Universidad de las Ciencias Informáticas
Facultad 6

Título: Sistema de Manejo de Datos de Ensayos Clínicos Cubano para el Centro de Inmunología Molecular. Diseño e implementación del "Cronograma de Ejecución" del Módulo de Diseño.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores:

Liyanis Dube Argote

Yasiel Fonseca Fernández

Tutores:

Ing. Andrés Ballester Marsal

Ing. Aidacelys López Díaz

Habana, Junio 2008

Para el logro del triunfo siempre ha sido indispensable pasar por la senda de los sacrificios.

Simón Bolívar

DECLARACIÓN DE AUTORÍA

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

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

Liyanis Dube Argote

_____

Firma del Autor

Yasiel Fonseca Fernández

_____

Firma del Autor

Aidacelys López Díaz

_____

Firma del Tutor

Andrés Ballester Marsal

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Andrés Ballester Marsal

El compañero Andrés Ballester Marsal, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano para el Centro Inmunología Molecular. Diseño e implementación del submódulo Cronograma de Ejecución del módulo de Disño, fue graduado de la Universidad de las Ciencias Informáticas con el título de Ingeniero en Ciencias Informáticas en el año 2007, obteniendo como calificativo la bonificación de 5 pts. en la exposición de su trabajo de diploma, actualmente se desempeña como profesora de la facultad 6 impartiendo las asignaturas Gráfico por Computadoras y Programación Web, participa en las tutorías, oponencias y tribunales de los trabajos de diplomas, además es el Arquitecto del proyecto Ensayos Clínico.

Ing. Aidacelys López Días

La compañera Aidacelys López Díaz, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano para el Centro Inmunología Molecular. Diseño e implementación del submódulo Cronograma de Ejecución del módulo de Disño, fue graduada de la Universidad de las Ciencias Informáticas con el título de Ingeniera en Ciencias Informáticas en el año 2007, obteniendo como calificativo la bonificación de 5 pts. en la exposición de su trabajo de diploma, actualmente se desempeña como profesora de la facultad 6, participa en las tutorías, oponencias y tribunales de los trabajos de diplomas, además pasee el rol de Líder del proyecto Ensayos Clínicos perteneciente a la facultad 6.

AGRADECIMIENTOS

A Aidacelys por su apoyo y dedicación

A Ballester por ser incansable, por ser tan bella persona, por creer que si podíamos.

A Luis Castro Rico ese niño que creyó siempre en mí, me apoyo en los buenos y malos momentos. A ti mi vida, un beso y un te amo.

A Elena María Rico y Luis Adolfo Castro por acogerme como un miembro más de su familia.

A mis amigas Yuriskenia y Yamirita se los debo, gracias por su amistad

A los viejos y nuevos amigos que de todos me llevo lo mejor, para guardarlo siempre conmigo.

A todos los que me apoyaron, que me preguntaron ¡que tal la tesis! Gracias.

Liyanis Dube Argote

A la tutora Aidacelys por su paciencia, por sus continuas y acertadas recomendaciones.

Al tutor Andrés Ballester por estar siempre dispuesto a ayudarnos, por las noches enteras que dedicó a la revisión de este trabajo, por sus continuos consejos gracias a los cuales ha sido posible llevar adelante la investigación, por ser un ejemplo de nobleza y conocimientos, dos cualidades admirables y poco comunes en una persona: a usted muchas gracias.

A mis amistades con las que he compartido estos cinco años que aunque a muchos no los vuelva a ver han dejado huella en mi carácter y mi personalidad, me han ayudado a encontrar el camino a seguir en la vida.

A todas esas personas que me aprecian y estiman que muchas veces, incluso sin darse cuenta, me han dado con su ejemplo lecciones que me han ayudado a conseguir todo lo que soy.

A mi padre con el cual he aprendido muchas cosas importantes de la vida.

Yasiel Fonseca Fernández.

DEDICATORIA

A mi madre por ser mi guía y mi luz en este camino tan difícil que es la vida, por ser ella esa personita que siempre estará ahí para mi, por ser simplemente la persona que más amo.

A mi padre por creer que podía lograrlo. Te quiero.

Liyanis Dube Argote

A mi madre a quien debo todo lo que soy o seré en la vida, quien es la fuerza que siempre me inspira a seguir adelante.

Yasiel Fonseca Fernández

RESUMEN

La investigación surge en el marco de trabajo del proyecto: Ensayos Clínicos, en colaboración con el Centro Inmunología Molecular (CIM) y la Universidad de las Ciencias Informáticas (UCI). El CIM tiene como objetivo principal la creación de biofármacos, para el tratamiento de enfermedades crónicas no trasmisibles; para comprobar la eficacia y seguridad de estos productos, se realizan Ensayos Clínicos (EC), los mismos se efectúan con pacientes enfermos que se ofrecen como voluntarios para someterse a tratamientos bajo investigación. Cada estudio clínico presenta un protocolo en el cual se define la realización de un cronograma de ejecución (CE).

En la presente investigación se llevará a cabo el diseño e implementación del cronograma de ejecución, para el módulo de diseño, del proyecto Ensayos Clínicos. Se partirá de un análisis previo realizado por la investigación precedente, en la cual se realizó la modelación del negocio, el levantamiento de requisitos, el modelo de análisis, la propuesta de las interfaces de usuario, y una propuesta de diseño, la cual será redefinida debido a la introducción del marco de trabajo Symfony. De ahí que el trabajo presente se centre en la realización de los artefactos correspondientes al rol de diseñador e implementador, dígase, diagramas de clases del diseño, diagramas de interacción, diagrama de despliegue y desarrollo de un prototipo ejecutable.

PALABRAS CLAVE
Ensayos Clínicos, CIM, Cronograma de Ejecución.

ÍNDICE

ÍNDICE DE FIGURA

ÍNDICE DE TABLAS

INTRODUCCIÓN

El cáncer es una de las principales causas de muerte en todo el mundo. De los 58 millones de muertes que se registraron en el mundo en 2005, 7,6 millones (13%) se debieron al cáncer. [1]

El cáncer es el *nombre de las enfermedades en las cuales células anormales se multiplican sin control. Las células cancerosas pueden invadir los tejidos vecinos y pueden diseminarse a través del torrente sanguíneo y el sistema linfático a otras partes del cuerpo.* [2]

El 5 de diciembre de 1994 se inaugura el Centro de Inmunología Molecular (CIM). Este tiene como objetivo principal la producción de biofármacos, para el tratamiento de enfermedades crónicas no transmisibles, especialmente el tratamiento del cáncer. Los productos biofarmacéuticos que allí se fabrican tienen que ser aprobados y certificados, para ello se realizan Ensayos Clínicos.

Los Ensayos Clínicos son estudios que se efectúan con el fin de determinar o confirmar los efectos clínicos y farmacológicos en los seres humanos, estos presentan un protocolo, documento que establece la razón de ser del estudio, sus objetivos, diseño, métodos y el análisis previsto de sus resultados, así como las condiciones bajo las que se realizará y desarrollará el estudio. Por estas razones llevan asociados una gran documentación la cual debe ser almacenada por no menos de 15 años posterior al cierre del ensayo, de modo que pueda ser inspeccionada en cualquier momento por las agencias regulatorias.

La información de los ensayos clínicos es recogida en los Cuadernos de Recogida de Datos (CRD), donde cada ensayo clínico presenta un CRD por fase y estos se aplican a cada paciente involucrado.

Para llevar a cabo el control de los pacientes, así como la frecuencia de realización de los diferentes exámenes y el llenado de los modelos correspondientes, se realiza el cronograma de ejecución general, que contiene el conjunto de modelos con la información que se le debe recoger al paciente en cada visita; definiendo los días y el margen de tiempo permisible para el llenado de los mismos, además, establece los modelos que su llenado está condicionado porque se presenten los síntomas en un período de tiempo establecido.

A partir del cronograma general se define el cronograma específico de cada paciente, quedando establecida la fecha de llenado de cada modelo a partir del momento en que es incluido el paciente. El llenado de los modelos planificados puede ser afectado por la presencia de eventos adversos severos, abandono voluntario del EC por parte del paciente, fallecimiento o cualquier otra causa que impida continuar con el tratamiento establecido.

La confección del cronograma de ejecución se realiza manualmente, provocando interpretaciones erróneas por parte de los especialistas, esto implica inconsistencias en el llenado de los modelos e incumplimientos con respecto al margen de tiempo permisible para el llenado de los mismos; además los médicos definen, según su criterio, el tratamiento a seguir por el paciente y como tratarlo frente a la presencia de eventos adversos, lo cual trae como consecuencia la falta de estandarización y el incumplimiento del protocolo establecido.

Debido a estos problemas un grupo de analistas se dieron la tarea de definir los procesos de gestión de la confección del cronograma y como resultado obtuvieron 27 requisitos funcionales, 9 requisitos no funcionales, 7 casos de uso con sus descripciones correspondientes y una propuesta de diseño, la cual será redefinida debido a la introducción del marco de trabajo Symfony; quedando la necesidad de realizar el diseño e implementación de dichos casos de uso, para la confección del diseño del submódulo cronograma de ejecución.

Dada la situación problemática para este trabajo se define el siguiente problema científico:

¿Cómo obtener un módulo funcional a partir de los requerimientos identificados en el módulo de diseño para la confección del cronograma de ejecución del Sistema de Manejo de Datos de Ensayos Clínicos Cubano?

Con vista a la solución del problema anterior se plantea como objeto de estudio:

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

A partir del objeto de estudio se delimita el siguiente campo de acción:

Los procesos de gestión de la información de aplicaciones web para el cronograma de ejecución de los Ensayos Clínicos.

Con el fin de solucionar el problema planteado anteriormente se define como objetivo general de este trabajo:

Desarrollar el diseño e implementación del submódulo Cronograma de Ejecución del módulo de Diseño para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.

El objetivo general se desglosa en los siguientes objetivos específicos:

- **Diseñar el submódulo Cronograma de Ejecución del módulo de Diseño para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.**

- **Implementar el submódulo Cronograma de Ejecución del módulo de Diseño para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano. .**

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes tareas:

- **Revisión de la solución propuesta por la investigación precedente.**
- **Estudio de aplicaciones Web existentes para el Manejo de Datos de Ensayos Clínicos.**
- **Estudio y selección de las de las herramientas a utilizar para la implementación del submódulo cronograma de ejecución.**
- **Realización de los diagramas de clases del diseño.**
- **Elaboración de los diagramas de interacción.**
- **Realización del diagrama de clases persistentes.**
- **Elaboración del diagrama Entidad-Relación.**
- **Descripción de las tablas de la Base de datos.**
- **Elaboración del diagrama de despliegue.**
- **Elaboración del mapa de navegación.**
- **Implementación del submódulo cronograma de ejecución.**

El documento estará estructurado con los siguientes capítulos:

Capítulo 1. Fundamentación teórica: En este capítulo se brinda el estado del arte del objeto de estudio; se fundamentan las metodologías, tecnologías y herramientas utilizadas para el desarrollo de la aplicación web.

Capítulo 2. Propuesta de solución: En este capítulo se describe el estilo arquitectónico a utilizar y se definen los artefactos correspondientes al rol de diseñador e implementador dígase diagrama de clases del diseño, diagramas de interacción, diagrama de despliegue, y quedará reflejado el código fuente de las principales clases.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se brinda el estado del arte de los procesos de desarrollo de los Sistemas de Manejo de Datos Ensayos Clínicos con el fin de darle solución al objeto de estudio definido en la presente investigación; se fundamentan las metodologías, tecnologías y herramientas utilizadas para el desarrollo de la aplicación Web. Además se muestran los artefactos desarrollados por los roles de diseñador y programador según RUP.

1.1 ¿Qué es un ensayo clínico?

Tipo de estudio de investigación que comprueba si un enfoque médico nuevo funciona bien en las personas. Estos estudios prueban nuevos métodos de detección, prevención, diagnóstico o tratamiento de una enfermedad [3]

Todas las características del ensayo clínico estarán íntegramente definidas en un protocolo y la realización del ensayo se ajustará al contenido del protocolo autorizado, en este último se define el Cronograma de Ejecución del estudio clínico.

1.2 ¿Qué es un cronograma de ejecución?

Un cronograma de ejecución para un ensayo clínico esta dado por la lista de todos los elementos terminales del ensayo con su fecha prevista de inicio y fin. También se puede considerar como el detalle minucioso de las actividades necesarias para dar cumplimiento al ensayo, todo esto con el objetivo de alcanzar las metas propuestas en un período de tiempo establecido.
El Cronograma de Ejecución (CE) define el tiempo durante el cual serán considerados los pacientes incluidos en el ensayo, las diferentes etapas del mismo, así como la planificación tanto de los modelos sintomáticos como de los asintomáticos.

Las etapas son períodos de tiempo dentro del cronograma que permiten identificar los diferentes estados por los cuales pasan los pacientes, ejemplo: Tratamiento, Seguimiento, Consolidación…

Existen modelos que se planifican para una fecha determinada y con un margen de tiempo permitido para el llenado del mismo, a estos se les denomina asintomáticos; en cambio los modelos que se desconoce cuando puede ocurrir el evento que provoque su llenado son denominados sintomáticos, a estos se les especifica el período de tiempo en el cual es relevante recoger dicha información y la frecuencia con que se pueden presentar el mismo.

1.3 Aplicaciones Web para Sistemas de Manejo de Datos de Ensayos Clínicos.

1.3.1 Hipócrates

Hipócrates fue creado para el control burocrático y archivo total de consultas médicas adaptado a la dinámica habitual de una consulta médica y sin necesidad de conocimientos informáticos. Aporta un sistema de registro adaptado, perfectamente, al tiempo y mecánica de la consulta médica. Es capaz de registrar absolutamente toda la actividad de consulta, datos, historia, pruebas complementarias, prescripciones de los enfermos. [4]

Hipócrates no resulta una vía de solución para los problemas que presentan los investigadores del CIM porque el software es propietario y una reproducción o alteración en la estructura de la misma llevaría consigo una demandada judicial. Este a su vez ofrece la posibilidad de ver el diagnóstico codificado y las actividades necesarias para el diagnóstico definitivo y acciones futuras a realizar. Pero no realiza un cronograma general para el ensayo, sino que es definible para cada usuario.

1.3.2 OpenClinica

OpenClinica es un sistema para manejar estudios clínicos, es de código abierto y basado en Java. Presenta una interfaz fácil de usar, permite la extracción de datos y filtrado de archivos, posibilita el intercambio de recursos de forma segura y transparente, admite exportación de herramientas para la migración de datos clínicos en las hojas de cálculo Excel y base de datos locales. OpenClinica presenta una infraestructura escalable, utiliza Java J2EE, con base de datos relacionales PostgreSQL incluidas y Oracle 10G, además usa el patrón de arquitectura Modelo Vista Controlador y Subversion como controlador de versiones.

OpenClinica presenta un cronograma de ejecución el cual no responde a las necesidades del CIM debido a que este no presenta un cronograma estándar para cada paciente incluido sino que ha medida que los pacientes se van incluyendo en el ensayo el médico le va asociando modelos. Además no tratan de manera diferente la programación de modelos sintomáticos y asintomáticos simplemente no los tienen en cuenta.

Después del estudio realizado a estos sistemas se llegó a la conclusión de que los mismos no cumplen con las necesidades del Centro de Inmunología Molecular, ya que no presentan una herramienta capaz de controlar el cumplimiento del protocolo del ensayo. Por lo que se hace necesario el desarrollo de una aplicación que permita reflejar y manejar la mayor cantidad de información posible, definida en el protocolo del ensayo.

1.4 Herramientas y tecnologías de desarrollo.

En este epígrafe se presentan las tecnologías y herramientas usadas en la investigación. La selección de las mismas fue una decisión en conjunto con los arquitectos del proyecto, partiendo del estudio realizado por la investigación precedente y teniendo en cuenta la necesidad del país de llevar a cabo la migración a software libre. Para el lector interesado en profundizar al respecto se recomienda remitirse al documento de arquitectura del proyecto.

1.4.1 Metodología de desarrollo.

Proceso Unificado de Desarrollo de Software (RUP, Rational Unified Process)

El Proceso Unificado de desarrollo de software (RUP), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP *es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto.*

…los aspecto definitorios del proceso unificado se resumen en tres frases claves : dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. [5]

RUP se divide en 4 fases de desarrollo del software:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario.

- **Transición**: El producto se convierte en versión beta, en esta versión se crea un grupo reducido de usuarios con experiencia para probar el producto e informan los defectos e ineficiencias del mismo. Puede implicar reparación de errores.

RUP define las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

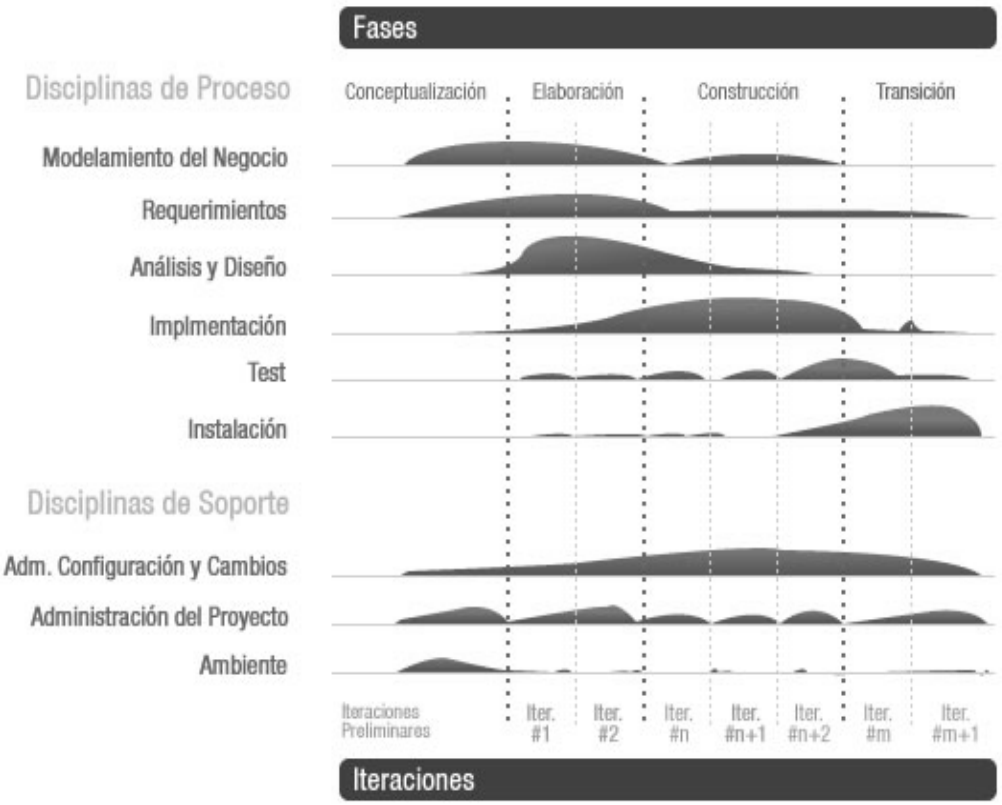

Figura 1 RUP en dos dimensiones.

Flujos de trabajo:

• ***Modelamiento del negocio:*** *Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.*

• ***Requerimientos:*** *Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.*

• ***Análisis y diseño:*** *Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.*

• ***Implementación:*** *Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.*

• ***Prueba (Test):*** *Busca los defectos a los largo del ciclo de vida.*

• ***Instalación:*** *Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.*

• ***Administración de configuración y cambios:*** *Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.*

• ***Administración del proyecto:*** *Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.*

• ***Ambiente:*** *Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.* [6]

Dentro de los flujos de trabajo definidos por RUP existen diferentes roles y artefactos de los cuales se implementarán en este trabajo:

Análisis y diseño

Trabajadores:

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

Artefactos

- Mapa de navegación
- Prototipos de interfaz de usuario.
- Paquetes de diseño

- Clases del diseño.
- Realización de los casos de uso del diseño
- Diagramas de despliegue.
- Vista lógica.
- Modelo de datos.

Implementación

Trabajadores:

- Arquitecto
- Programador

Artefactos:

- Modelo de implementación
- Prototipo ejecutable

Lenguaje Unificado de Modelado (UML) (versión 2.0)

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software, permite la modelación de sistemas con tecnología Orientado a Objetos. En UML podemos encontrar elementos (abstracciones que constituyen los bloques básicos de construcción), relaciones (Ligan los elementos) y diagramas (Es la representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas).

1.4.2 Herramientas de Modelado Visual.

Visual Paradigm SWITE. (versión 3.1)

Es una herramienta CASE que utiliza UML como herramienta de modelado, es multiplataforma y a su vez software libre. Este proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de una forma rápida. Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal.

1.4.3 Herramientas de desarrollo.

Las herramientas de programación para páginas Web mejoran cada día, por lo que merecen especial atención. En la presente investigación se trabajará con las siguientes herramientas de programación: Kompozer como editor de código HTML, eclipse como IDE de desarrollo para PHP.

Eclipse (versión 3.2.1.1)

Eclipse es una comunidad de código abierto cuyos proyectos están centrados en la construcción de una plataforma de desarrollo extensible, módulos de ejecución y la aplicación de marcos para la construcción, el despliegue y la gestión de software en todo el ciclo de vida del software. [7]

Eclipse es multiplataforma (GNU/Linux, Solaris, Mac OSX, Windows), permite el control de versiones con Subversión, compilación en tiempo real, presenta estructura de plug-in que hace sencillo añadirle nuevas características y funcionalidades, así como Asistentes (wizards) para la creación, exportación e importación de proyectos; para generar esqueletos de códigos.

El proyecto PDT (PHP Development Tools, eclipse) ha tenido una gran respuesta entre los desarrolladores de PHP ya que tan solo en su versión actual (1.0) se pueden encontrar las siguientes características:

- Editor sensible al contexto, este proporciona autocompletado, asistente y resalta el código.
- Integración con el modelo del proyecto Eclipse, permitiendo inspeccionar el uso de las vistas del contorno del fichero y del proyecto, así como la nueva vista PHP Explorer.
- Presenta soporte para el debug incremental del código de PHP.
- En este IDE se pueden encontrar extensos frameworks y APIs que permiten a los desarrolladores y fabricantes de software independientes extender fácilmente eclipse PDT para crear nuevas herramientas orientadas al desarrollo de PHP.

Kompozer

Kompozer es una herramienta de creación de páginas Web. Es un software libre, estable, fácil de usar, puede editar archivos de gran tamaño y presenta muchas características parecidas a FrontPage e incluso al Dreamweaver.

Algunas de las herramientas de KompoZer son:

Administrador de sitios FTP: Cualquier sitio que el usuario haya especificado en sus Opciones de Publicación, podrá ser navegado en una barra lateral. También permite filtrar y mostrar archivos o solo documentos HTML o imágenes.

Selector de colores: Se puede elegir entre el selector de colores verde, azul y rojo para crear la tonalidad deseada, así como también elegir la saturación del matiz y el brillo. Si se desea podrá elegirse el color con el mouse.

Pestañas: Una de las herramientas más conocidas de Mozilla esta disponible para KompoZer facilitando el trabajo y brindándole la posibilidad al usuario de realizar acciones de manera más fluida navegando entre pestañas. Se podrá utilizar la herramienta deshacer y rehacer independientemente en cada una de las pestañas.

Barras de tareas personalizables: se podrán elegir los botones que aparezcan en las barras y los que no según los gustos y/o necesidades.

Presenta edición visual: A medida que el usuario va definiendo tipos de letra, enlaces, o insertando imágenes se irá generando código HTML en segundo plano permitiendo que se visualice en el editor como mismo se verá en el navegador.

