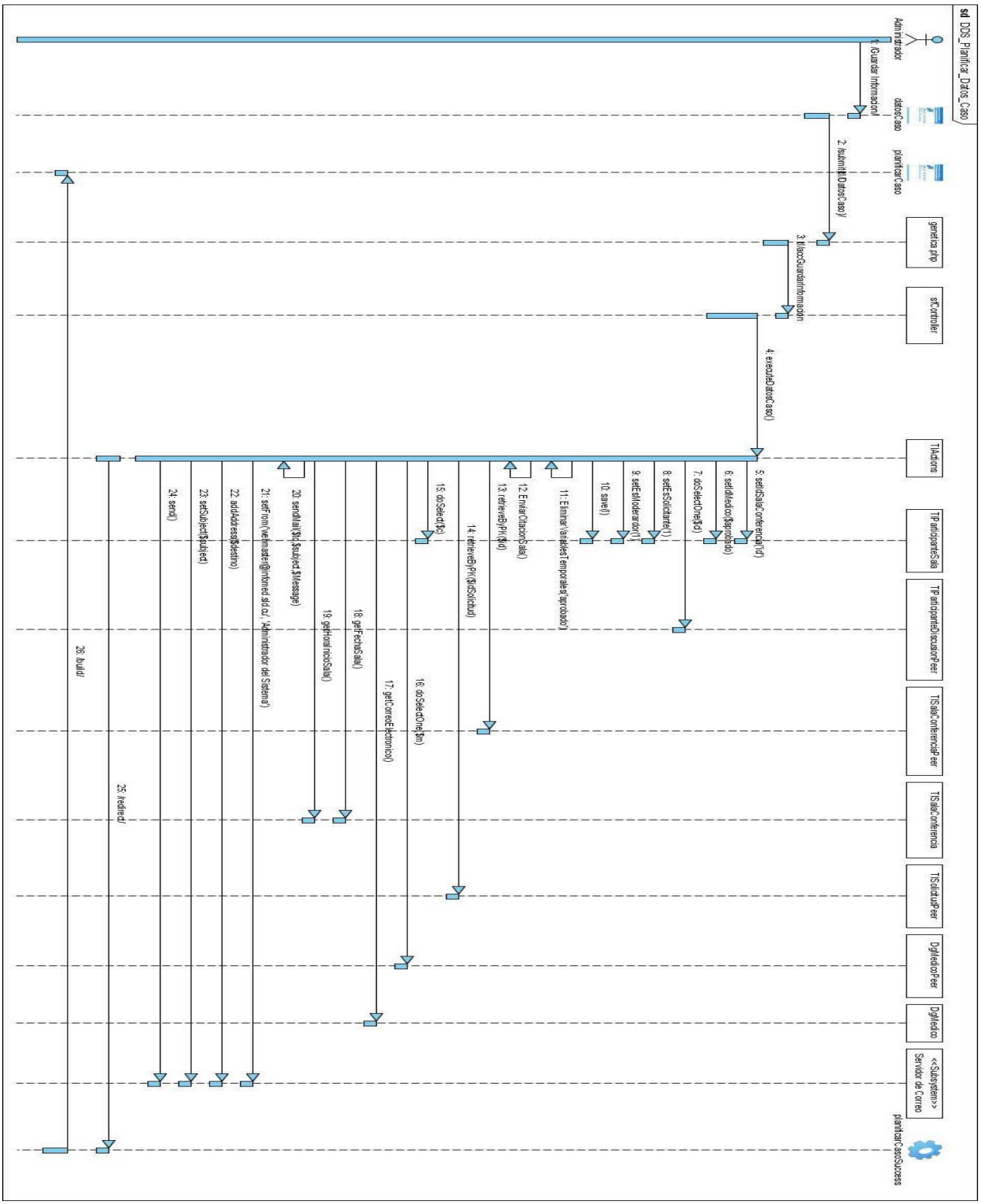


Diagrama de secuencia “Planificar Datos Caso”.

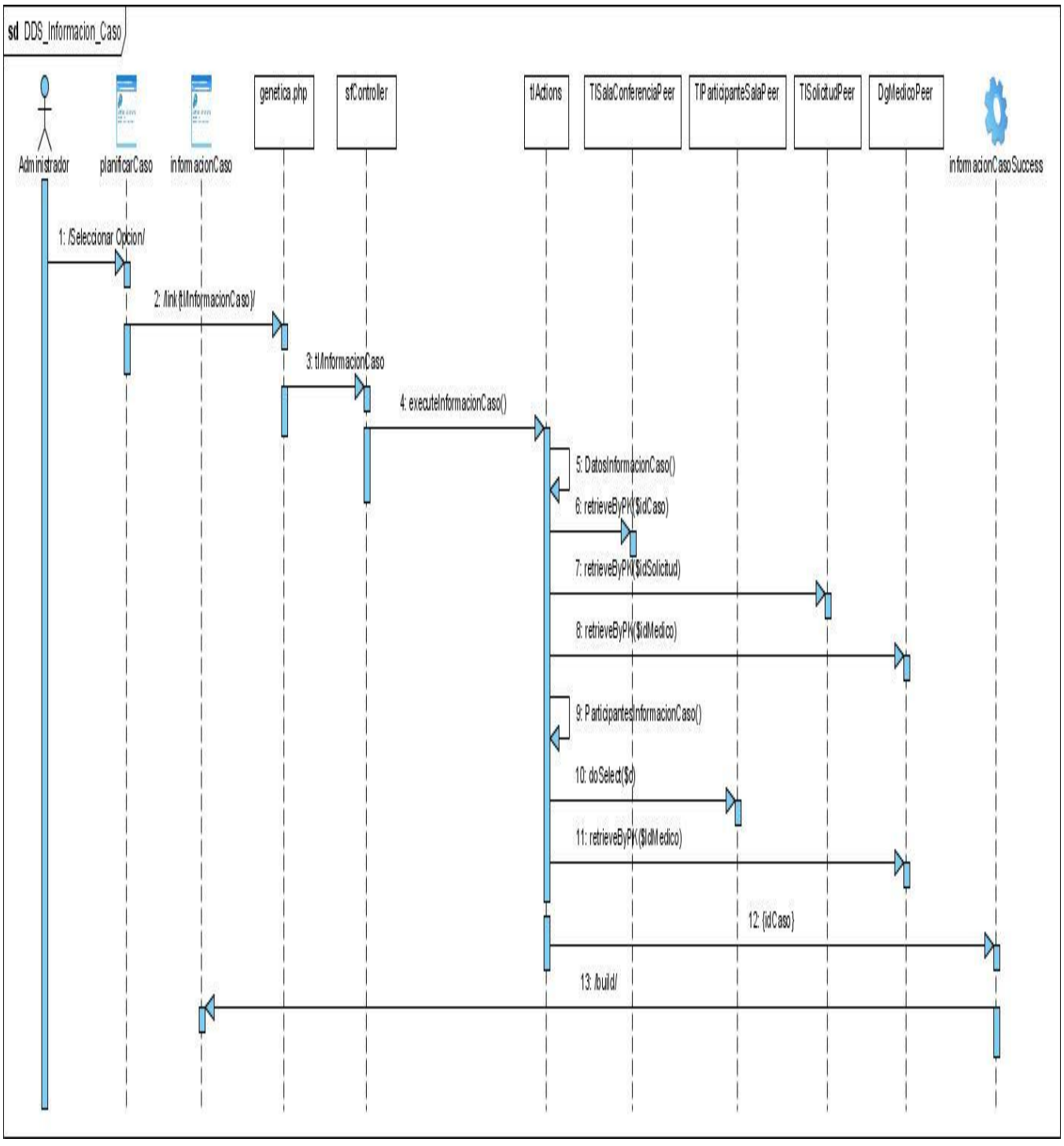


Diagrama de secuencia "Información Caso".