1.4.4 Tecnologías de desarrollo.

PHP(versión 5.2.0)

PHP, cuyas siglas responden a un acrónimo recursivo (PHP: Hypertext Pre-processor) es un lenguaje de script usado principalmente para scripts a ejecutar en servidores web. PHP es un lenguaje multiplataforma. Este lenguaje es libre, por lo que se presenta como una alternativa de fácil acceso para todos. PHP no requiere definición de tipos de variables.

Es bueno destacar que PHP presenta un gran número de gestores de bases de datos a los que puede acceder.

- Adabas D, dbm, dBase, filePro, Hyperwave, Informix, InterBase, LDAP, Microsoft SQL server, mSQL, MySQL, ODBC, Oracle, PostgreSQL, Solid, Sybase.

La versión que se utilizará en esta investigación es PHP5.2.0, debido a que este presenta un soporte sólido y REAL para Programación Orientada a Objetos (POO), mejoras de rendimiento, mejoras del soporte a XML (XPath, DOM) y soporte integrado para SOAP.

Apache (versión 2.2.3)

Apache es el servidor web mas usado en ámbitos empresariales, tecnológicos y educativos esto esta dado por su calidad de servicio, su robustez y estabilidad.

Apache tiene disimiles características las que se muestran a continuación:

- Es una herramienta multiplataforma.
- Ofrece tecnología libre y de código abierto, otorgándole una transparencia y dando la posibilidad de conocer que es lo que realmente se está instalando.
- Es altamente configurable y de diseño modular, capaz de ampliar su funcionalidad y calidad de servicios.
- Trabaja en conjunto con gran cantidad de Lenguajes de Programación interpretados como PHP (PHP Hypertext Pre-processor), Perl, soporte con CGI (Common Gateway Interface), Java, JSP (Java Server Pages) y otros lenguajes de script.
- Permite configurar y personalizar cada uno de los mensajes de error que se pueden producir por la utilización del servidor.
- Presenta archivos Log, en donde registra gran cantidad de información global del sistema, errores producidos en un determinado tiempo, en la cual estos archivos son de gran importancia para los administradores de sistemas y pueden influenciar de alguna manera las políticas de seguridad debido a la gran cantidad de información que contiene.
- Se puede encontrar gran cantidad de documentos, ejemplos y ayuda en internet en todos los idiomas referido al servidor web apache.

1.4.5 Gestor de Base de datos.

Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. [8]

PostgreSQL (versión 8.2.6.5)

PostgreSQL es un servidor de base de datos libre. El desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). Este gestor de base datos presenta una alta concurrencia y

una amplia variedad de tipos nativos así como ventajas que hacen de él una vía factible para la creación de base de datos, ejemplos de ellas:

Instalación ilimitada

- PostgreSQL se distribuye bajo licencia BSD, la cual permite su uso y distribución sin pago de licencias en aplicaciones tanto comerciales como no comerciales.

Extensible

- El código fuente está disponible a cualquier usuario sin costo alguno; si se necesita extender o personalizar PostgreSQL de alguna manera, puede hacerse con un mínimo esfuerzo y sin costos adicionales.

Multiplataforma

- PostgreSQL está disponible para los principales sistemas operativos (Linux, Windows, Mac OS X, Solaris, BSD, Tru64).

Control de Bloqueos

- PostgreSQL usa una estrategia de almacenamiento de filas llamada Control de Concurrencia Multi-Versión (MVCC), esta tecnología evita bloqueos innecesarios cuando se trata de acceder a datos que están siendo modificados.

Write Ahead Logging (WAL)

- La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que de producirse una caída de la base de datos, existirá un registro de las transacciones a partir del cual se podrá restaurar la misma.

Escalabilidad en cuanto a usuarios concurrentes (Cliente/Servidor)

- PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor, definiendo un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectarse a PostgreSQL.

1.4.6 Framework.

Los framework definen una filosofía de trabajo. Proporcionan librerías y funciones que permiten ahorrar trabajo y tiempo; los marcos de trabajo permiten producir aplicaciones más fáciles de mantener y evitan código duplicado.

Symfony (versión 1.0.10)

Symfony es un framework libre, que usa como patrón el Modelo-Vista-Controlador (MVC). Proporciona estructura al código fuente, forzando a crear código más legible y más fácil de mantener; facilita la programación de aplicaciones ya que encapsula operaciones complejas en instrucciones sencillas. Symfony utiliza Propel como herramienta para el mapeo objeto-relacional (ORM). Permite generar las vistas a partir del modelo de datos. Incluye Helper que permiten a partir de PHP generar todo el código cliente (css, javascript, html). También incluye división del sistema por módulos y presenta un buen soporte para pruebas unitarias y funcionales.

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

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- 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.

1.4.7 Herramienta para Control de Versiones.

Subversion (versión 1.4.2).

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS (Concurrent Versions System). Este software es libre, se conoce como SVN. Una característica importante de Subversion es que, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

1.5 Conclusiones

En este capítulo se presenta una panorámica del estado actual de los Sistemas de Manejo de Datos de Ensayos Clínicos; quedan especificados artefactos y roles a desarrollar así como las herramientas, tecnologías y metodologías a utilizar.

- Para el desarrollo del sistema se utiliza la metodología de desarrollo RUP la cual utiliza el lenguaje unificado de modelado, y se emplea Visual Paradigm como herramienta CASE para el desarrollo de la Ingeniería de Software.
- Se Usará Kompozer como editor de código HTML y Apache como servidor web.
- Se empleará eclipse como IDE de desarrollo para PHP5 el cual constituirá el lenguaje de programación para el sistema en cuestión.
- El marco de trabajo es Symfony facilitando de esta forma la programación de la aplicación Web.
- Se utilizará Subversion como herramienta para el control de versiones
- Para el almacenamiento y gestión de los datos que se almacenan se consideró utilizar PostgreSQL porque funciona en múltiples plataformas y se puede acceder a él de forma gratuita.

CAPÍTULO 2: SOLUCIÓN PROPUESTA

En este capítulo se describe los patrones de diseño y arquitectura a utilizar y se definen los artefactos correspondientes al rol de diseñador dígase diagrama de clases del diseño, los diagramas de interacción y diagrama de despliegue, además quedará reflejado el código fuente de los principales métodos.

Para el desarrollo del trabajo presente se tomará como punto de partida los resultados obtenidos por la investigación precedente, donde se definieron 27 requisitos funcionales y 7 casos de uso. Para obtener más detalles referirse a los anexos 1y 2 respectivamente.

2.1 Patrones de Diseño y Arquitectura.

2.1.1 Patrón Arquitectónico.

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software; su importancia esta dada en que trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

En el Modelo-Vista-Controlador se pueden encontrar los siguientes elementos:

- Modelo: datos y reglas de negocio
- Vista: muestra la información del modelo al usuario
- Controlador: gestiona las entradas del usuario

Las responsabilidades de cada unas de sus partes son:

- El modelo es el responsable de:
  - Acceder a la capa de almacenamiento de datos.
  - Define las reglas del negocio.
  - Ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente.
- El controlador es responsable de:
  - Recibir los eventos de entrada.

- Contiene reglas de gestión de eventos.

- Las vistas son responsables de:
  - Recibir datos del modelo y mostrarlos al usuario.

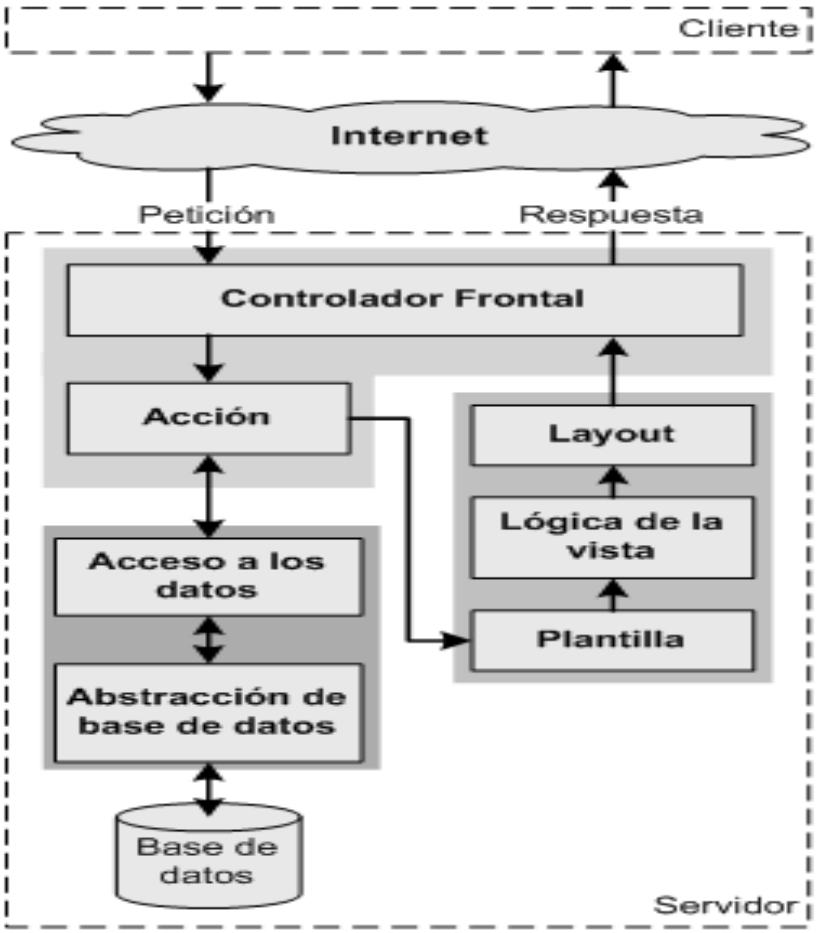

Figura 2 Patrón arquitectónico MVC

2.1.2 Patrones de diseño.

Los patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

- ***Experto:*** *Asigna una responsabilidad al experto en la información, o sea, se asigna la responsabilidad a la clase que cuenta con la información necesaria para cumplir la responsabilidad.*
- ***Creador:*** *Asigna a la clase B la responsabilidad de crear una instancia A en uno de los siguientes casos:*
  - *B agrega los objetos A.*
  - *B contiene los objetos A.*
  - *B registra la instancia de los objetos A.*
  - *B utiliza específicamente los objetos A.*
  - *B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado.*
- ***Alta Cohesión:*** *Asigna una responsabilidad de modo que la cohesión siga siendo alta.*
- ***Bajo Acoplamiento:*** *Asignar una responsabilidad para mantener bajo acoplamiento.*
- ***Controlador:*** *Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.*
- ***Polimorfismo:*** *las responsabilidades del comportamiento se asignan mediante operaciones polimórficas a los tipos en el que el comportamiento presenta variantes.*
- ***Fabricación Pura:*** *asigna un conjunto de responsabilidades a una clase artificial que no representa nada en el dominio del problema. Las responsabilidades que se asignan brindan soporte a una alta cohesión y bajo acoplamiento.*
- ***Indirección:*** *Asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios y estos no terminen directamente acoplados.*
- ***No hables con extraños:*** *Asigna la responsabilidad a un objeto del cliente para que colabore con un objeto indirecto, de modo que el cliente no necesita saber nada del objeto indirecto.* [9]

Los patrones de diseño brindan una solución probada y documentada a problemas comunes en el desarrollo del software. Estos presentan un nombre, describen el problema y su solución.
Los patrones de diseño se clasifican en:

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

- **Patrones Estructurales:** Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- **Patrones de Comportamiento:** Describen objetos o clases así como la comunicación entre ellos.
- **De clase**: Esta basado en la herencia de clases.
- **De objeto**: Basados en la utilización dinámica de objetos.

Singleton (Creacional)

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

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

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

- *La propia clase es responsable de crear la única instancia.*
- *Permite el acceso global a dicha instancia mediante un método de clase.*
- *Declara el constructor de clase como privado para que no sea instanciable directamente.* [10]

Fachada (Estructural)

El patron fachada es usado para *definir una sola clase que unifique las interfaces y asignarle la responsabilidad de colaborar con el subsistema.* [11]

Decorator (Estructural)

El patrón de diseño tiene la intención de decorar las responsabilidades de un objeto dinámica y transparentemente a sus clientes. Alternativa a la herencia para decorar la responsabilidad de un subconjunto de objetos. [12]

2.1.3 El uso de patrones en la investigación

La presente investigación usa como marco de trabajo Symfony el cual implementa el patrón arquitectónico MVC, el mismo define la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador. Symfony implementa el patrón Decorator (Envoltorio) ya que este añade funcionalidad a una clase, dinámicamente. Este es usado en la vista para agregarle a las plantillas el código del layout (plantilla global). O sea, para decorar las plantillas con el layout.

También se tuvo en cuenta los patrones de asignación de responsabilidades, ejemplo de ello es el patrón bajo acoplamiento debido a que las clases utilizadas presentan poca dependencia entre ellas y no presentan herencias profundas. Se utilizó el patrón controlador con el propósito de centralizar las actividades y mantener una alta cohesión ya que mantiene una labor única dentro del submódulo. Se utiliza además el patrón fabricación pura, debido a la presencia de una clase artificial la cual da soporte a la reutilización; así como la existencia de una clase intermedia que permitirá la comunicación entre los módulos evidenciando una indirección; es usado el patrón Fachada proporcionando una clase que unifica la interfaz del submódulo, esta tiene la responsabilidad de interactuar con el mismo; permitiendo ocultar su complejidad y que en caso de producirse cambios en el módulo solo haya que actualizar la clase Fachada sin que el cambio afecte a los demás módulos; para proporcionar un único punto de acceso global se utilizó el patrón Singleton.

2.2 Modelo de diseño.

Es un modelo de objeto que describe la realización de guiones de uso, y sirve como una abstracción del modelo de implementación y el código fuente. El modelo de diseño se utiliza como entrada esencial para actividades en implementación y prueba. [12]

2.2.1 Paquetes del diseño

Un paquete de diseño es una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes. Este es usado para estructurar el modelo diseño, dividiéndolo en partes más pequeñas.

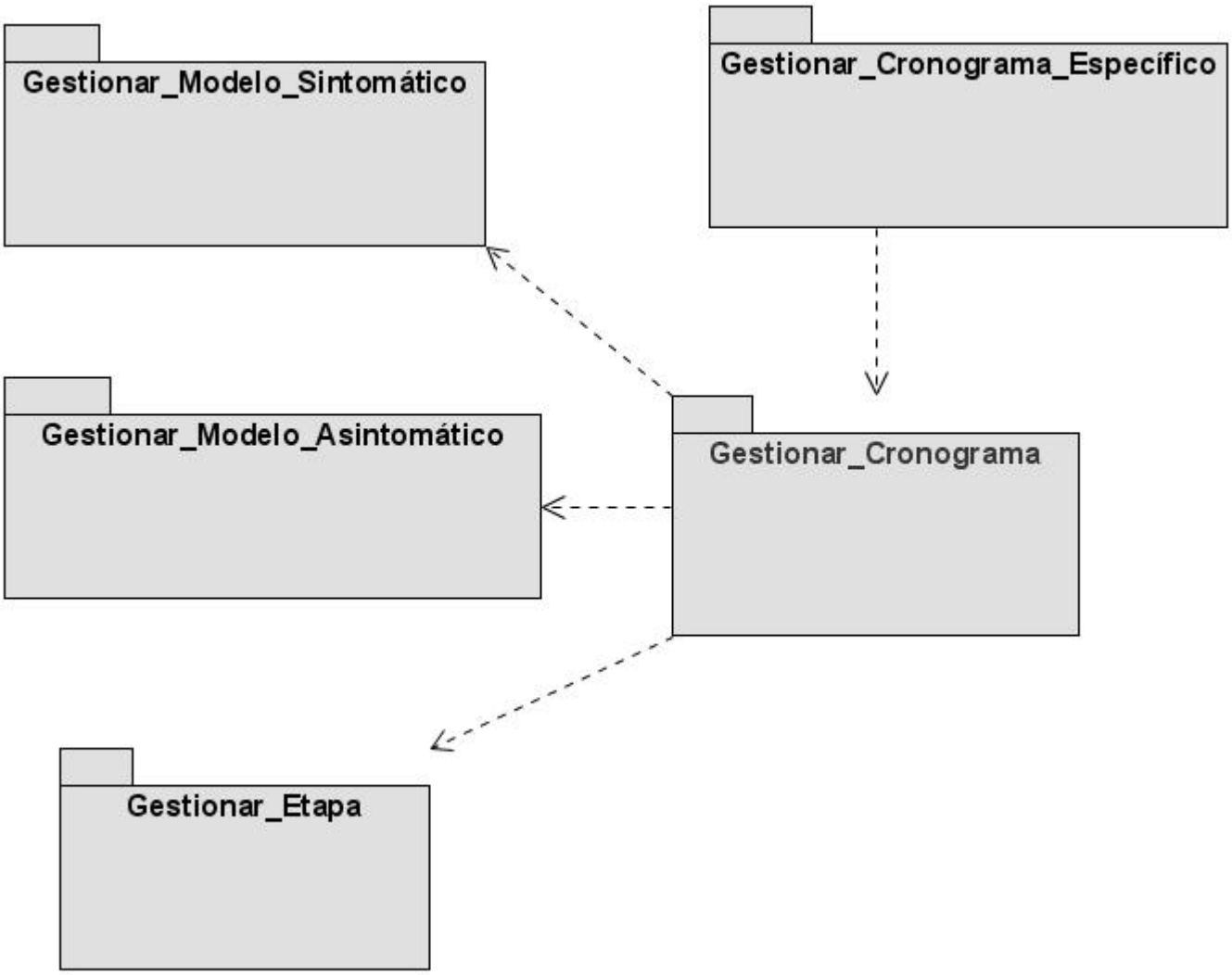

Figura 3 Paquetes del diseño

2.2.2 Descripción de las clases.

Tabla 1. Descripción de las clases del diseño: cron_crudCronogramaActions.

Nombre: cron_crudCronogramaActions	
Tipo: Clase controladora, correspondiente al paquete Gestionar Cronograma, encargada de controlar las acciones del usuario y realizar los cambios apropiados en la vista o el modelo.	
Atributos	**Tipo**

Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	En caso de estar creado el cronograma redirecciona a la página mostrar, sino, redirecciona a crear.
preExecute()	Verifica si esta Aprobado el cronograma, en caso de estarlo impide que se realicen cambios al mismo, redireccionando a la página aprobado.
executeShow()	Llama al método retrieveByPK de la clase CCronogramaPeer y muestra en la client_page correspondiente los datos del cronograma no se modifican los datos.
executeCreate ()	Recoge el id del cronograma y del ensayo al que se le va a crear el cronograma, muestra en la client_page de edit un formulario para insertar un nuevo cronograma de ejecución.
executeEdit()	Llama al método retrieveByPK de la clase CCronogramaPeer y muestra en la client_page correspondiente los datos del cronograma seleccionado por el diseñador del CRD electrónico, permitiendo modificar los datos del mismo.
executeUpdate()	Llama al método retrieveByPK de la clase CCronogramaPeer, si este método obtiene un CCronograma se edita este, sino, se crea uno nuevo, permitiendo ser rehusado tanto para crear como para editar un CCronograma, Los datos son guardados mediante el método save(). Redirecciona a la client_page show.
executeShowCronogramaGeneral ()	Obtiene el CCronograma a mostrar mediante el método retrieveByPK() de CCronogramaPeer. Obtiene un arreglo asociativo de refModeloVisita Llamando al método getArregloAsociativoRMV() del CCronograma obtenido. Se Obtiene un arreglo de

	Etapas. Todos estos elementos son recogidos para mostrar el cronograma general en la client_page showCronogramaGeneral.

Tabla 2. Descripción de las clases del diseño: cron_crudEtapaActions.

Nombre: cron_crudEtapaActions	
Tipo: Clase controladora, correspondiente al paquete Gestionar Etapa, encargada de controlar las acciones del usuario y realizar los cambios apropiados en la vista o el modelo.	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de las etapas existente en la base datos.
preExecute()	Verifica si esta Aprobado el cronograma, en caso de estarlo impide que se realicen cambios al mismo, redireccionando a la página aprobado.
executeList()	Llama al objeto Criteria, que muestra en la client_page correspondiente a la lista de las etapas planificadas por el diseñador del CRD electrónico.
executeShow()	Llama al método retrieveByPK de la clase CEtapaPeer y muestra en la client_page correspondiente los datos de la etapa seleccionada por el diseñador del CRD electrónico; no se modifican los datos.
executeCreate ()	Recoge el id del cronograma y del ensayo al cual pertenecerá la etapa, muestra en la client_page de edit un formulario para insertar una nueva etapa.
executeEdit()	Llama al método retrieveByPK de la clase CEtapaPeer y muestra en la client_page correspondiente los datos de la etapa seleccionada por el diseñador del CRD electrónico, permitiendo

	modificar los datos de la misma.
executeUpdate()	Llama al método retrieveByPK de la clase CEtapaPeer, si este método obtiene una Etapa se edita esta, sino, se crea una nueva, permitiendo ser rehusado tanto para crear, como para editar una CEtapa. Los datos son guardados mediante el método save(). Redirecciona a la client_page show.
executeDelete()	Llama al método retrieveByPK de la clase CEtapaPeer, elimina los datos seleccionados y redirecciona a la página donde se encuentra el listado de todas las etapas.

Tabla 3. Descripción de las clases del diseño: cron_crudRefModeloVisitaActions.

Nombre: cron_crudRefModeloVisitaActions	
Tipo: Clase controladora, correspondiente al paquete Referencia Modelo visita, encargada de controlar las acciones del usuario y realizar los cambios apropiados en la vista o el modelo.	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de modelos planificados, existentes en la base datos.
preExecute()	Verifica si esta Aprobado el cronograma, en caso de estarlo impide que se realicen cambios al mismo, redireccionando a la página aprobado.
executeList()	Llama al objeto Criteria, que muestra en la client_page correspondiente el listado de los modelos planificados por el diseñador del CRD electrónico.
executeShow()	Llama al método retrieveByPK de la clase CRefModeloVisita y muestra en la client_page correspondiente los datos del modelo planificado

	seleccionado por el diseñador del CRD electrónico; no se modifican los datos.
executeCreate ()	Recoge el id del cronograma y del ensayo al cual pertenecerá el modelo, muestra en la client_page de edit un formulario para insertar un nuevo modelo planificado.
executeEdit()	Llama al método retrieveByPK de la clase CRefModeloVisita y muestra en la client_page correspondiente los datos del modelo planificado seleccionado por el diseñador del CRD electrónico, permitiendo modificar los datos de la misma.
executeUpdate()	Llama al método retrieveByPK de la clase CRefModeloVisita, los datos que obtiene modificados o no los guarda llamando el método save() y redirecciona a la client_page Show.
executeDelete()	Llama al método retrieveByPK de la clase CRefModeloVisita, elimina los datos seleccionados y redirecciona a la página donde se encuentra el listado de todas los modelos planificados
executeProgramarModeloPeriodo()	Recoge el id del cronograma y del ensayo al cual pertenecerá el modelo, llama al método retrieveByPK de CCronogramaPeer muestra en la client_page de programar modelo en período un formulario para insertar.
executeGuardarModeloPeriodo()	Recoge los datos los datos enviados del formulario, busca el cronograma apropiado mediante el método retrieveByPK de CCronogramPeer y llama al método guardarModeloPeriodo() pasandole los datos recogidos del formulario.
executeEditSintomatico()	Llama los métodos retrieveByPK de las clases CRefModeloVisitaPeer, PmodeloPeer, CCronogramaPeery muestra en la client_page correspondiente los datos del modelo planificado

	seleccionado por el diseñador del CRD electrónico, permitiendo modificar los datos de la misma.
executeCreateSintomatico()	Recoge el id del cronograma y del ensayo al cual pertenecerá el modelo, llamo el método retrieveByPK de la clase CCronogramaPeer y muestro en la client_page de edit un formulario para insertar un nuevo modelo planificado.
executeUpdateSintomatico()	Llama al método retrieveByPK de la clase CCronogramaPeer, los datos que obtiene modificados o no los guarda llamando el método guardarRefModeloVisitaSintomatica y redirecciona a la client_page Show.
executeShowSintomatico()	Llama al método retrieveByPK de la clase CRefModeloVisita y muestra en la client_page correspondiente los datos del modelo planificado seleccionado por el diseñador del CRD electrónico; no se modifican los datos.
executeBuscar()	Llama al método retrieveByPK de la clase CRefModeloVisita y muestra la client_page a la que se le introducirá el criterio de búsqueda.
executeResultadosAsintomaticos()	Busca los modelos asintomáticos planificados que cumplan con los parámetros de búsqueda.
executeResultadosSintomaticos()	Busca los modelos sintomáticos planificados que cumplan con los parámetros de búsqueda.