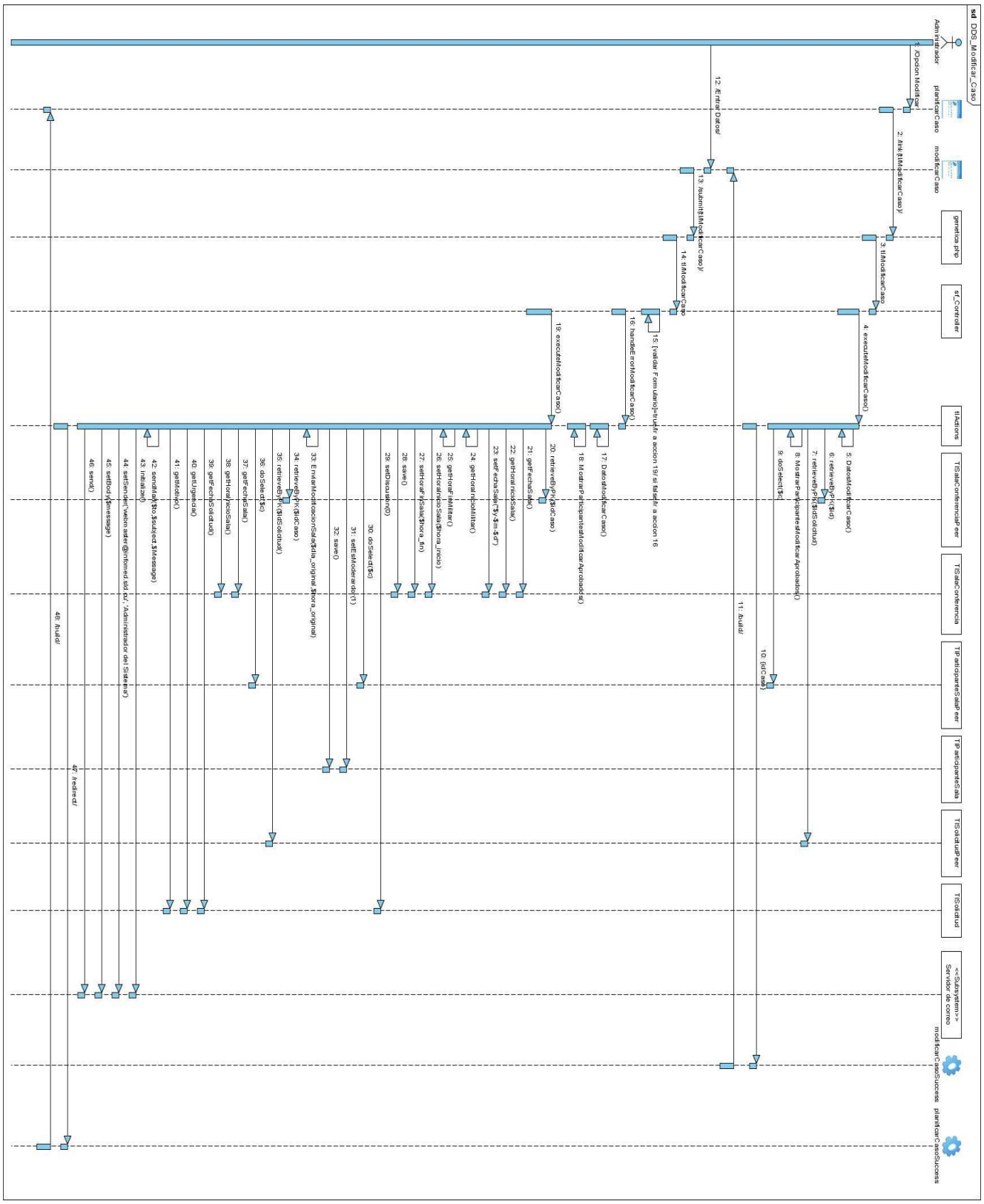


Diagrama de secuencia “Modificar Caso”.