Tabla 4. Descripción de las clases del diseño: fachadaCronograma.

Nombre: fachadaCronograma	
Tipo: Clase del modelo encargada de brindar un punto de acceso sencillo al submódulo, permitiendo ocultar la complejidad del mismo, y sirviendo como punto de entrada único.	
Atributos	**Tipo**

Responsabilidades	
Nombre:	**Descripción:**
getInstancia()	Garantiza que la clase sólo tenga una única instancia, proporcionando un punto de acceso global a la misma.
crearCronogramaPaciente()	Obtiene todas las ref_modelo_visita definidas para el Cronograma General, y registra cada una de ellas en la tabla pac_modelo definiéndole las fechas ideales en que se deben realizar las mismas por dicho paciente.
getArregAsocRefModelo()	Teniendo en cuenta un objeto criteria, devuelve un arreglo de modelos, donde las llaves son el id del modelo y el id_ref_modelo y el valor es el objeto ref_modelo, o sea, [id_modelo][id_ref_modelo]= ref_modelo
getArregloModelos()	Teniendo en cuenta un objeto criteria, devuelve un arreglo de modelos, donde la llave es el id del modelo y el valor el nombre del modelo. Incluye la opción 'Seleccione' para las listas desplegables.
getArregId_NombreModelos()	Teniendo en cuenta un objeto criteria, devuelve un arreglo asociativo de modelos, donde la llave es el id del modelo y el valor el nombre del modelo.
estaCronogramaAprobado()	Devuelve si está o no aprobado el cronograma.
notificarInclusionPaciente()	Verifica si el Cronograma está aprobado, en caso de no estarlo envía una excepción. Si está aprobado habilita los modelos de evaluación inicial e inclusión devolviendo este último.
estaGeneradoCronograma()	Se le pasa a este método un paciente para comprobar si está o no generado el cronograma específico para él.
notificarInsercionModeloPaciente()	Método llamado cuando el paciente completa algún modelo. Determina a partir del tipo de modelo cuales

	son las acciones a realizar, llamando el método adecuado en ControladorCronogramaEspecifico.
getCronograma()	Devuelve un cronograma teniendo en cuenta un objeto criteria.
listadoModelosSintomaticos()	Lista los modelos sintomáticos que puede llenar un paciente hasta una fecha determinada. Se lista por la fecha de planificación. Si no se le pasa una fecha los devuelve todos los que no estén interrumpidos.
listadoModelosAsintomaticos()	Lista los modelos asintomáticos que puede llenar un paciente hasta una fecha determinada. Se lista por la fecha de planificación. Si no se le pasa una fecha los devuelve todos los que no estén interrumpidos.

Tabla 5. Descripción de las clases del diseño: controladorCronogramaEspecifico.

Nombre: controladorCronogramaEspecifico.	
Tipo: Clase que controla el flujo de eventos del submódulo, a las clases del Cronograma Específico	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
habilitarModelosPaciente()	Le habilita a un determinado paciente un conjunto de CRefModeloVisita, recibidas como parámetro.
comprobarSiHaCompletadoAsintomaticos()	Retorna si el paciente ha completado al menos un modelo Asintomático.
getCriteriaTipoModeloEstado()	Este método devuelve un objeto criteria, de los PPacModelo que tengan un determinado comportamiento y estado.
addModeloFallecimiento()	Le adiciona un modelo de fallecimiento al

	paciente, interrumpiéndole todos los modelos restantes
addModeloInterrupcion()	Le adiciona un modelo de interrupción al paciente, interrumpiéndole todas las inmunizaciones restantes.
addModeloSintomaticoNoPlanificado()	Recoge todos los atributos necesarios para llenar un modelo sintomático y lo salva en la tabla p_pac_modelo.

Tabla 6. Descripción de las clases del diseño: ElementosComunes.

Nombre: ElementosComunes	
Tipo: Clase no representativa dentro del contexto de la solución, su objetivo es incrementar la reutilización de código, agrupando una serie de métodos comunes.	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getArregAsocPosiciones()	Dado un arreglo asociativo, devuelve un arreglo asociativo con la posición de cada llave([llave] = posición)
getElementosPagina()	Recibe un arreglo Asociativo y devuelve los elementos que estén en una página determinada, especificándole la cantidad de elementos que tiene cada página.
getLimiteInferiorPagina()	Recibe el número de la página, la cantidad de elementos que tendrá cada una de ellas y devuelve el primer elemento que debe tener la misma

Tabla 7. Descripción de las clases del diseño: CCronogramaPeer.

Nombre: CCronogramaPeer	
Tipo: Clase acceso a datos	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getArregAsocId_NombEtapa()	Recibe un arreglo de etapas y devuelve un arreglo asociativo, donde la llave es id_etapa y el valor el nombre de la misma.
getCriteria()	Devuelve una Criteria, que selecciona todas las ref_modelo_visita que se hagan en un día determinado, que sean de un modelo dado(id_modelo), y tengan cierto comportamiento. Si su último parámetro "comportamientoIgual" tiene valor false, se devuelven todas las que no tengan ese comportamiento. Además recibe el id_cronograma y el id_ensayo_clinico. Para más detalles referirse al epígrafe 2.4.2
getCriteriaSintomaticos()	Determina un objeto Criteria de todas las ref_modelo_vista sintomáticas que puedan hacerse un día determinado. Recibe el día, id_cronograma, y el id_ensayo_clinico
verificarSolapamientoDiaSintomatico()	Verifica si un día determinado en el cual se quiere planificar un modelo sintomático, se solapa en otro período sintomático ya definido para el mismo modelo.
haySolapamientoSintomatico()	Verifica si un modelo que se quiere planificar como sintomático, en un período de tiempo dado se solapa con otro ya definido.

getCriteriaAsintomaticaFiltro()	Devuelve una Criteria para seleccionar ref_modelo_visita asintomáticas, utilizando filtros. Por id_modelo, para un día dado, para un período de tiempo establecido, o que estén en una etapa dada. Si alguno de estos parámetros es NULL no se agrega el correspondiente filtro.
getCriteriaAsintomaticoEtapa()	Devuelve una Criteria para seleccionar ref_modelo_visita asintomáticas. Recibe el id_modelo deseado, y la etapa . Si alguno de estos parámetros es NULL no se agrega el correspondiente filtro.
getCriteriaAsintomaticoPeriodo()	Devuelve una Criteria para seleccionar ref_modelo_visita asintomáticas. Recibe el id_modelo, y el período de tiempo establecido. Si alguno de estos parámetros es NULL no se agrega el correspondiente filtro.
getCriteriaSintomaticosPeriodo()	Devuelve una Criteria para seleccionar ref_modelo_visita sintomáticas. Recibe el id_modelo, y el período de tiempo establecido. Si alguno de estos parámetros es NULL no se agrega el correspondiente filtro.
getCriteriaSintomaticoEtapa()	Devuelve una Criteria para seleccionar ref_modelo_visita sintomáticas. Recibe el id_modelo deseado, y la etapa. Si alguno de estos parámetros es NULL no se agrega el correspondiente filtro.
getCriteriaSintomaticaFiltro()	Devuelve una Criteria para seleccionar ref_modelo_visita sintomáticas, utilizando filtros. Por id_modelo, para un día dado, para un período de tiempo establecido, o que

	estén en una etapa dada. Si alguno de estos parámetros es NULL no se agrega el correspondiente filtro.
getRefModeloCorrecta()	Recibe el ensayo clínico, el id_modelo, el comportamiento, el tiempo de llenado y si afecta_cronograma. Si ya existe una CRefModelo con esas características la devuelve, sino, la crea.
getCriteriaEtapas()	Devuelve la criteria para seleccionar todas las etapas que estén en un determinado ensayo clínico y en un cronograma determinado.

Tabla 8. Descripción de las clases del diseño: CCronograma.

Nombre: CCronograma	
Tipo: Clase acceso a datos	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getNuevoId()	Recibe un ensayo clínico y devuelve un nuevo id_cronograma
getReferenciasModeloVisita()	Devuelve un arreglo con todas los Objetos referencias modelo visita Asintomaticas pertenecientes al cronograma.
getEtapas()	Devuelve todas las etapas del Cronograma.
getEtapa()	Recibe un día y un arreglo de etapas y devuelve la etapa que contenga dicho día.
getArregloAsociativoId_NombreEtapa()	Devuelve un arreglo asociativo que su llave es el id_etapa y el valor el nombre de la etapa.

getArregloAsociativoRMV()	Recorre las ref_modelo_visita asintomáticas y devuelve un arreglo asociativo [dia][id_modelo]=true.
getArregRefModeloVisita()	Devuelve todas las ref_modelo_visita que estén programadas para un día y que tengan un comportamiento determinado. Si $comportamientoIgual tiene valor true devuelve las que tengan comportamiento igual al parámetro $comportamiento, sino, las que su comportamiento sea distinto.
getArregRefModeloVisitaSintomatico()	Devuelve las ref_modelo_visita sintomáticas. Que puedan hacerse un día determinado.
getInicioEtapa()	Devuelve el primer día que no está en ninguna etapa, a partir de 0. Si se le pasa un día busca cual es el primer día anterior a el que no esté en ninguna etapa. Si dia tiene valor NULL devuelve el primer día que no esté asignado a ninguna etapa.
getInicioEtapa2()	Debe pasársele un día, y busca cual es el primer día anterior a él que no esté en ninguna etapa
getFinEtapa()	Recibe un día determinado y devuelve el último día en que pudiera terminar la etapa.
getArregloDias()	Devuelve un arreglo Asociativo de días, pasándole el día de inicio y día de fin Si se le pasa NULL como día de inicio y fin, devuelve todos los días del Cronograma
getArregAsocRefModelo_Sintomaticas()	Devuelve un arreglo asociativo de las ref_modelo Sintomaticas, [id_modelo][id_ref_modelo]= tiempo_llenado
getArregAsocDiasIntervalosSintomaticos()	Devuelve un arreglo Asociativo con todos los

	días de inicio y fin de cada período sintomático [dia_inicial]= true, [dia_final]= true
getDiasSintomaticos()	Devuelve un arreglo Asociativo([dia][id_modelo] = id_ref_mod_visita) con los días en que un modelo se comporta de forma Sintomática.
agregarModelosPorDefecto()	Agrega los modelos por defecto, o sea aquellos, que no se planifican en el cronograma pero tienen que ser completados por todos los pacientes incluidos en el ensayo.
guardarModeloPeriodo()	Recibe el id_modelo, el tiempo de llenado, la frecuencia con que se desea planificar el modelo, y el tiempo durante el cual se desea planificar el mismo. Guarda las planificaciones de dicho modelo durante el tiempo especificado a partir del día definido. Ver más en epígrafe 2.4.1
guardarCRefModeloVisitaAsintomatica()	Guarda una CRefModeloVisita asintomática. Si no existe una crefModelo con comportamiento, tiempo de llenado, afecta_cronograma, e id_modelo igual a la CRefModeloVisita que se está creando se crea una nueva.
guardarRefModeloVisitaSintomatica()	Guarda una CRefModeloVisita sintomática. Si no existe una crefModelo con comportamiento, tiempo de llenado, afecta_cronograma, e id_modelo igual a la CRefModeloVisita que se está creando se crea una nueva.
verificarModeloProgramadoParaPeriodo()	Verifica si en ninguno de los días que se quiere programar un modelo asintomático en

	un período , no está ya registrado el mismo, o sea, para que no se programe el mismo modelo más de una vez el mismo día. Devuelve true si el modelo no está planificado en ninguno de los días que se desea.

Tabla 9. Descripción de las clases del diseño: CEtapa.

Nombre: CEtapa	
Tipo: Clase acceso a datos.	
Atributos	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
existeEtapa()	Recibe como parámetro id_etapa, id_cronograma, id_ensayo_clinico, comprueba si lo que se le pasa esta en CCronogramaPeer devolviendo true (si existe) o false (no existe).
getNuevoId()	Recibe un ensayo clínico y devuelve un nuevo id_cronograma.

Tabla 10. Descripción de las clases del diseño: CEtapaPeer.

Nombre: CEtapaPeer	
Tipo: Clase acceso a datos.	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getPaginador()	Devuelve un paginador para etapa.

Tabla 11. Descripción de las clases del diseño: CRefModelo.

Nombre: CRefModelo	
Tipo: Clase acceso a datos.	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getNuevoId()	Recibe un ensayo clínico y devuelve un nuevo id_cronograma.
existeRefModelo()	Recibe id_modelo, id_ensayo_clinico, tiempo_llenado_ref_modelo, comportamiento, afecta_cronograma comprueba si lo que se le pasa está en CRefModeloPeer devolviendo true (si existe) o false (no existe).
estaReferenciado()	Comprueba a través de un objeto criteria si el modelo esta referenciado o no.

Tabla 12. Descripción de las clases del diseño: CRefModeloVisita.

Nombre: CRefModeloVisita	
Tipo: Clase acceso a datos.	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getNuevoId()	Recibe un ensayo clínico y un cronograma y devuelve un nuevo id_cronograma.
existeRefModeloVisita()	Recibe id_ref_modelo_visita, id_cronograma, id_ensayo_clinico comprueba si lo que se le

	pasa está en CRefModeloVisitaPeer devolviendo true (si existe) o false (no existe).
getCRefModeloReEscrita()	Retorna un objeto CrefModeloPeer.

Tabla 13. Descripción de las clases del diseño: CRefModeloVisitaPeer.

Nombre: CRefModeloVisitaPeer	
Tipo: Clase de acceso a datos.	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getPaginadorRefMV()	Devuelve un paginador para referencia modelo visita.

2.2.3 Diagrama de clases Web del diseño.

Los diagramas de clases muestran un conjunto de clases, interfaces, colaboraciones y sus relaciones, proporcionando una perspectiva estática del sistema. Los diagramas de clases para páginas Web difieren un poco del resto de las aplicaciones debido a que estas modelan los aspectos del lado del servidor como una clase y los aspectos del lado del cliente como otra; las extensiones que se presentan para este tipo de aplicaciones son estereotipos, valor etiquetado, restricciones.

Gestionar_Cronograma.

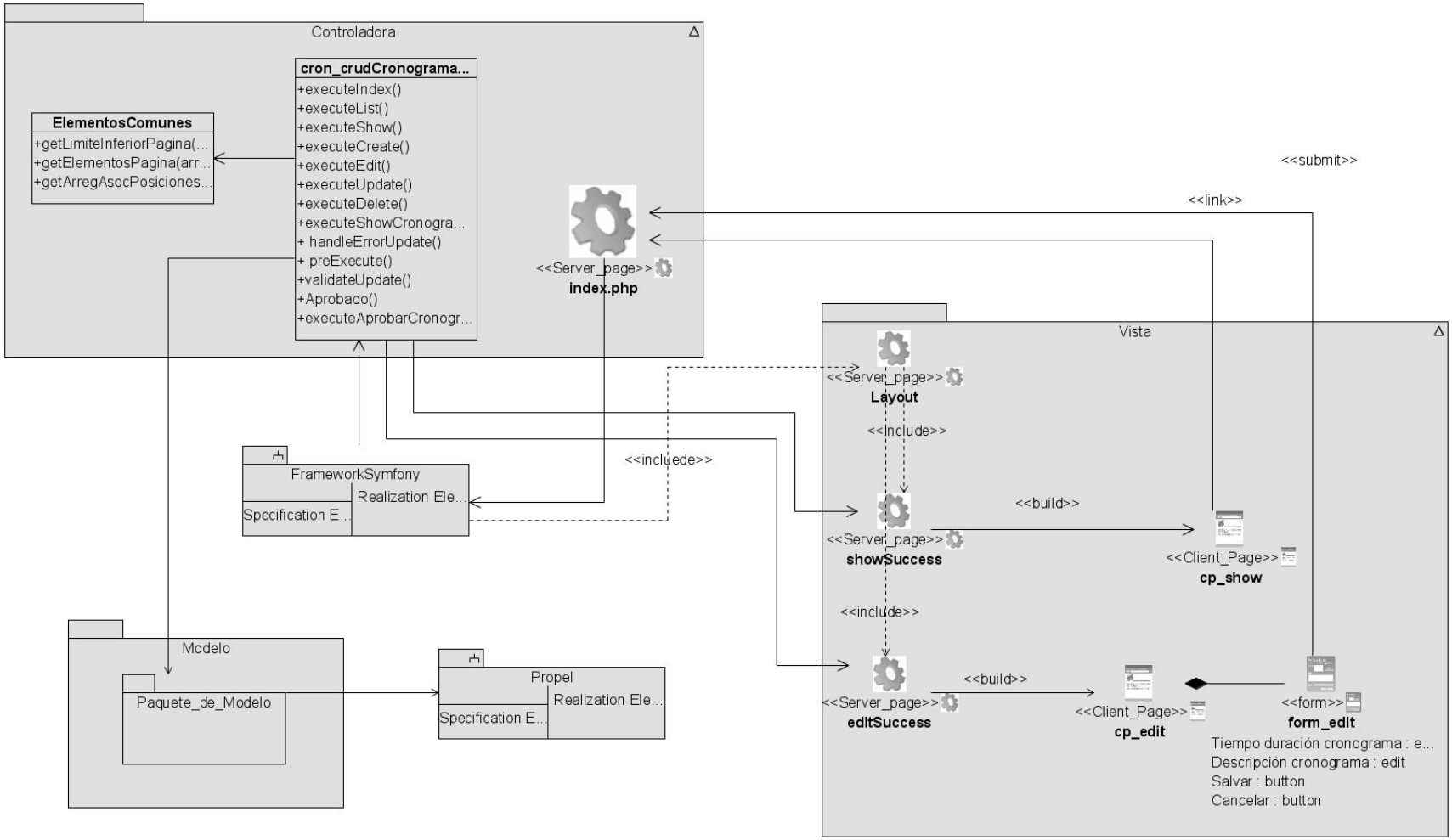

Figura 4 Diagrama de clases del diseño CU_Gestionar_Cronograma.

Mostrar_Cronograma_General.

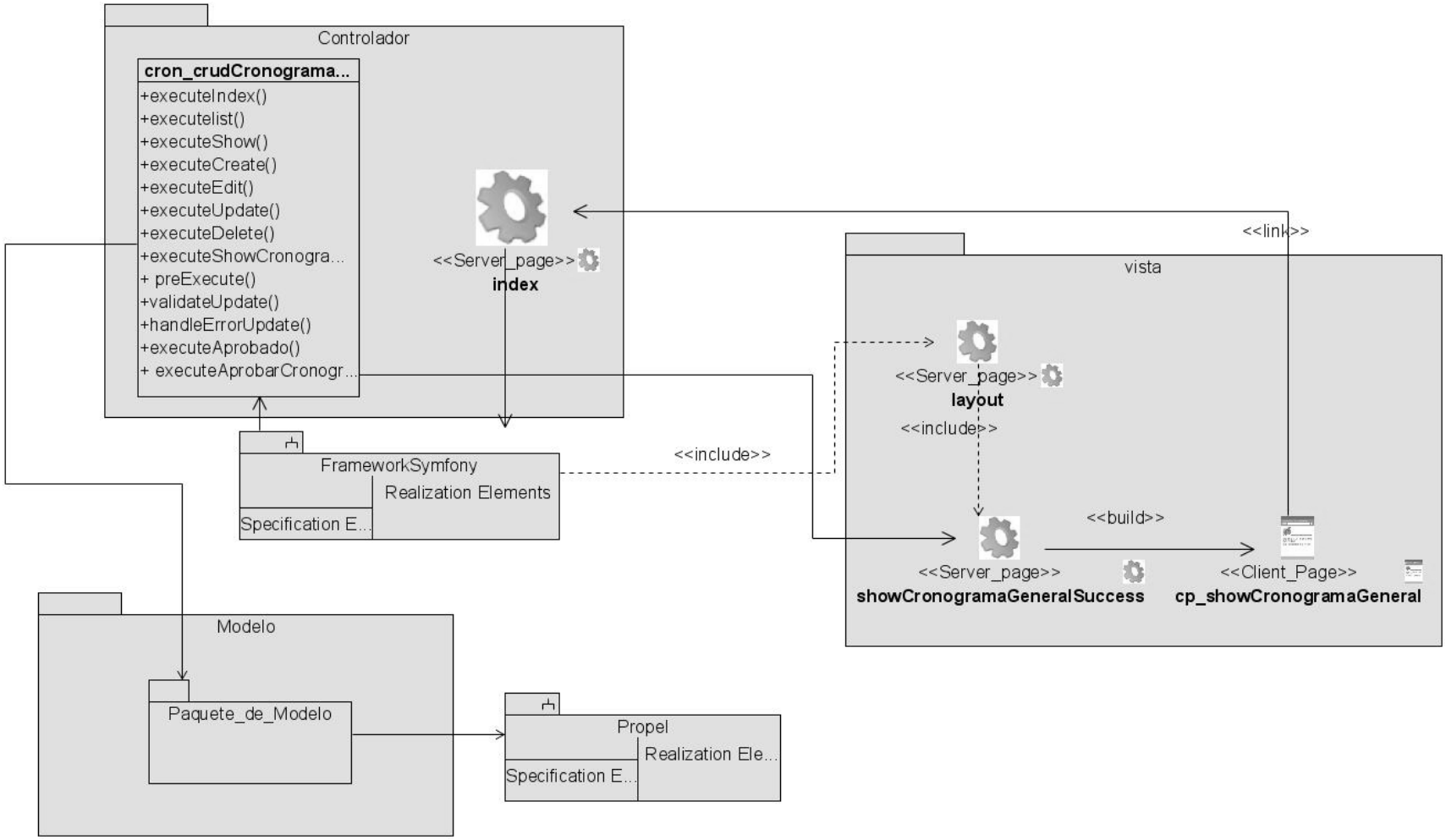

Figura 5 Diagrama de clases del diseño CU_Mostrar_Cronograma_General.

Gestionar_Etapa.