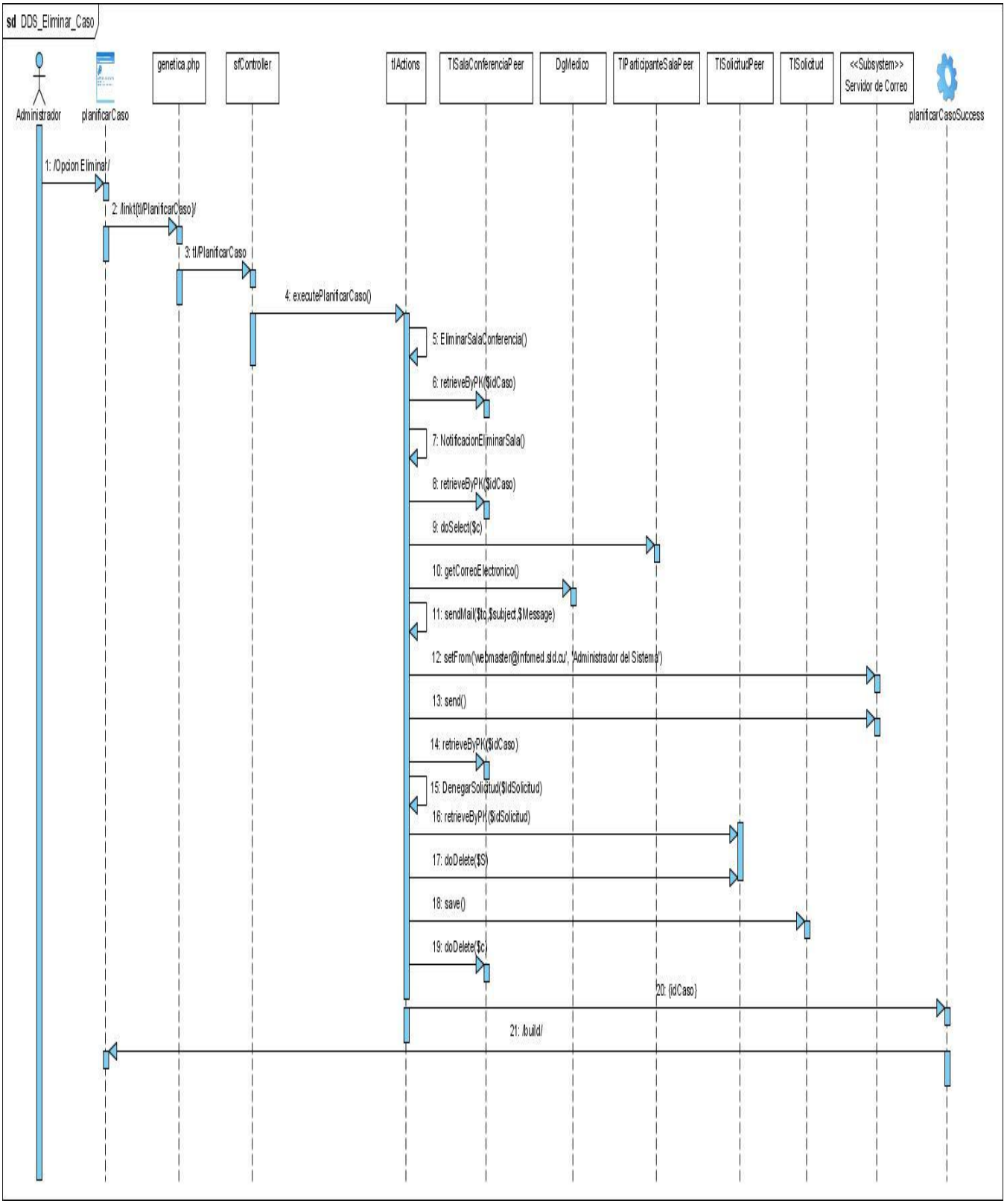


Diagrama de secuencia "Eliminar Caso".