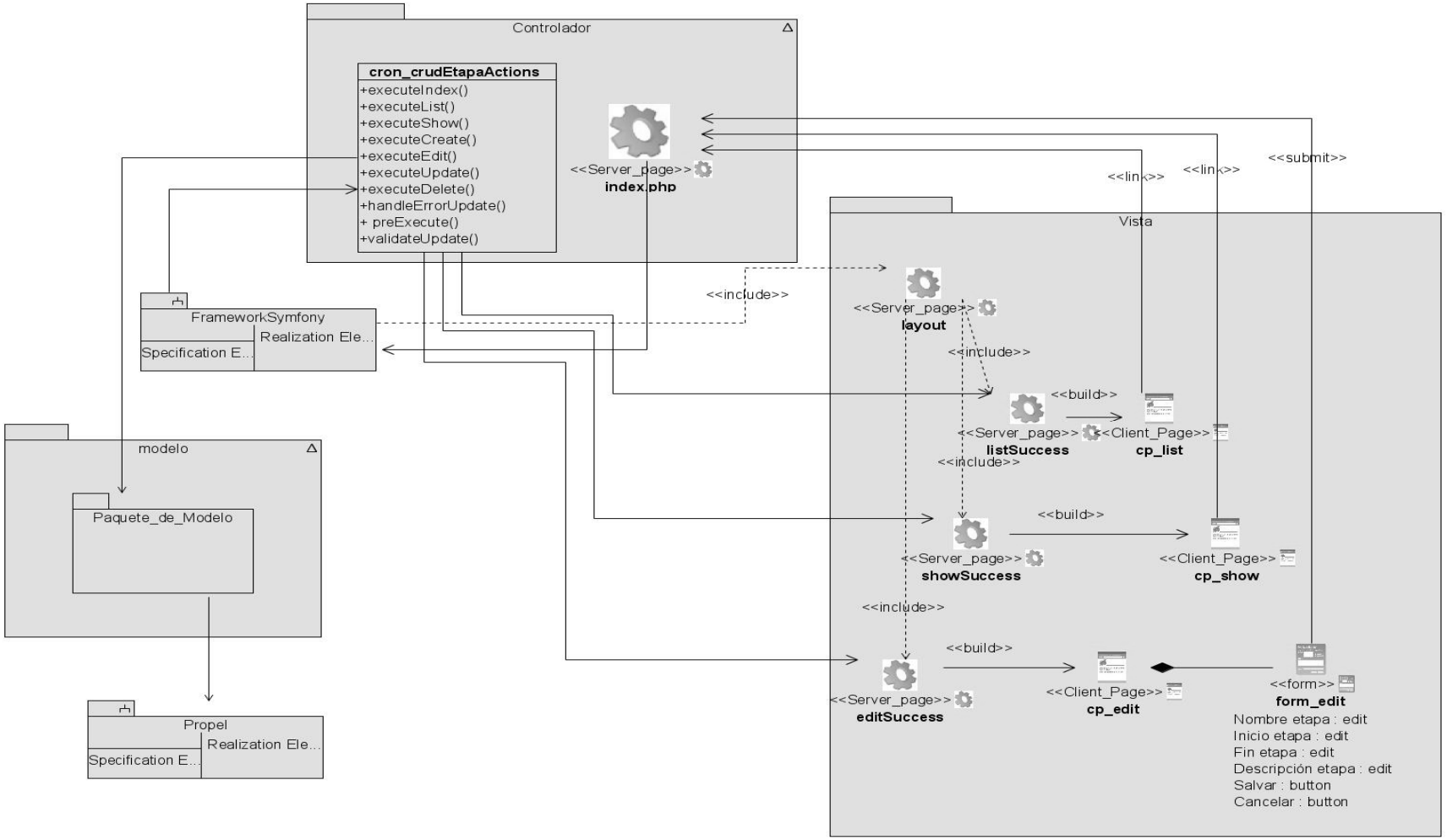

Figura 6 Diagrama de clases del diseño CU_Gestionar_Etapa.

Evento_Periódico.

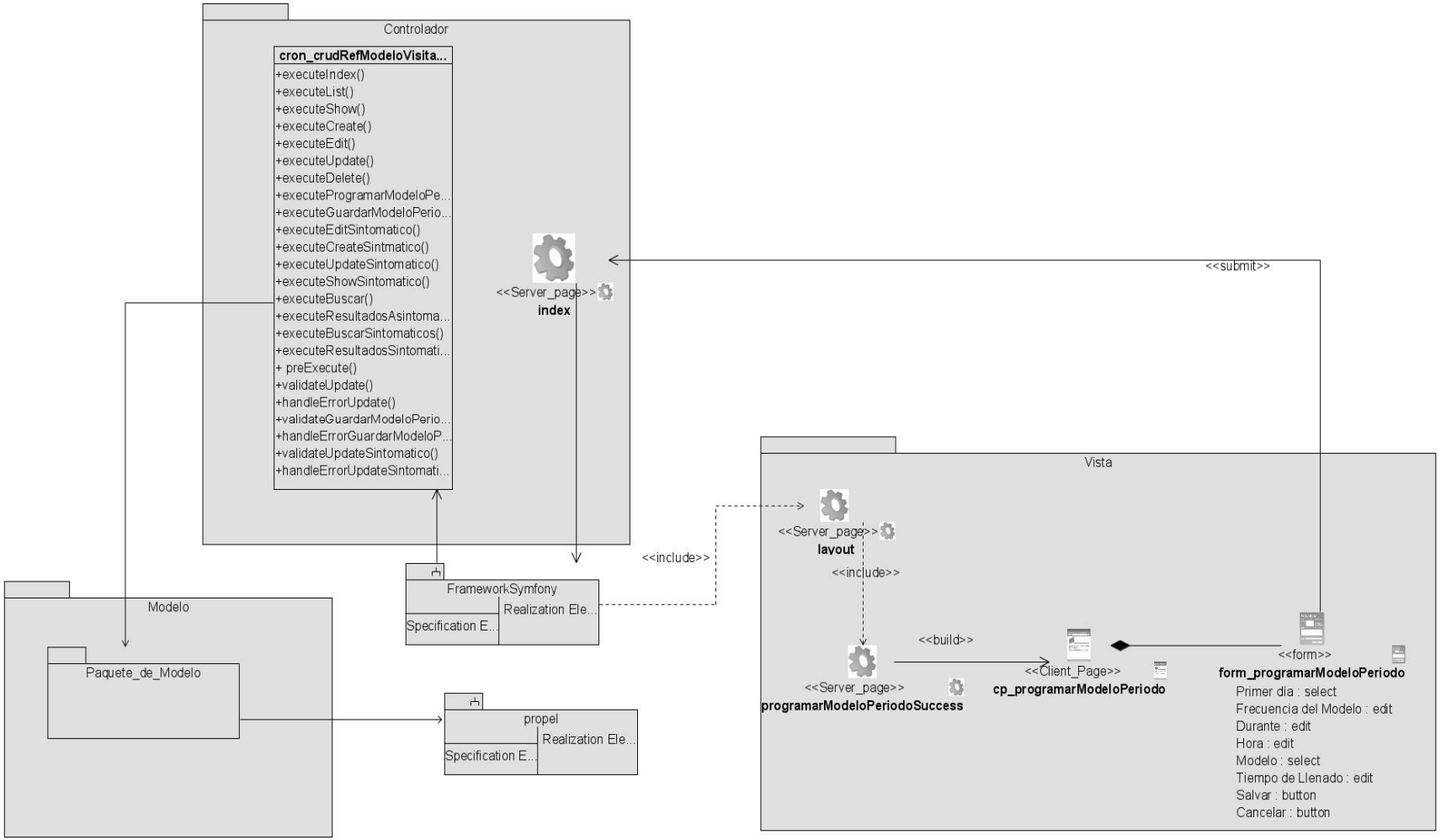

Figura 7 Diagrama de clases del diseño CU_Evento_Periódico.

Gestionar_Modelo_Asintomático.

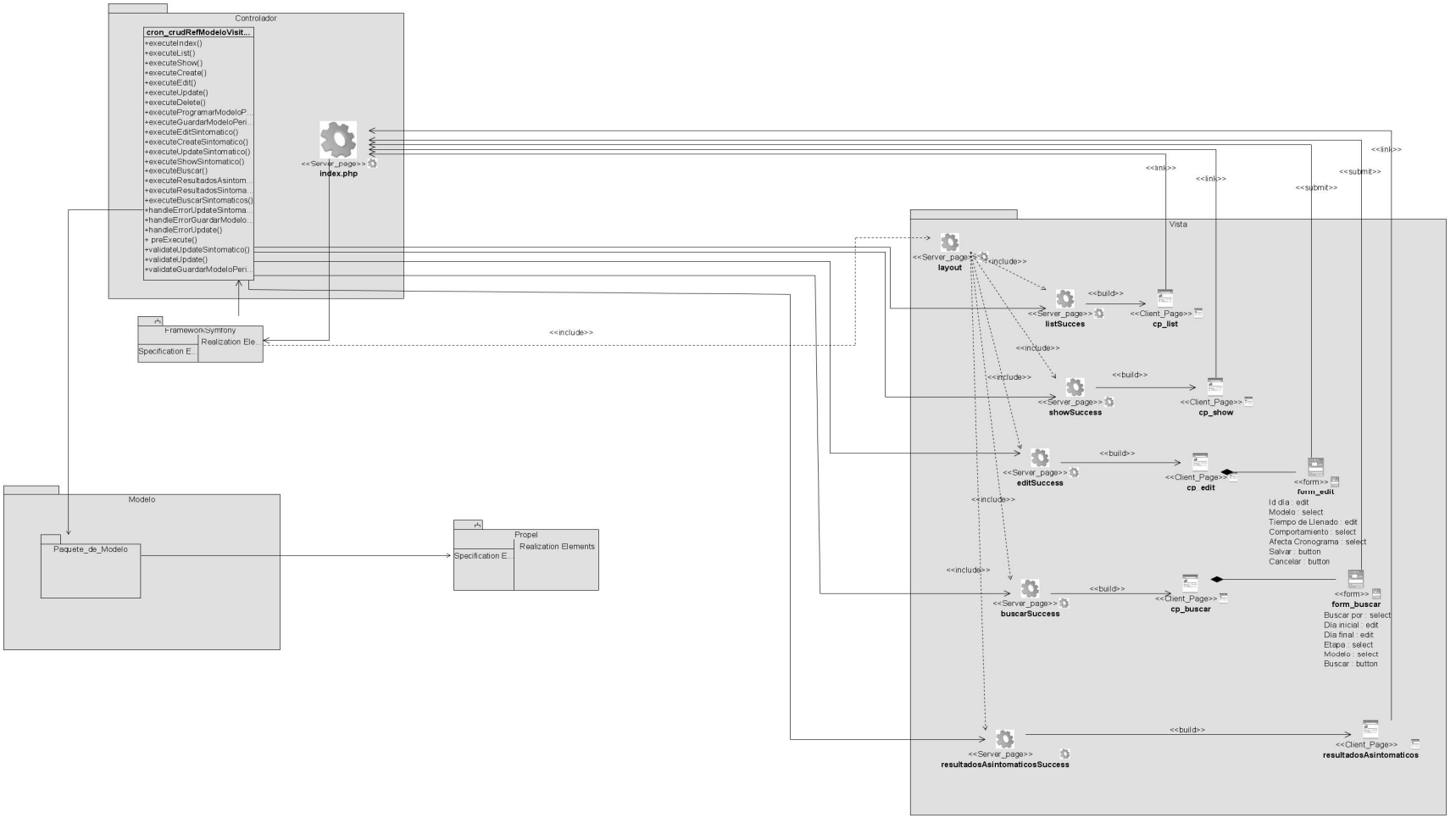

Figura 8 Diagrama de clases del diseño CU_Gestionar_Modelo_Asintomático.

Gestionar_Modelo_Sintomático.

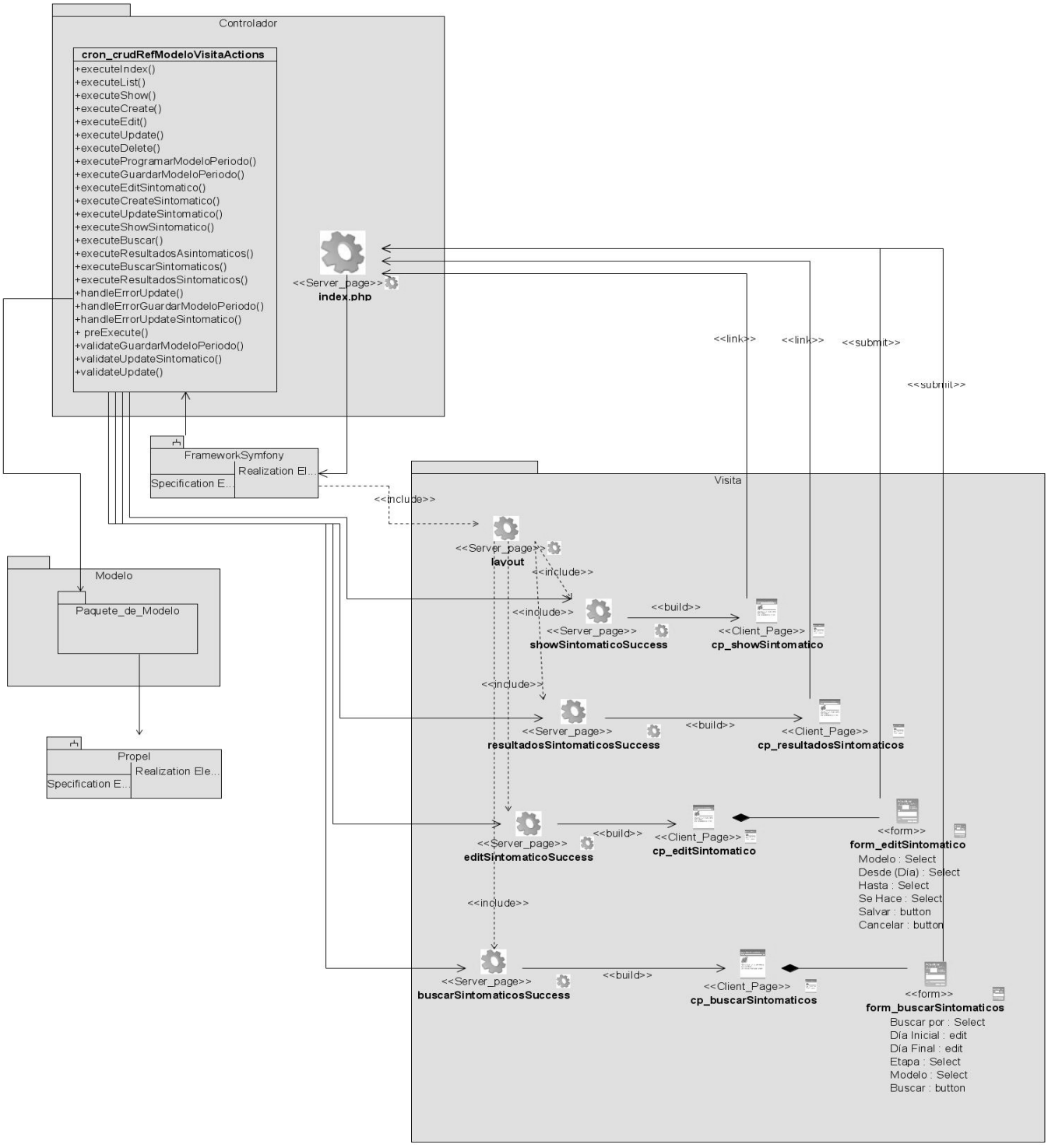

Figura 9 Diagrama de clases del diseño CU_Gestionar_Modelo_Sintomático.

Paquete modelo.

Symfony es un framework orientado objetos, sin embargo las bases de datos siguen una estructura relacional. Por esta razón, para acceder a los datos como si estos tuviesen una estructura orientada a objetos symfony utiliza Propel, el cual es el encargado de realizar el mapeo de objetos a la base de datos (ORM), generando este las clases de acceso a datos.

Este conjunto de clases representan la capa modelo en la arquitectura MVC propuesta por el framework.

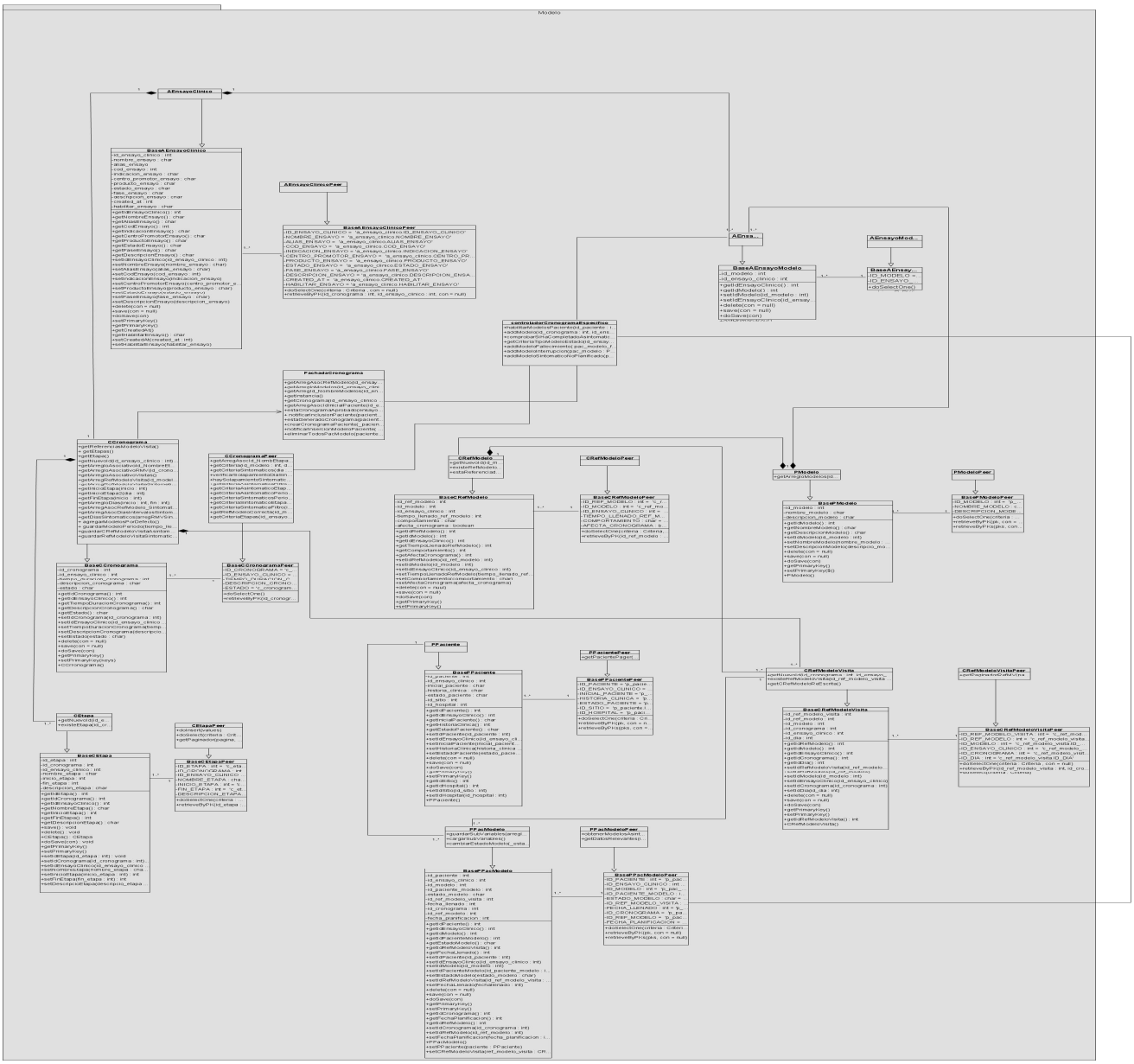

Figura 10 Paquete modelo.

2.2.4 Diagrama de Interacción (Secuencia)

Los Diagramas de Interacción muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas. Existen dos tipos de diagramas de interacción: Diagramas de secuencia, Diagramas de colaboración.

En la presente investigación se realizan los diagramas de secuencia estos muestra interacciones de objetos organizadas en una secuencia de tiempo, estos no incluye relaciones de objetos.

Diagrama de interacción para Gestionar_Cronograma

Gestionar_Cronograma (Crear Cronograma).

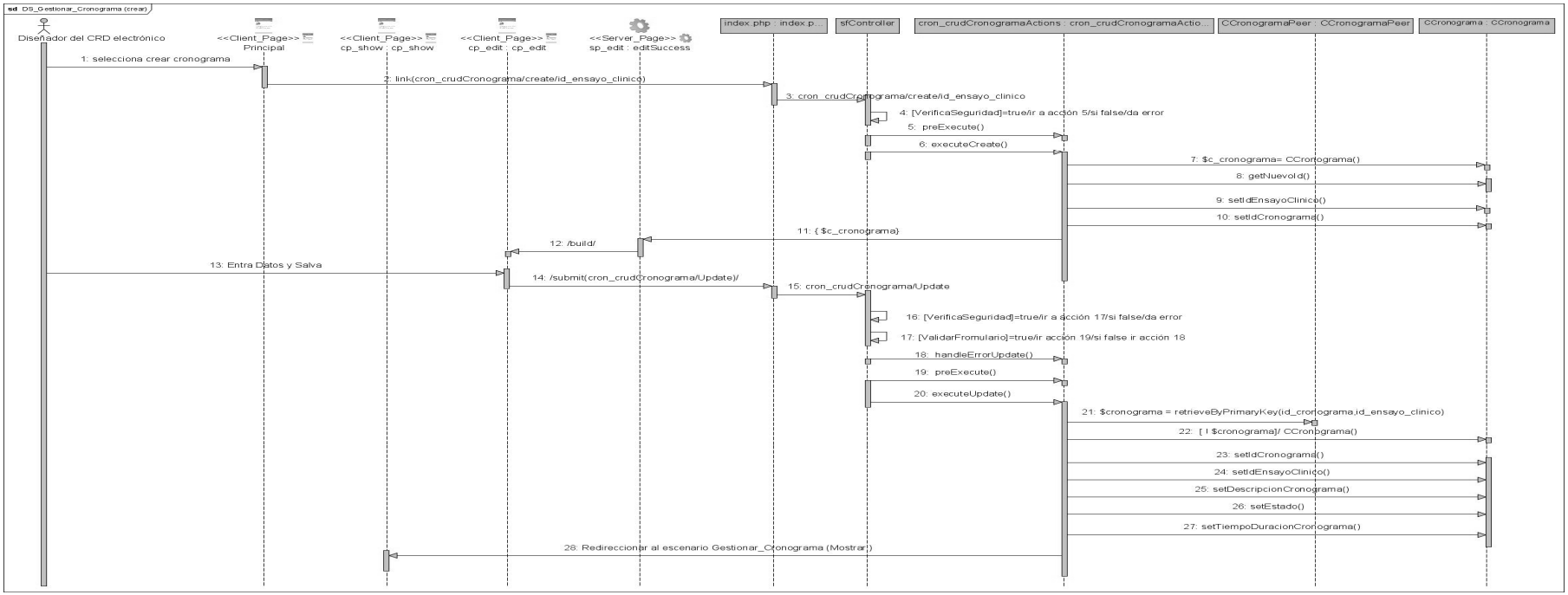

Figura 11 Diagrama de interacción Gestionar_Cronograma (Crear Cronograma).

Gestionar_Cronograma (Editar Cronograma).

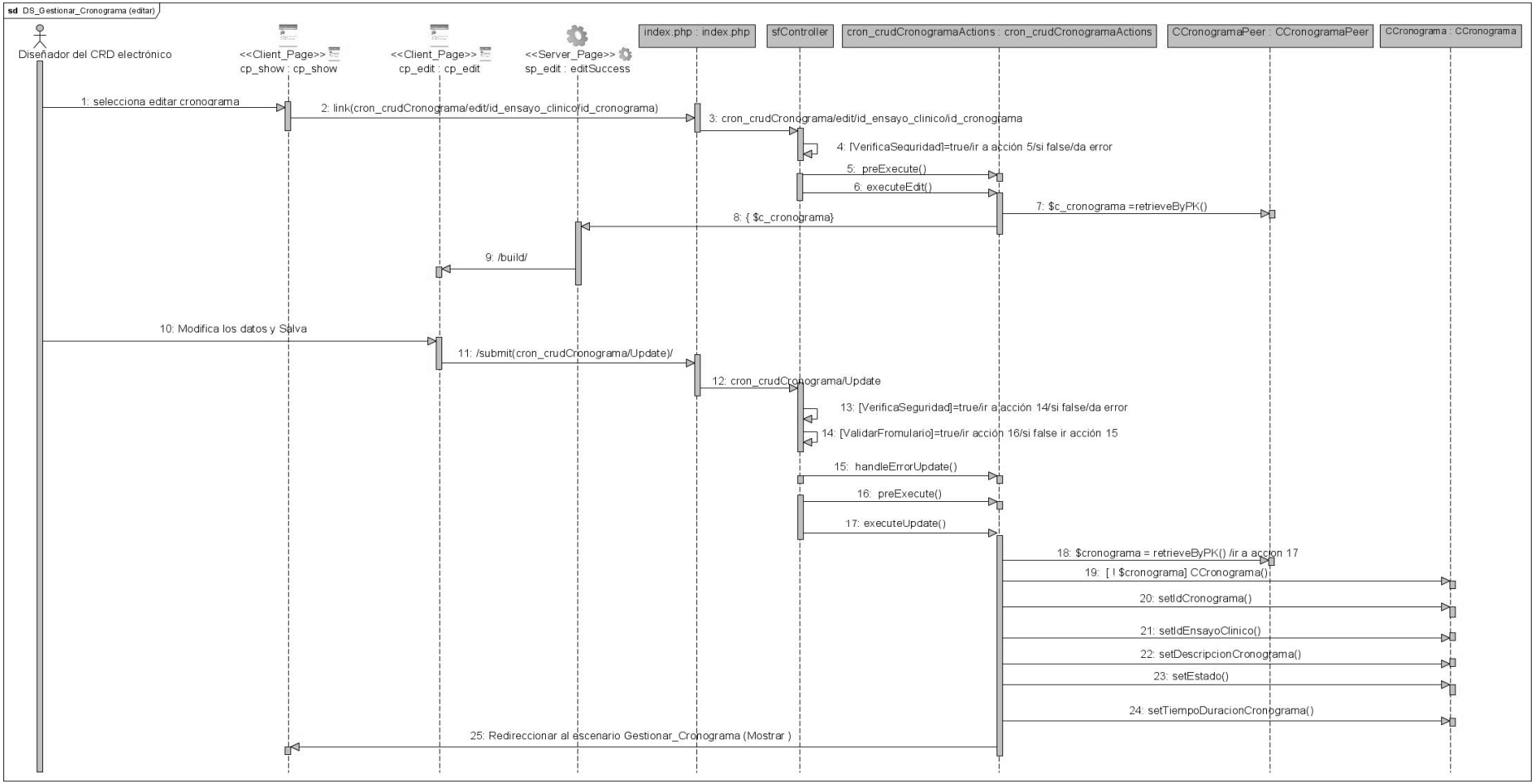

Figura 12 Diagrama de interacción Gestionar_Cronograma (Editar Cronograma).

Gestionar_Cronograma_General (Mostrar Cronograma General).

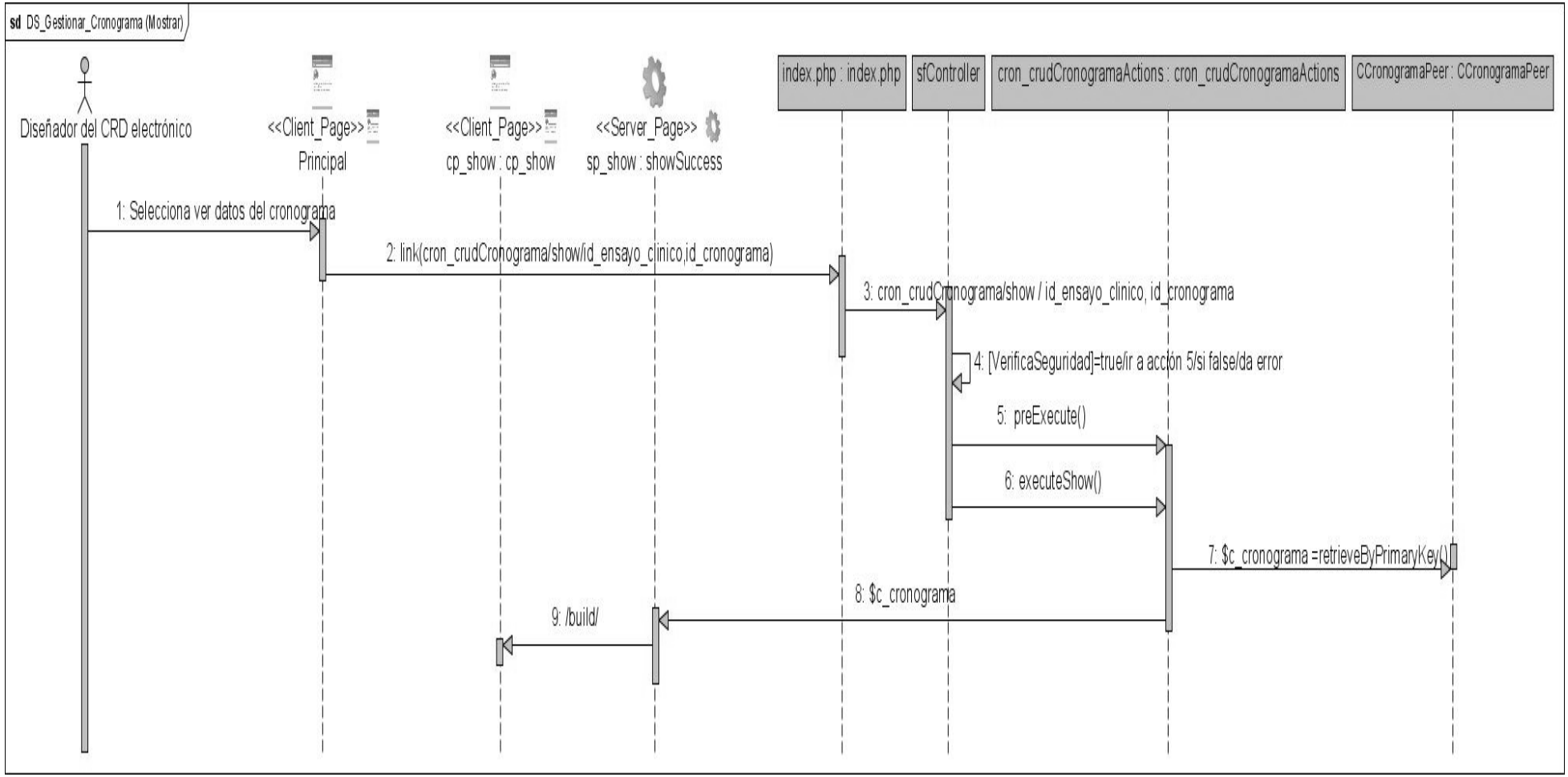

Figura 13 Diagrama de interacción Gestionar_Cronograma (Mostrar Cronograma General).

Diagrama de interacción para Mostrar_Cronograma_General

Mostrar_Cronograma_General.

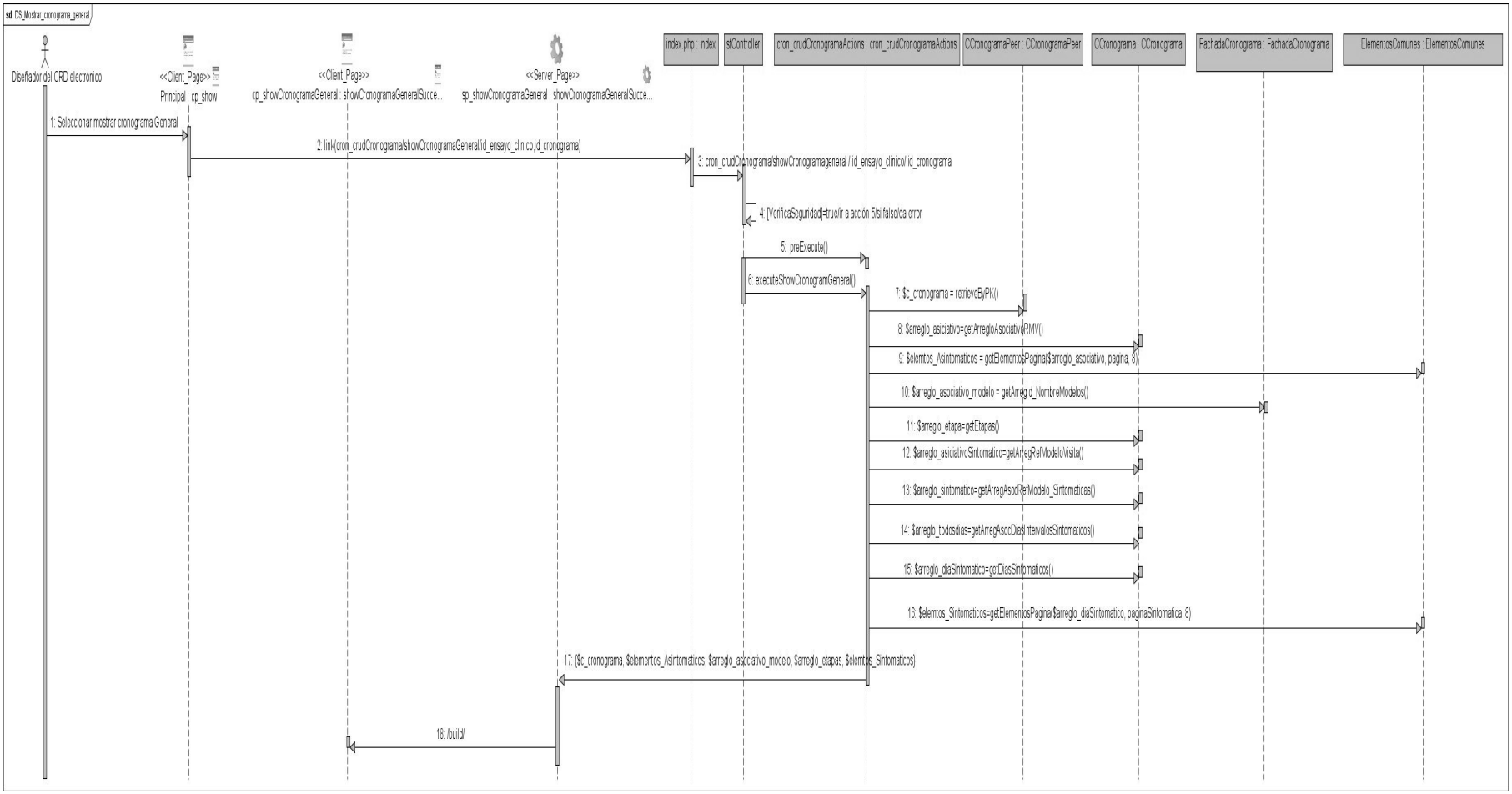

Figura 14 Diagrama de interacción Mostrar_Cronograma_General.

Diagrama de interacción para Gestionar_Etapa

Gestionar_Etapa (Crear Etapa).

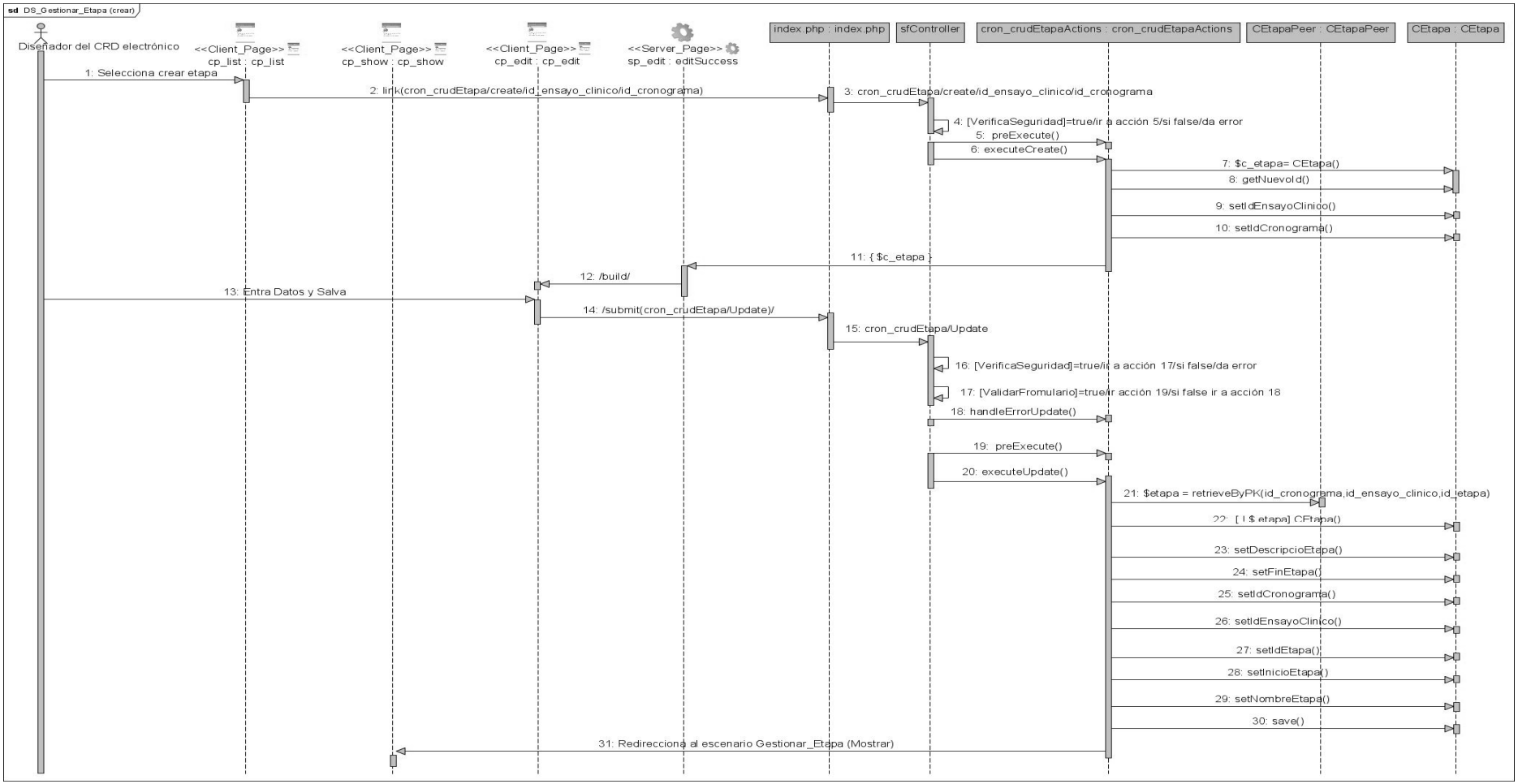

Figura 15 Diagrama de interacción Gestionar_Etapa (Crear Etapa).

Gestionar_Etapa (Editar Etapa).

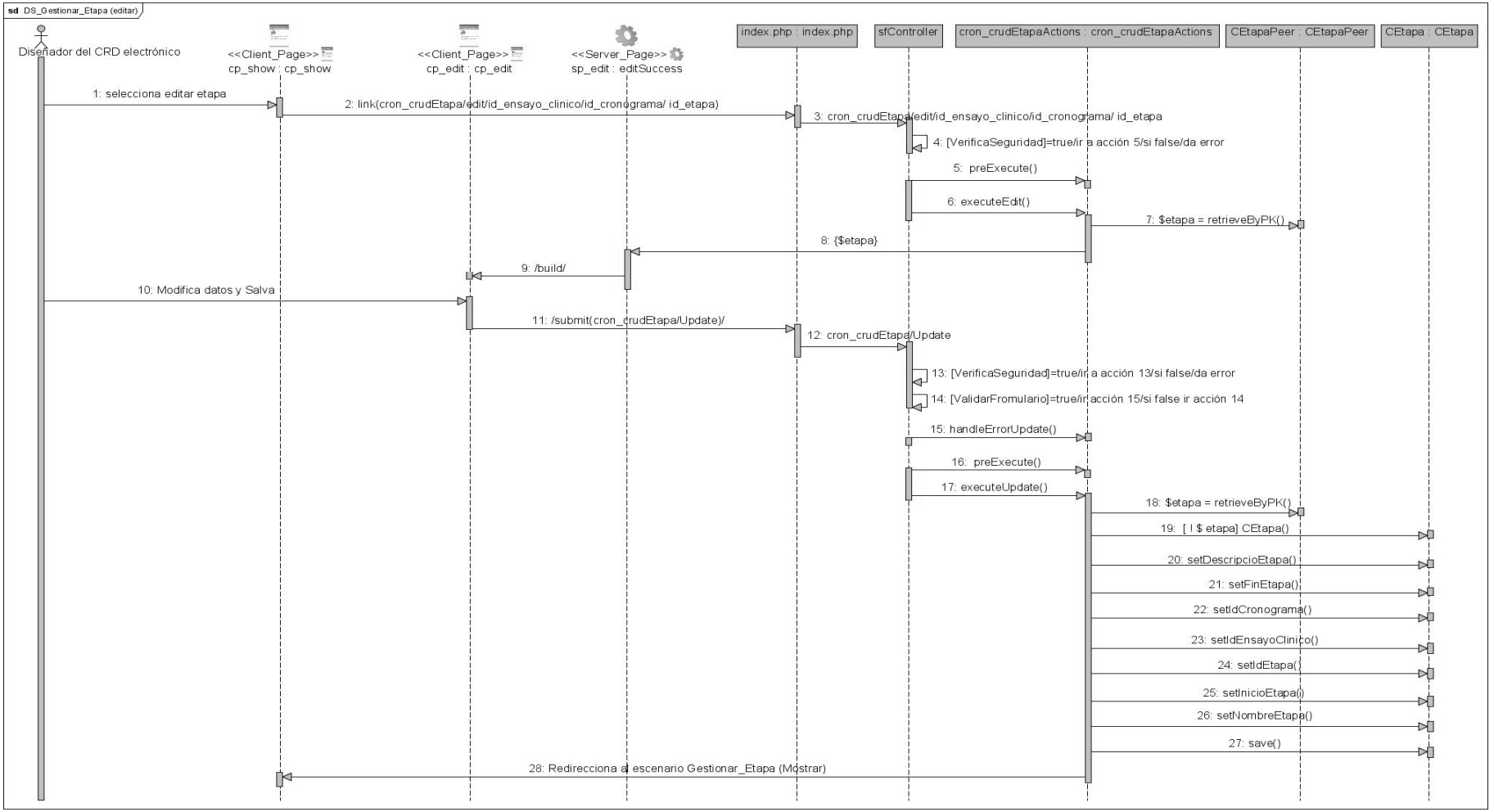

Figura 16 Diagrama de interacción Gestionar_Etapa (Editar Etapa).

Gestionar_Etapa (Mostrar Etapa).

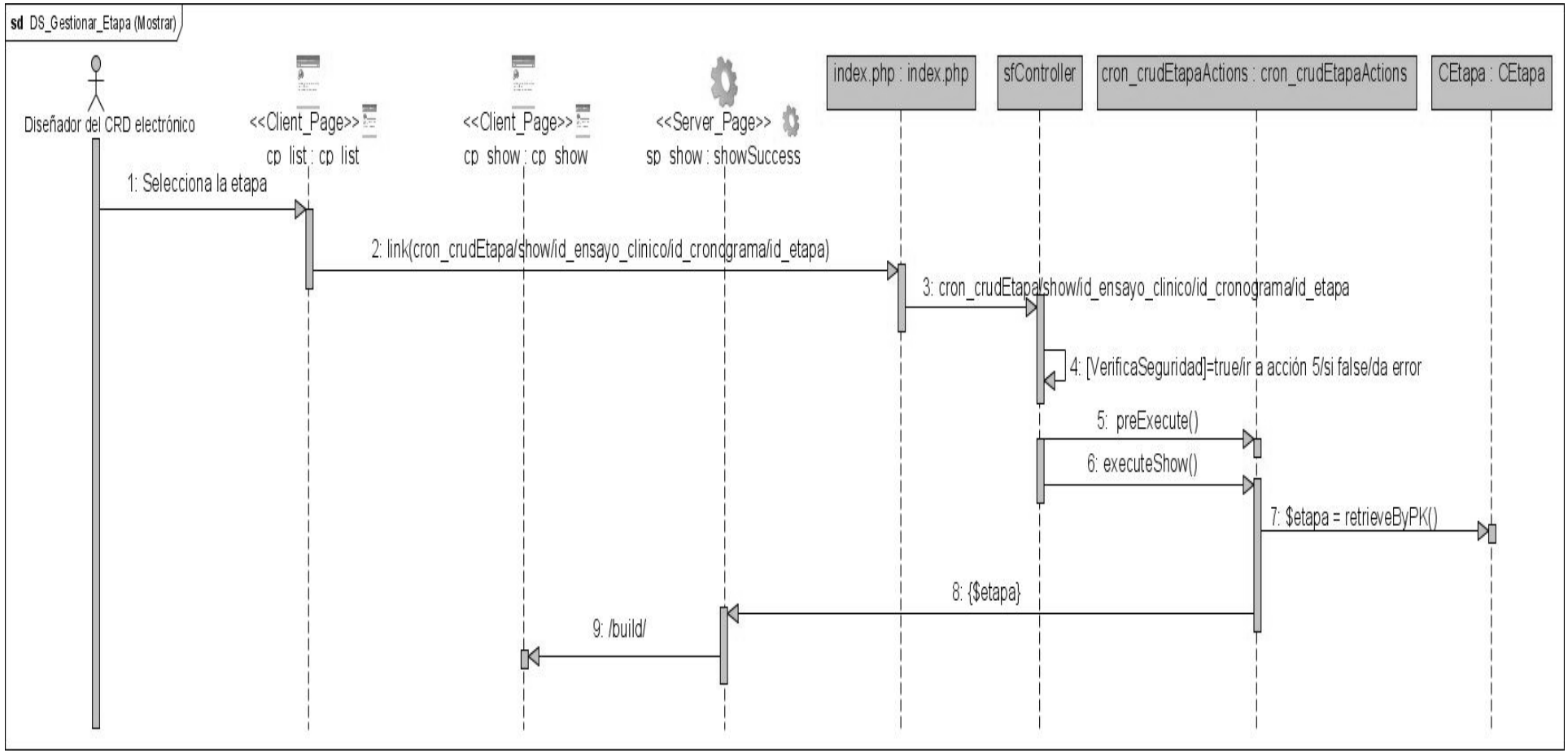

Figura 17 Diagrama de interacción Gestionar_Etapa (Mostrar Etapa).

Gestionar_Etapa (Listar Etapa).

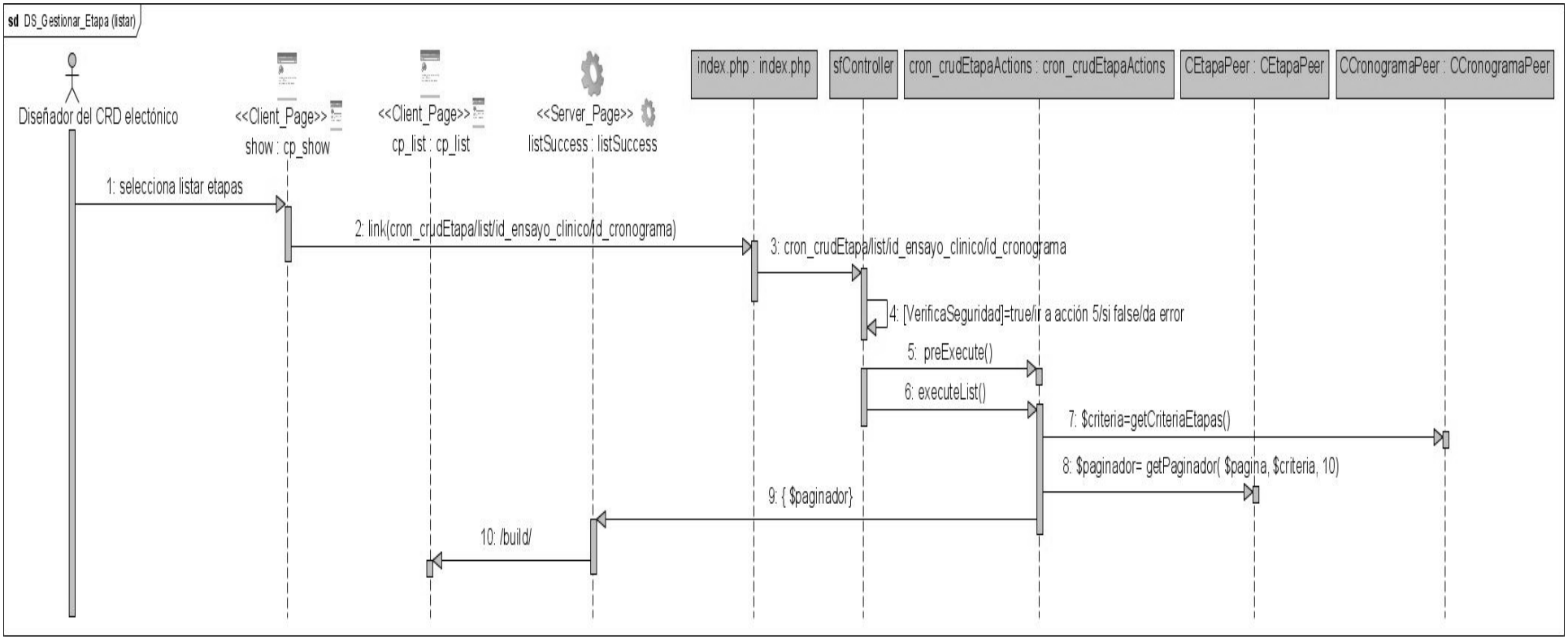

Figura 18 Diagrama de interacción Gestionar_Etapa (Listar Etapa).