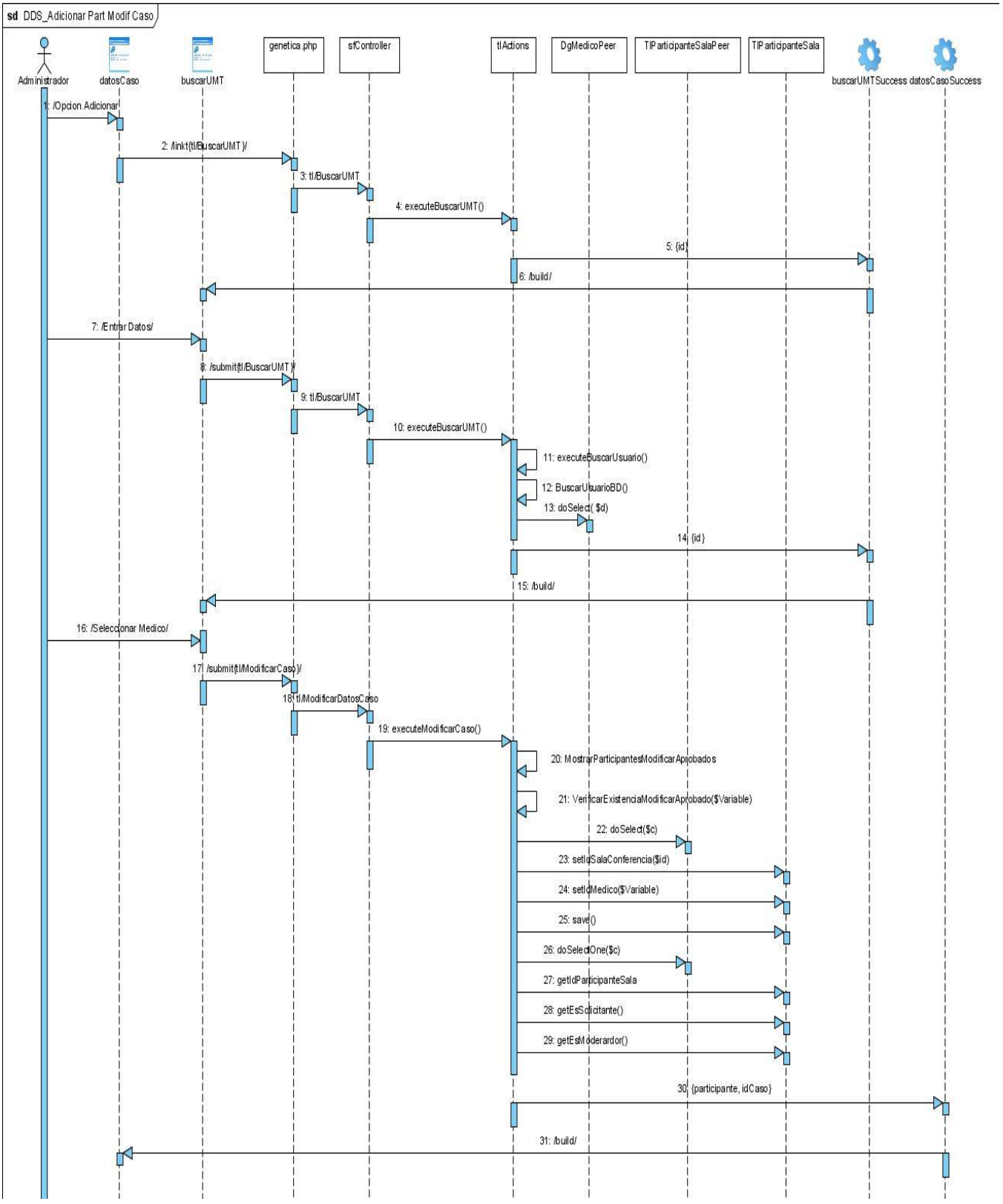


Diagrama de secuencia "Adicionar Participante Modificar Caso".