Gestionar_Etapa (Eliminar Etapa).

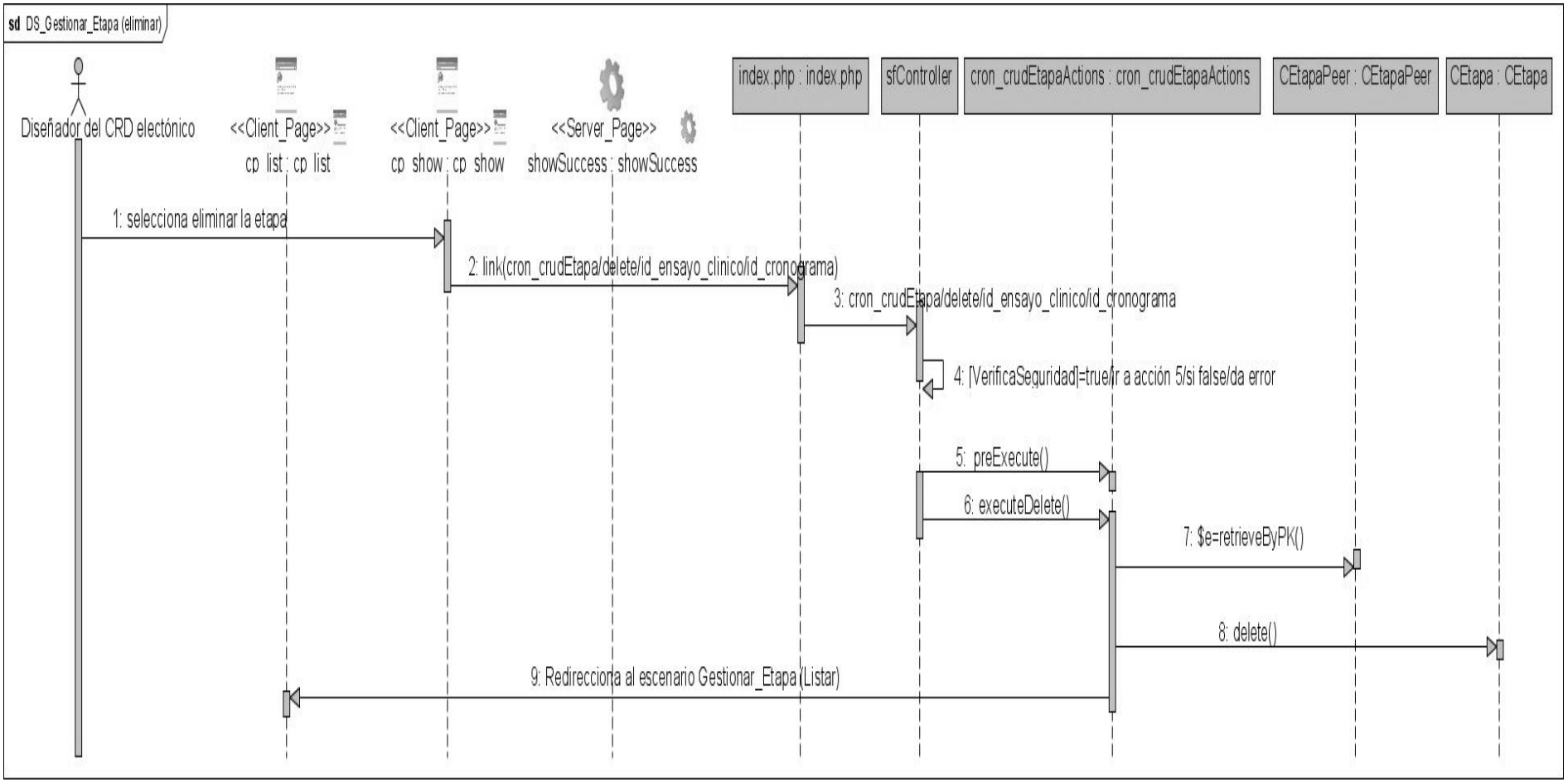

Figura 19 Diagrama de interacción Gestionar_Etapa (Eliminar).

Diagrama de interacción para Gestionar_Modelo_Sintomático

Gestionar_Modelo_Sintomático(Mostrar Modelo Sintomático).

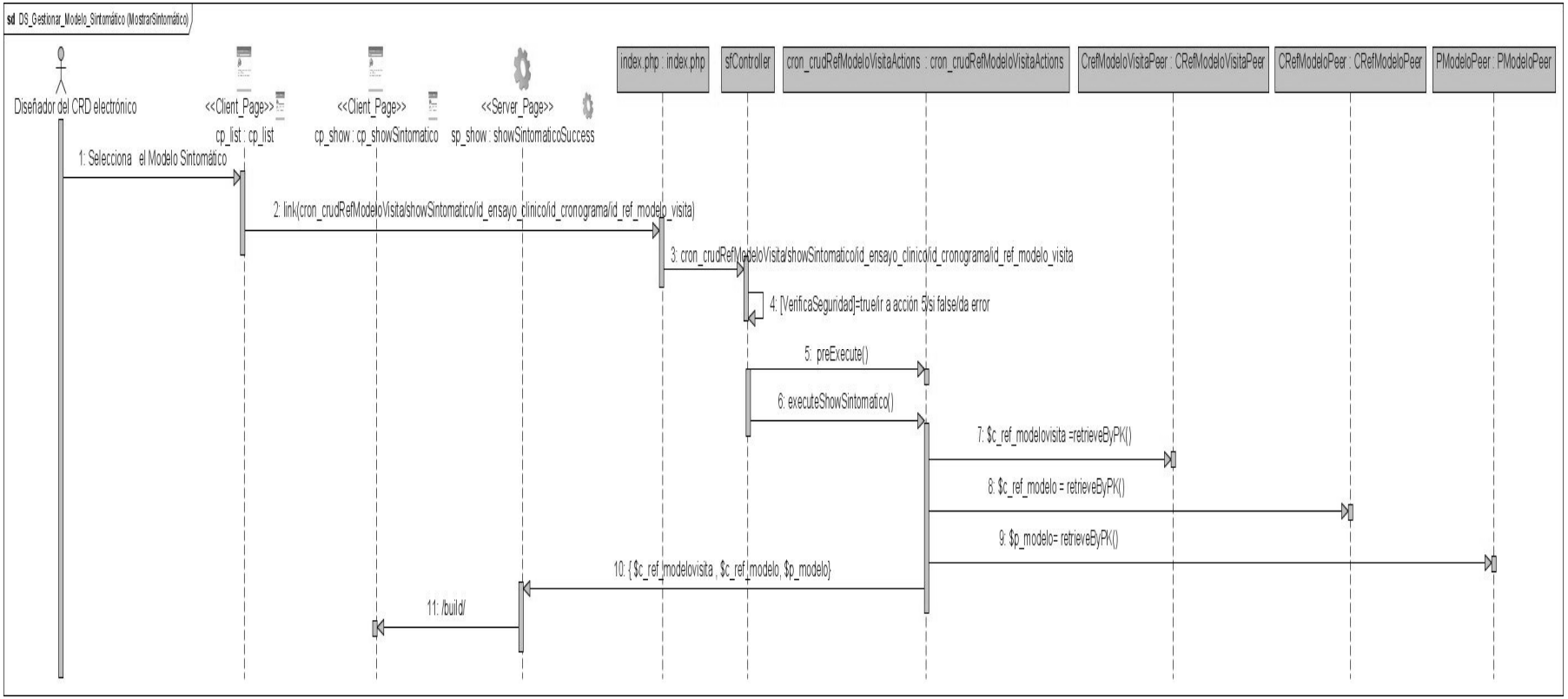

Figura 20 Diagrama de interacción Gestionar_Modelo_Sintomático(Mostrar Modelo Sintomático).

Gestionar_Modelo_Sintomático(Editar Modelo Sintomático).

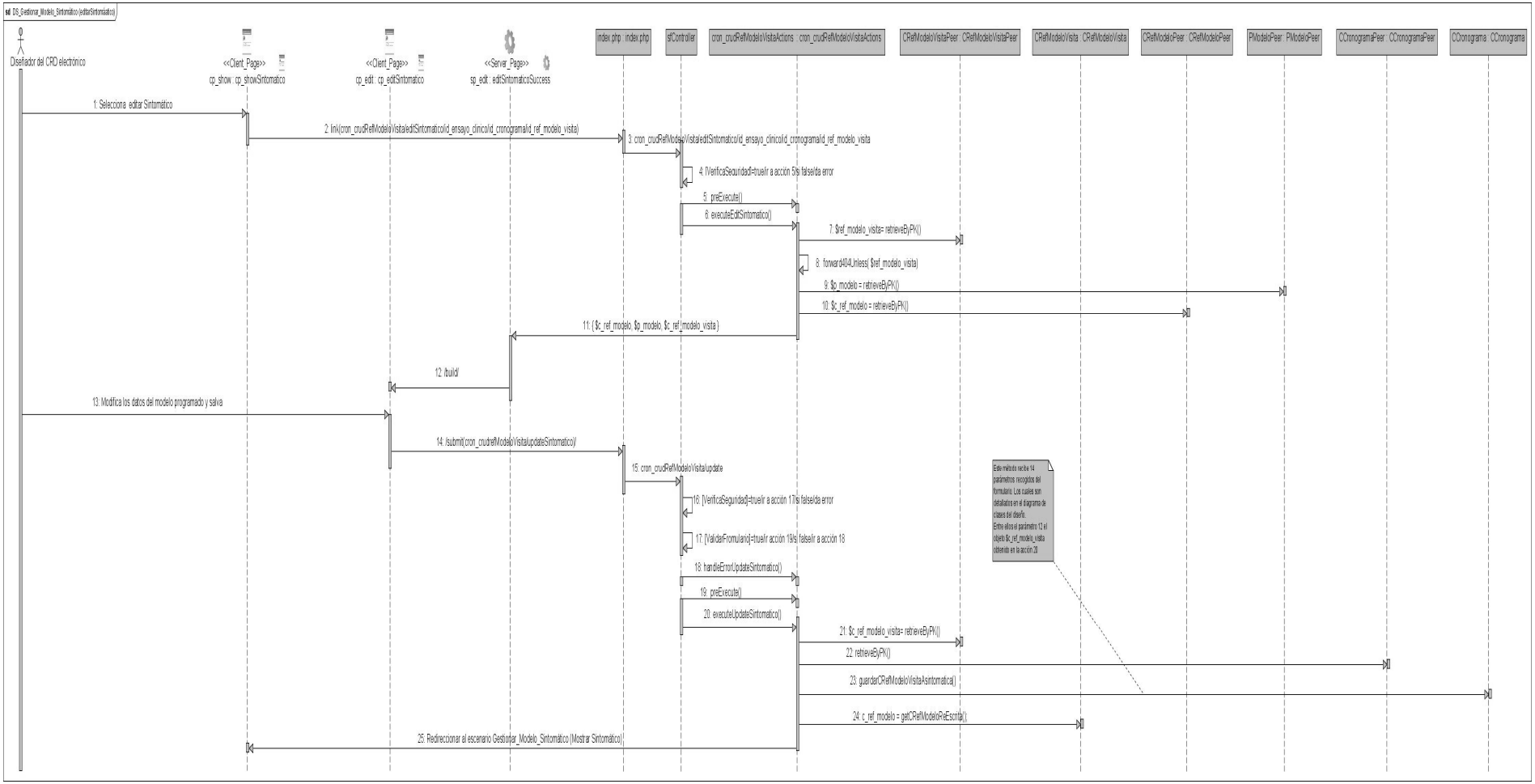

Figura 21 Diagrama de interacción Gestionar_Modelo_Sintomático(Editar Modelo Sintomático).

Gestionar_Modelo_Sintomático(Definir Modelo Sintomático).

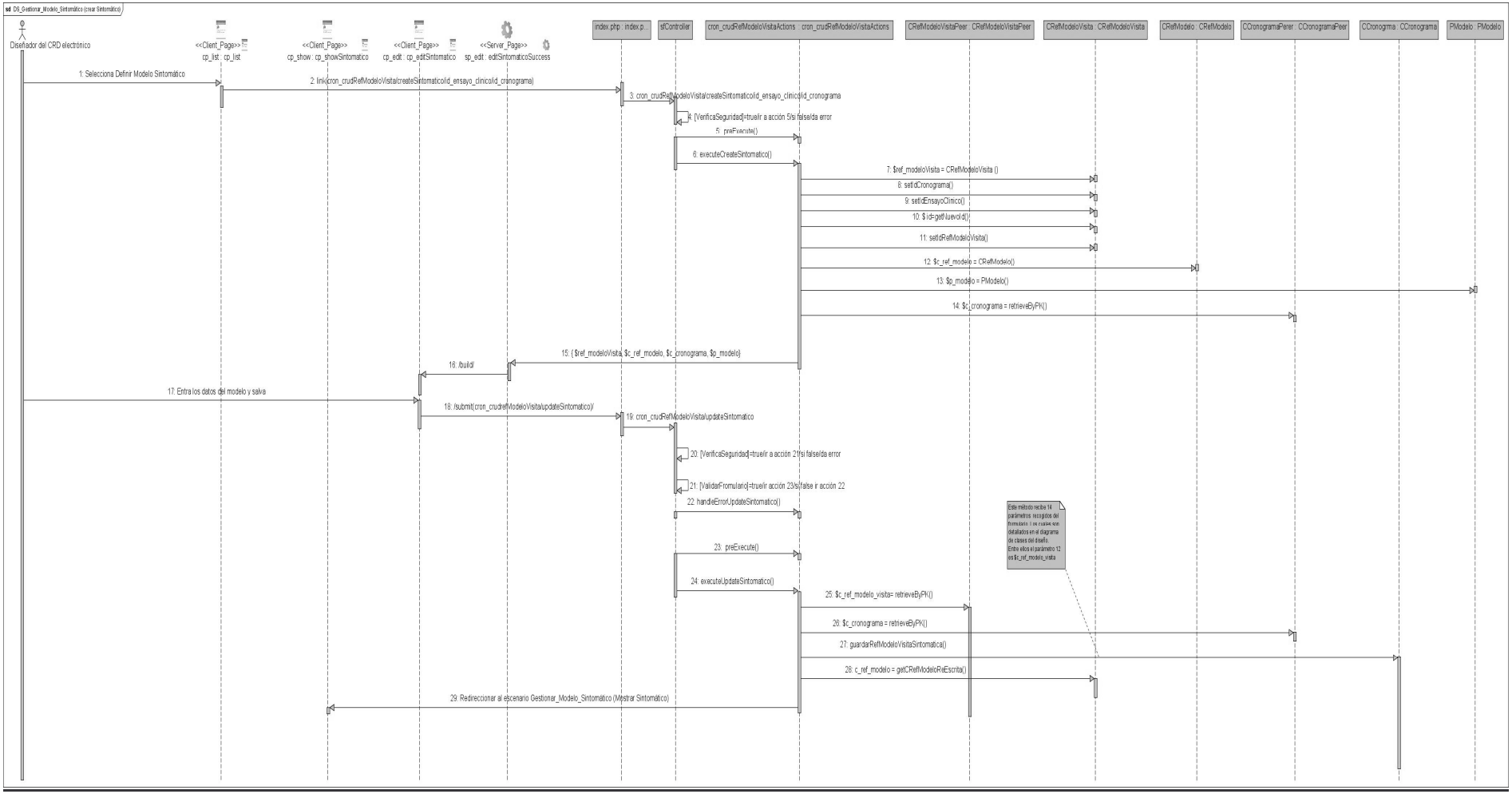

Figura 22 Diagrama de interacción Gestionar_Modelo_Sintomático(Definir Modelo Sintomático).

Gestionar_Modelo_Sintomático(Búsqueda avanzada de Sintomáticos).

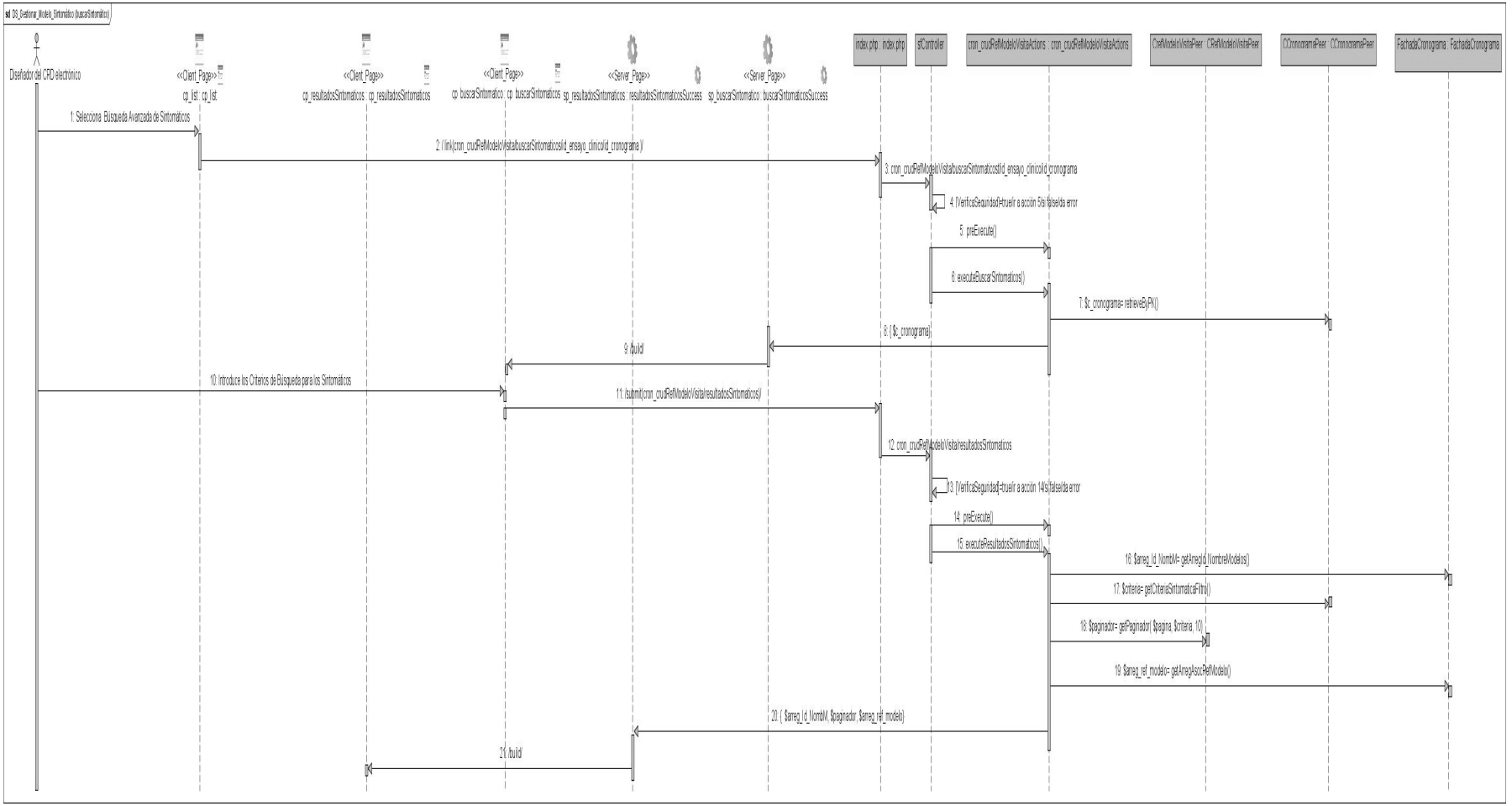

Figura 23 Diagrama de interacción Gestionar_Modelo_Sintomático(Búsqueda avanzada de Sintomáticos).

Diagrama de interacción para Evento Periódico

Evento_Periódico.

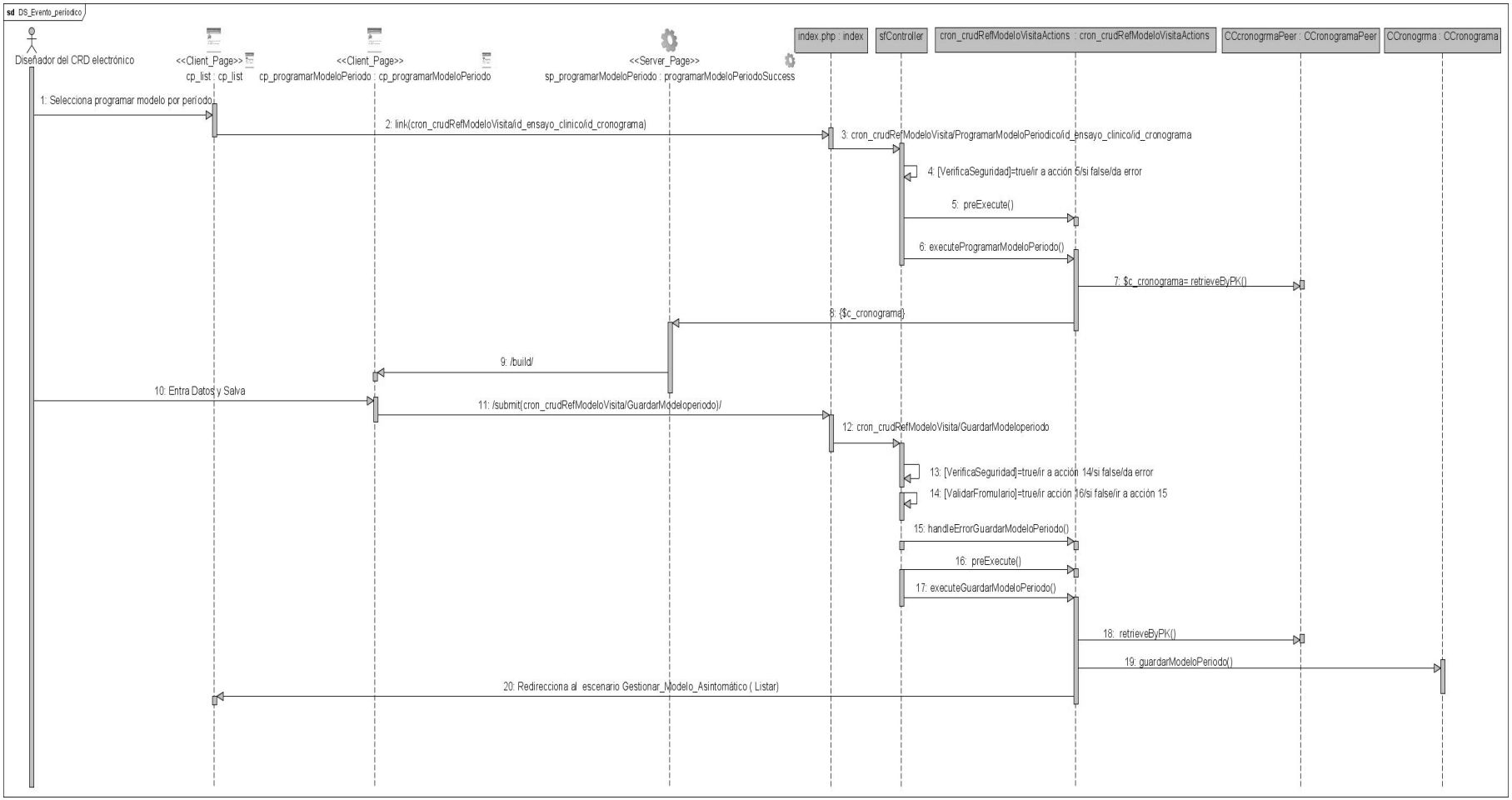

Figura 24 Diagrama de interacción Evento_Periódico.

Diagrama de interacción para Gestionar_Modelo_Asintomático

Gestionar_Modelo_Asintomático (Programar Modelo Asintomático).

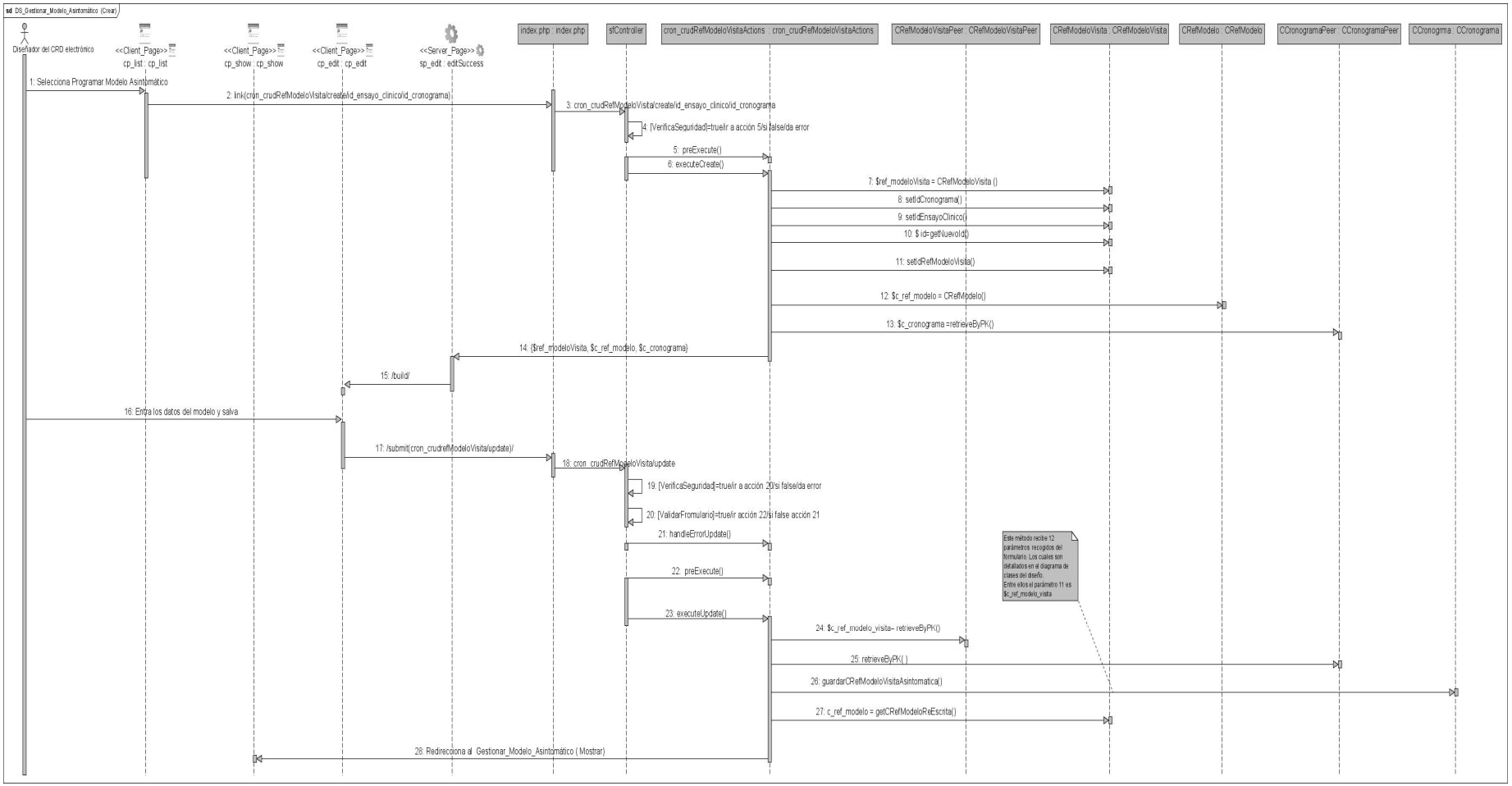

Figura 25 Diagrama de interacción Gestionar_Modelo_Asintomático (Programar Modelo Asintomático).

Gestionar_Modelo_Asintomático (Editar Modelo Asintomático).

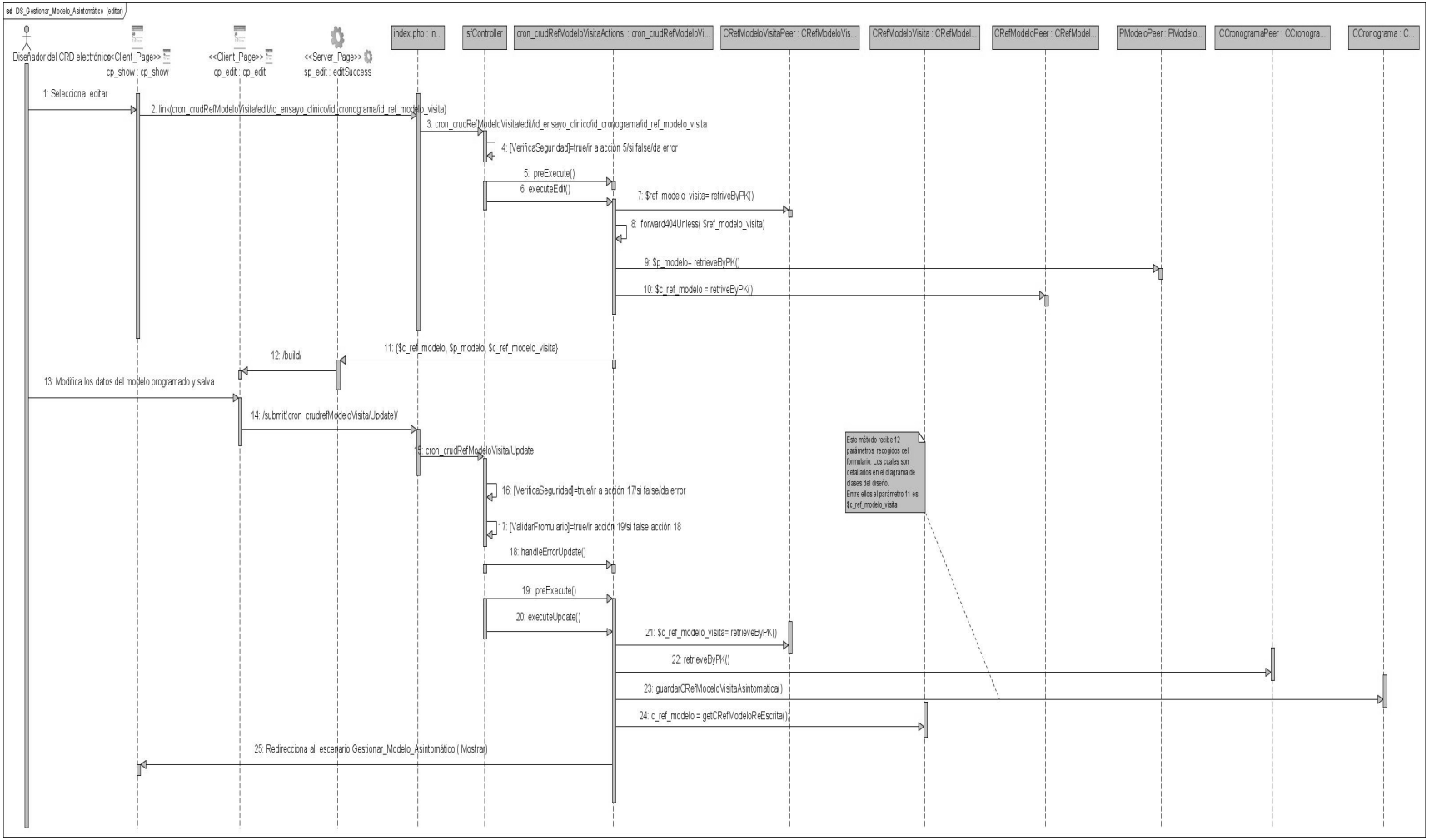

Figura 26 Diagrama de interacción Gestionar_Modelo_Asintomático (Editar Modelo Asintomático).

Gestionar_Modelo_Asintomático (Eliminar Modelo Asintomático).

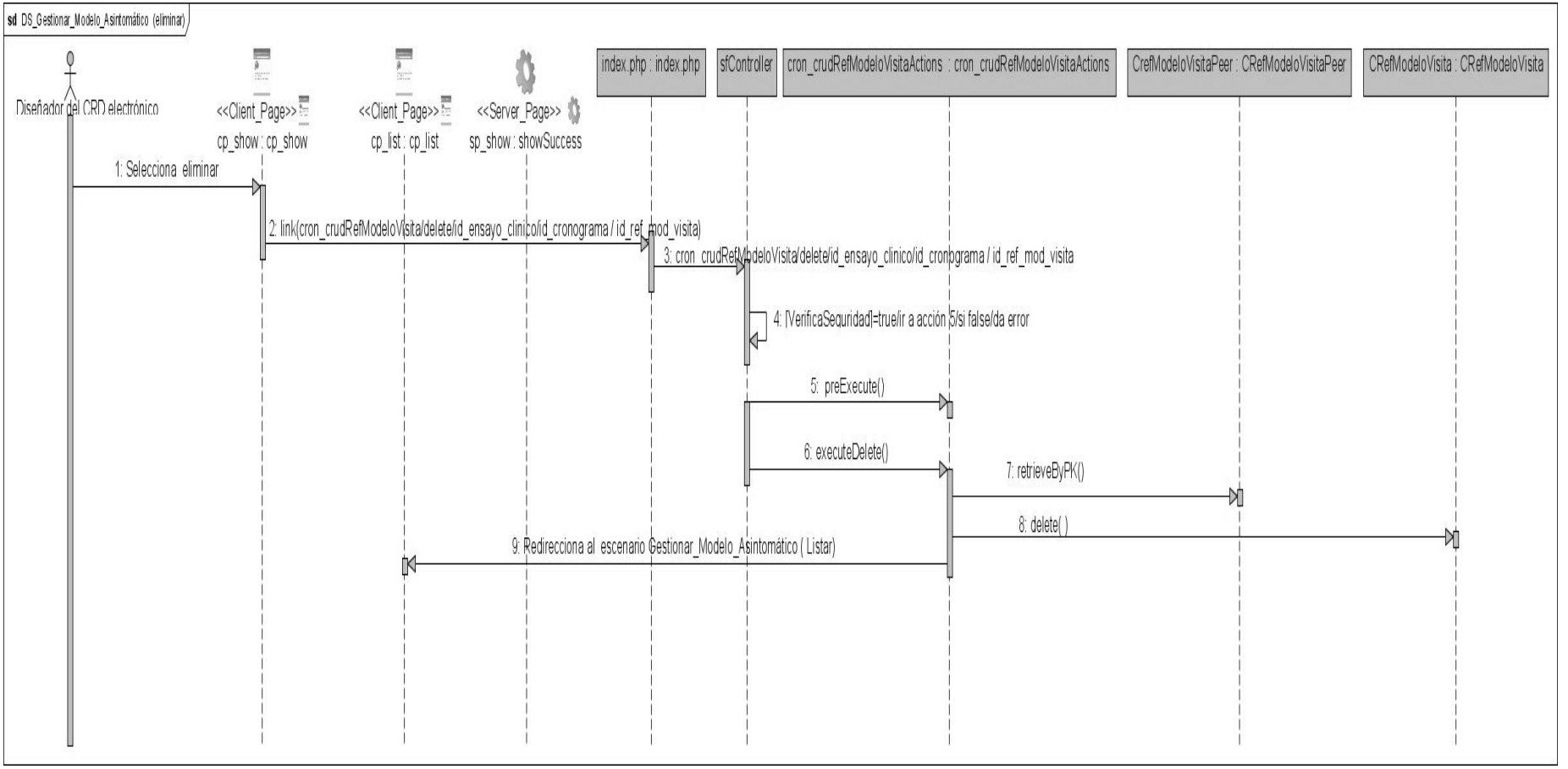

Figura 27 Diagrama de interacción Gestionar_Modelo_Asintomático (Eliminar Modelo Asintomático).

Gestionar_Modelo_Asintomático (Planificación de Modelos).

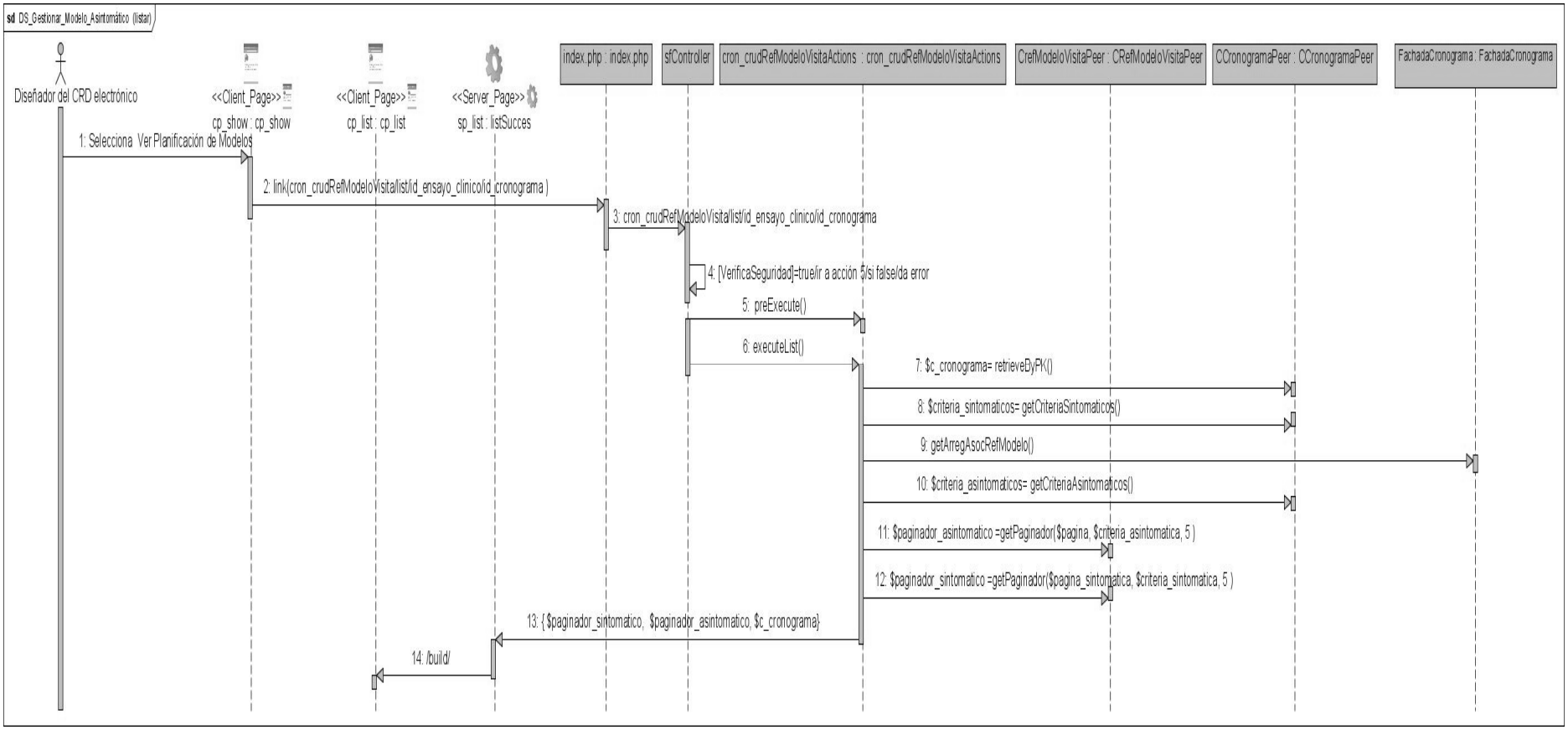

Figura 28 Diagrama de interacción Gestionar_Modelo_Asintomático (Planificación de Modelos).

Gestionar_Modelo_Asintomático (Mostrar Modelo Asintomático).

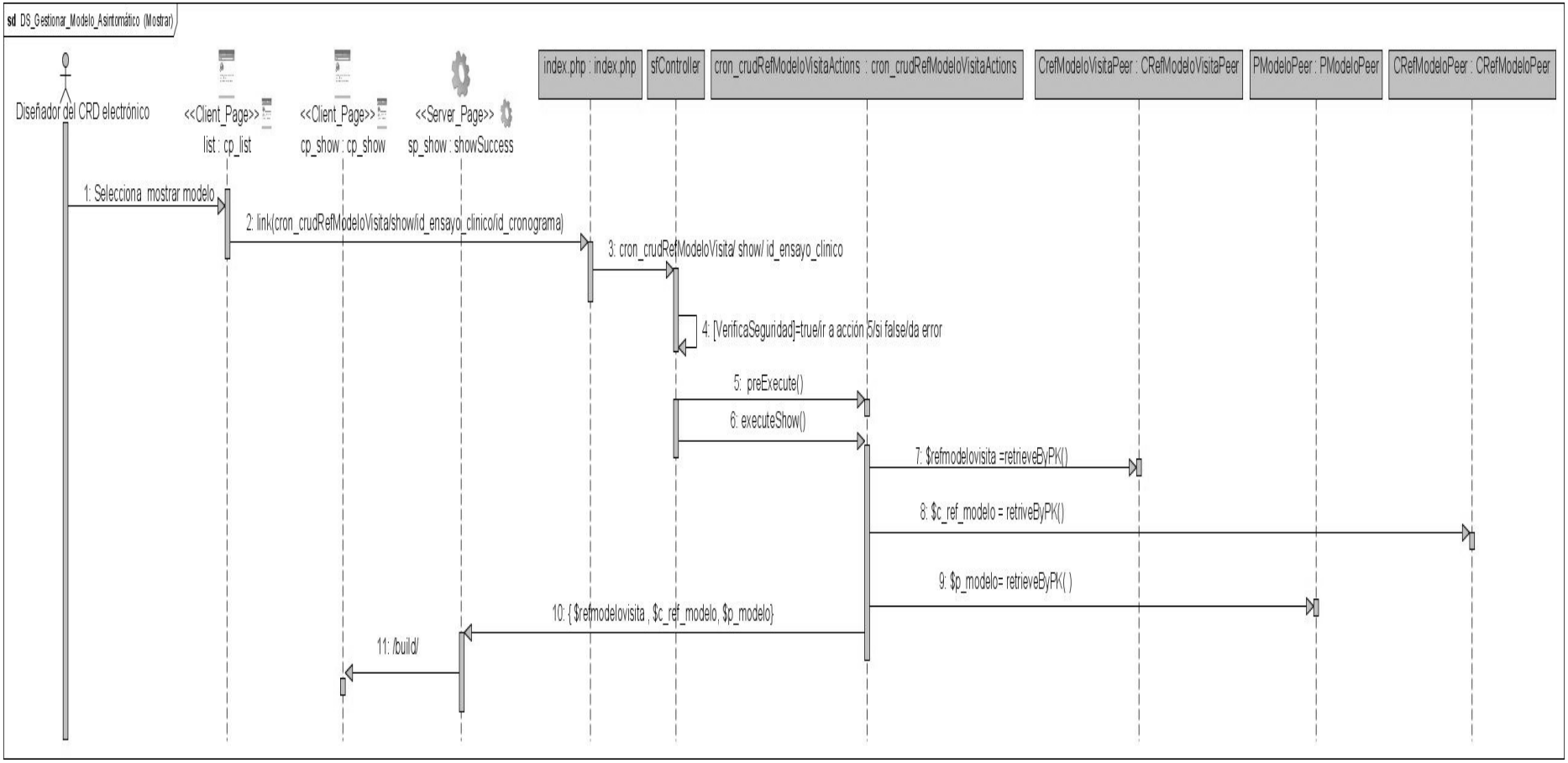

Figura 29 Diagrama de interacción Gestionar_Modelo_Asintomático (Mostrar Modelo Asintomático).

Gestionar_Modelo_Asintomático(Búsqueda avanzada de Asintomático).

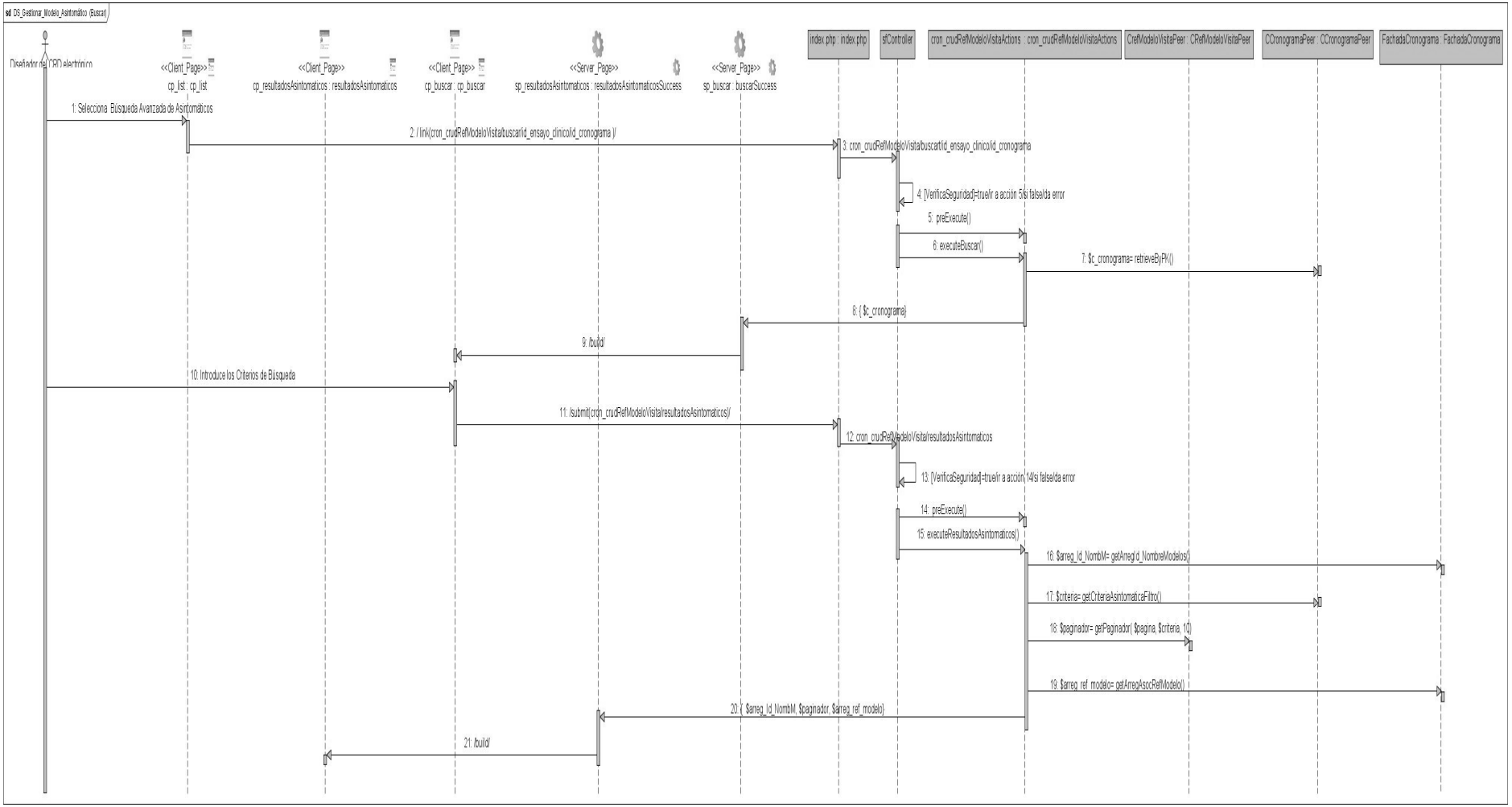

Figura 30 Diagrama de interacción Gestionar_Modelo_Asintomático (Búsqueda avanzada de Asintomáticos).