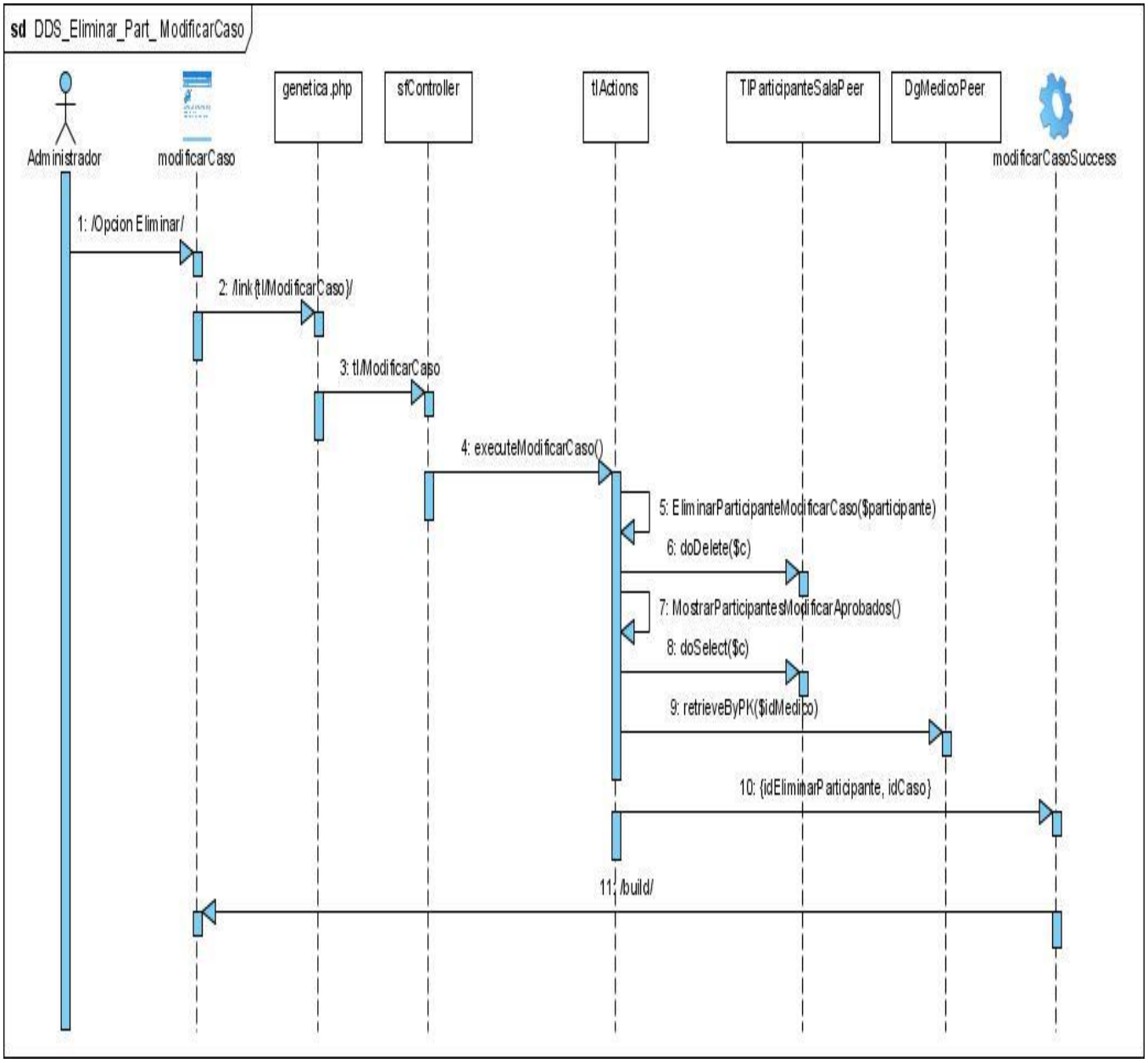


Diagrama de secuencia "Eliminar Participante Modificar Caso".

Nota: Todas las descripciones detalladas de los casos de uso del sistema y los diagramas de secuencia correspondiente a los casos de uso "Proponer participantes a la discusión", "Autorizar solicitud", "Notificar negación de solicitud", "Citar participante a la discusión", "Participar en discusión de un caso" y "Visualizar informe", se encuentran en el expediente del proyecto.

Glosario de Términos

Base de datos: Conjunto de datos organizados entre los cuales existe una correlación y que están almacenados con criterios independientes de los programas que los utilizan. La filosofía de las bases de datos es la de almacenar grandes cantidades de datos de una manera no redundante y que permita las posibles consultas de acuerdo a los derechos de acceso.

Caso a discutir: Solicitud que ya ha sido aprobada por el administrador de la Teleconsulta.

Caso de uso: Fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

CNGM: Centro Nacional de Genética Médica

Enfermedad rara: Enfermedades Raras, incluidas las de origen genético, son aquellas enfermedades con peligro de muerte o de invalidez crónica, que tienen una frecuencia (prevalencia) baja, menor de 5 casos por cada 10.000 habitantes en la comunidad.

Genética: Rama de las ciencias biológicas que trata de comprender cómo la herencia biológica es transmitida de una generación a la siguiente, y cómo se efectúa el desarrollo de las características que controlan estos procesos.

GPL: General Public License (inglés: Licencia Pública General) es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software Libre.

Herramienta CASE: Aplicación informática destinada a aumentar la productividad en el desarrollo de software.

HTML: Es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

Hosteado: Sistema informático ejecutado a través de un servidor o host.

Interfaz: Es la parte de un programa informático que permite a este comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

Malformativo: Alteración o deformidad de nacimiento en alguna parte del organismo.

Plugin: Es un programa de ordenador que interactúa con otro programa para aportarle una función o utilidad, generalmente muy específica.

Protocolo: Conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red.

Servidor Web: Programa que implementa el protocolo HTTP (Hypertext Transfer Protocol). Este protocolo está diseñado para transferir los llamados hipertextos, páginas web o páginas HTML (Hypertext Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

SGBD: Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.

SIGM: Sistema Informático para la Red Nacional de Genética Médica

Teleconsulta: Es la interacción compartida de imágenes e información médica en la que el diagnóstico primario es realizado por el doctor en la locación del paciente. El propósito de la teleconsulta es proveer una segunda opinión por un especialista remoto para confirmar el diagnóstico o para ayudar al médico local a llegar a un diagnóstico correcto.

Versión: Término que nombra las actualizaciones de un producto, se utiliza cuando se saca al mercado, o bien, cuando las modificaciones que se hacen del antiguo son muy numerosas o de gran alcance.