Diagrama de interacción para Gestionar_Cronograma_específico

Gestionar_Cronograma_Específico (Notificar Inserción Modelo Paciente).

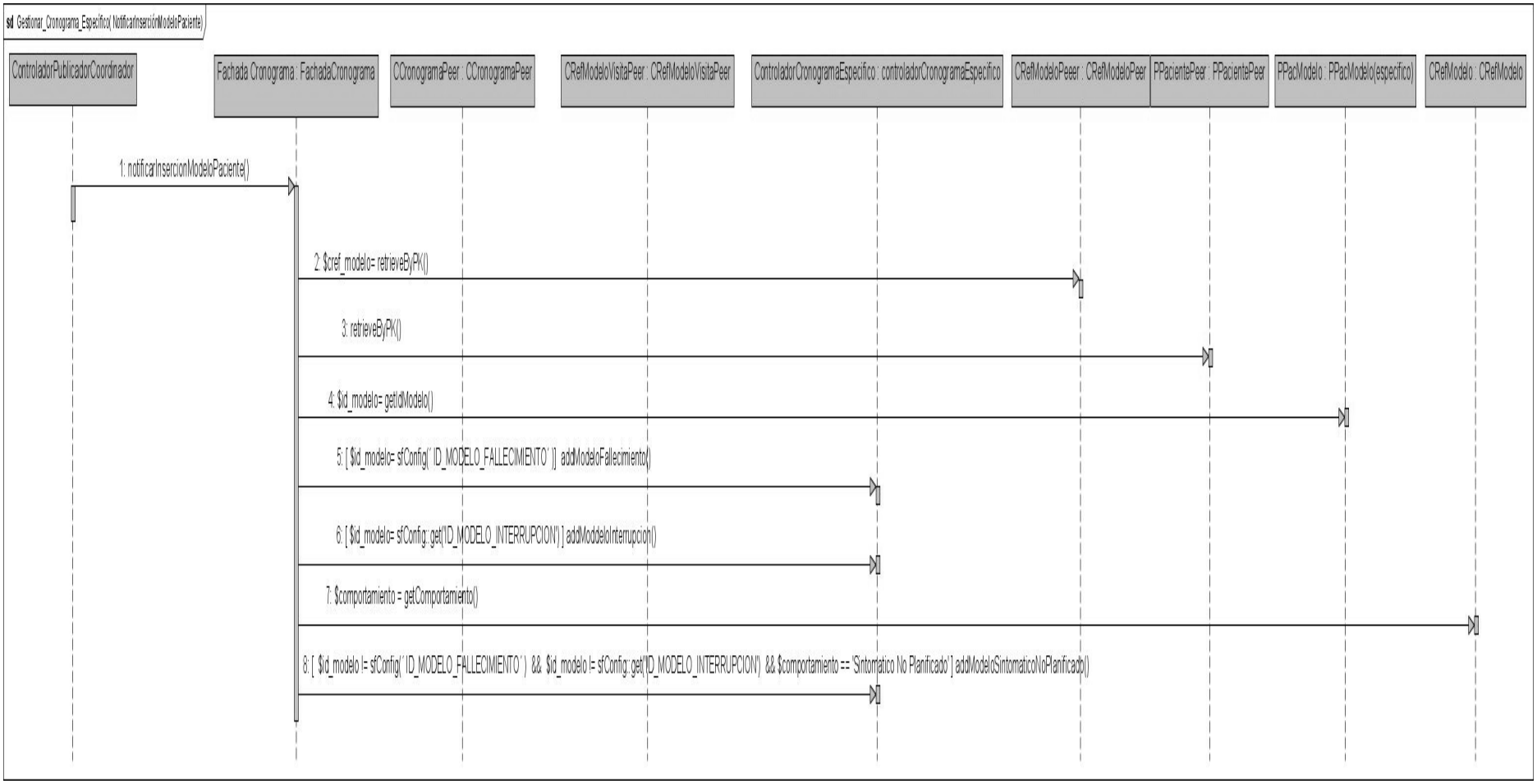

Figura 31 Diagrama de interacción Gestionar_Cronograma_Específico (Notificar Inserción Modelo Paciente).

Gestionar_ Cronograma_Específico (Notificar Inclusión de Paciente).

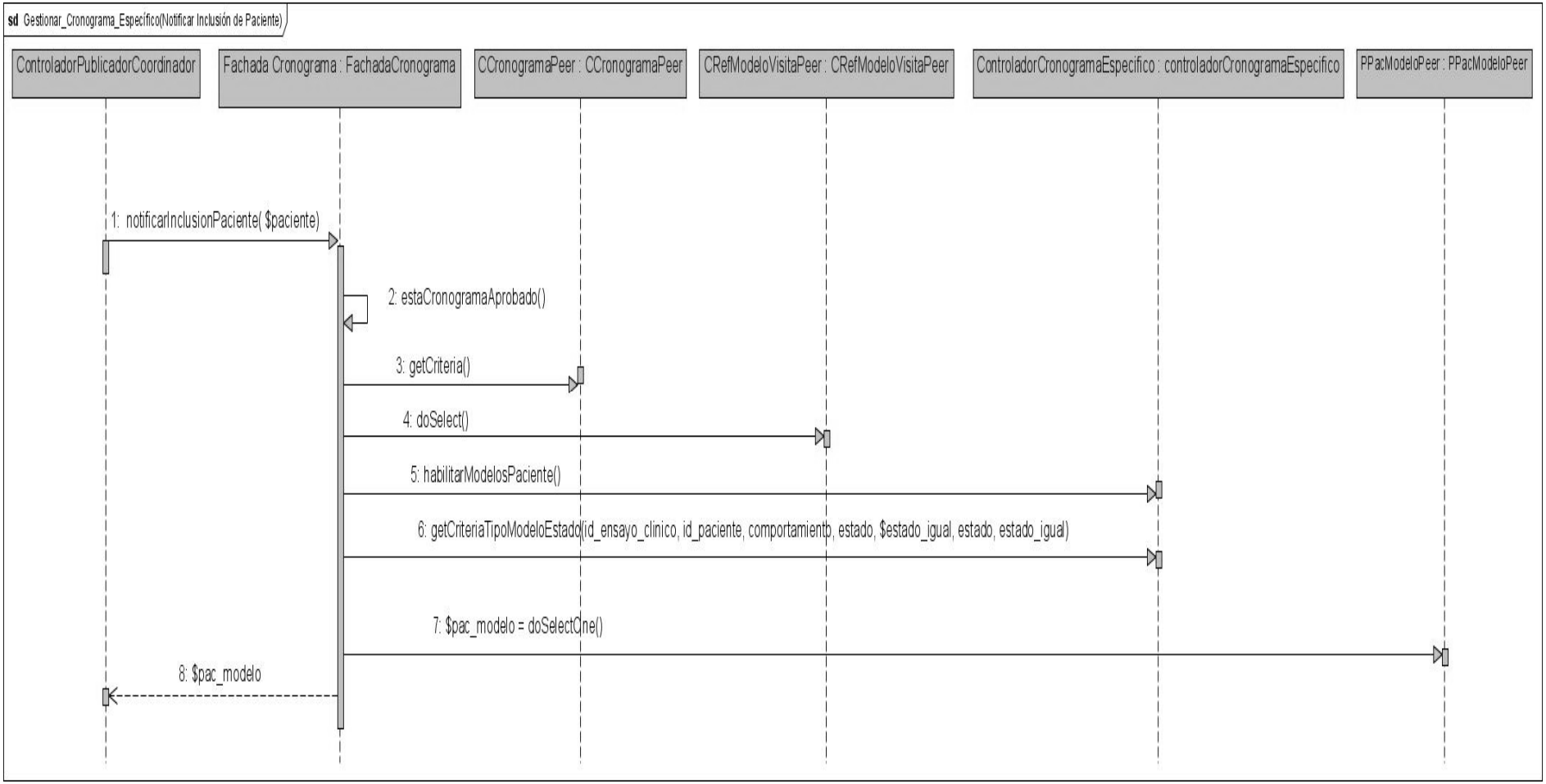

Figura 32 Diagrama de interacción Gestionar_Cronograma_Específico (Notificar Inclusión de Paciente).

Gestionar_ Cronograma_Específico (Crear Cronograma Específico).

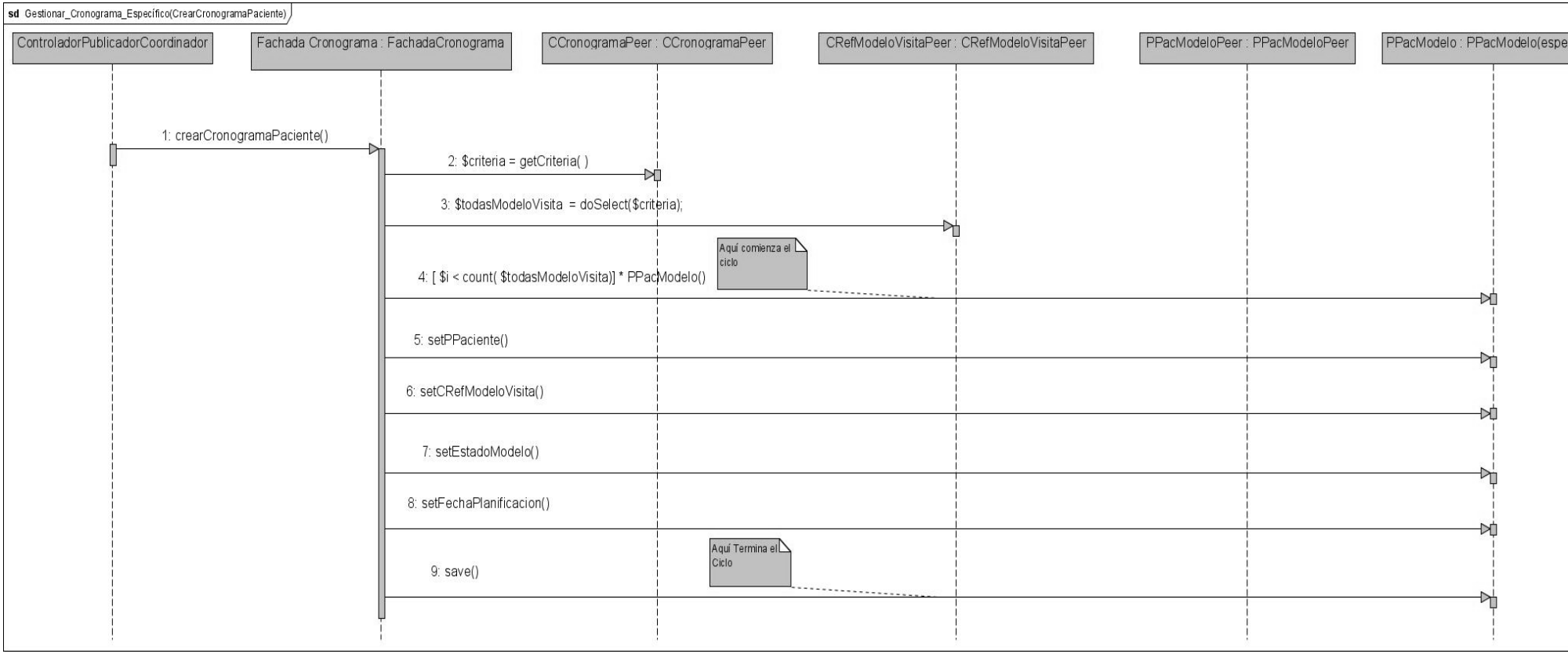

Figura 33 Diagrama de interacción Gestionar_ Cronograma_Específico (Crear Cronograma Específico).

2.2.5 Interfaz de Usuario

Las interfaces facilitan la comunicación entre el usuario y la computadora, permitiendo que los usuarios aprovechen al máximo el rendimiento del programa; a continuación se muestran interfaces pertenecientes al submódulo Cronograma de Ejecución.

Mostrar Cronograma.

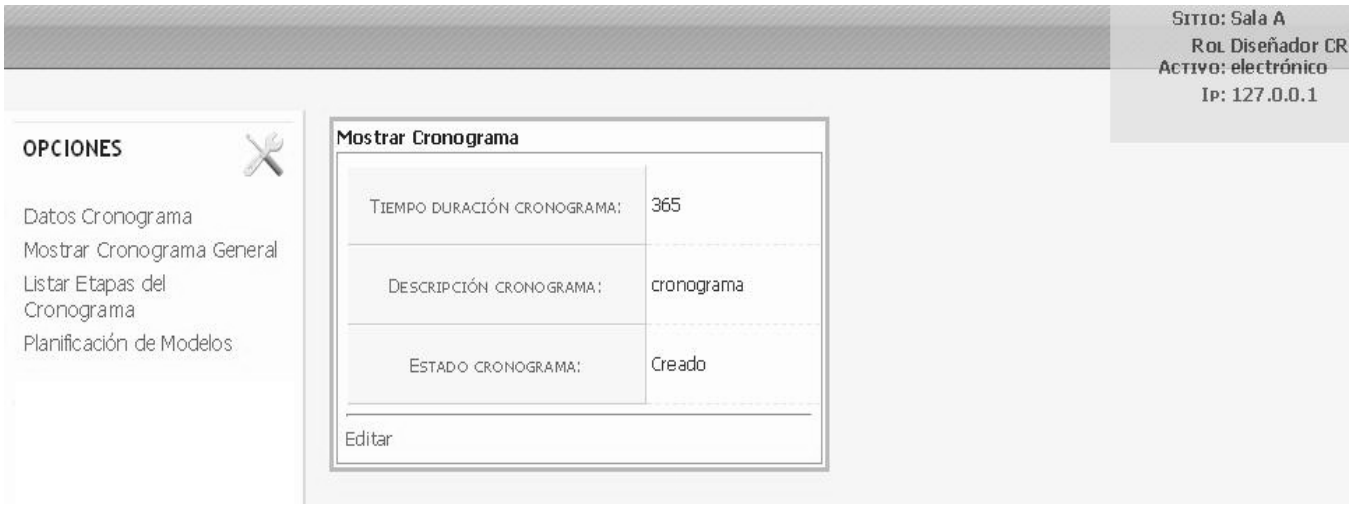

Figura 34 Mostrar Cronograma.

Crear Cronograma.

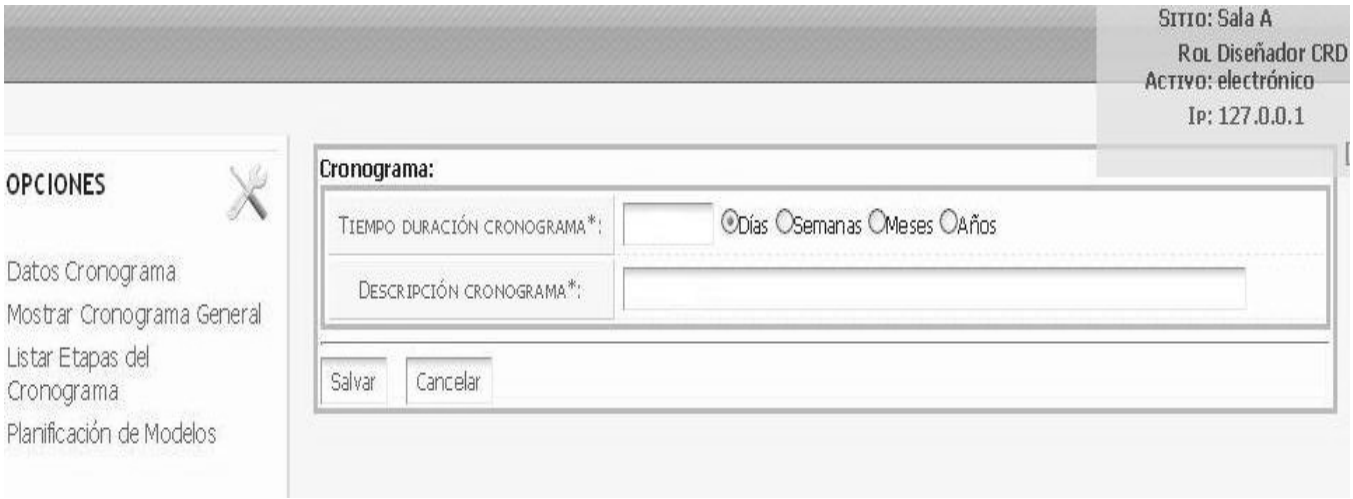

Figura 35 Crear Cronograma.

Mostrar Cronograma General.

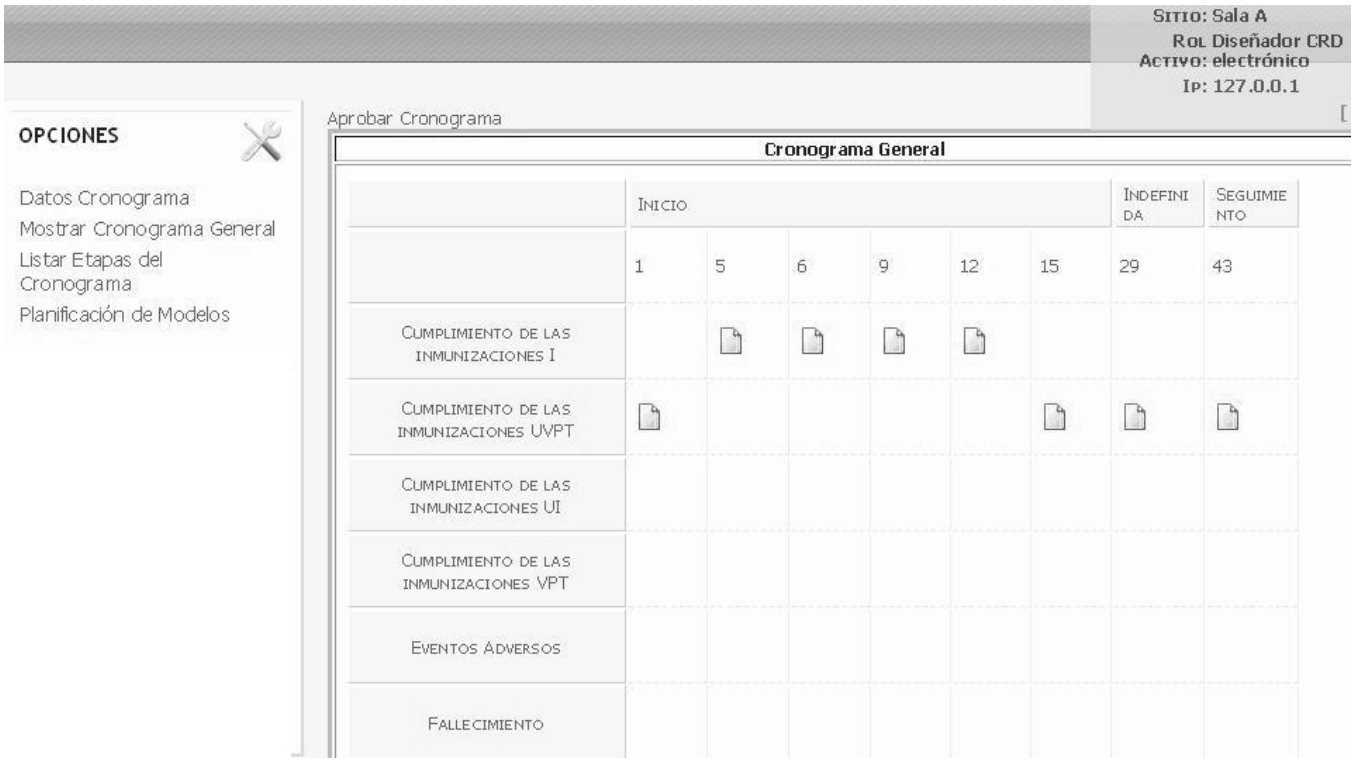

Figura 36 Mostrar Cronograma General.

Crear Etapa.

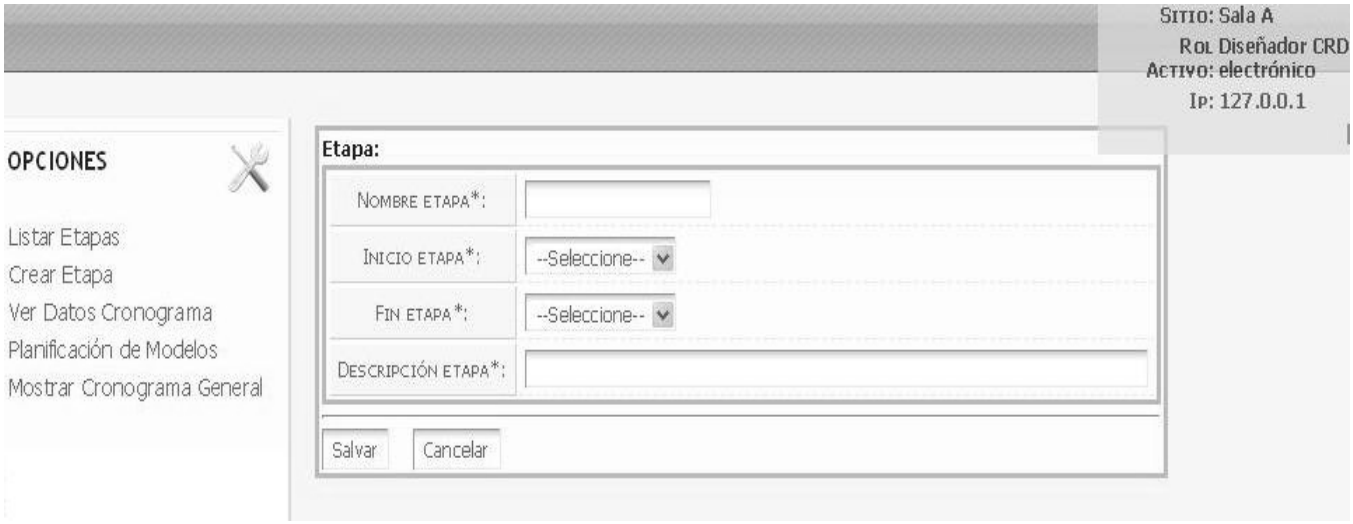

Figura 37 Crear Etapa.

Listar Etapa.

Figura 38 Listar Etapa.

Mostrar Etapa.

Figura 39 Mostrar Etapa.

Programar Modelo por período.

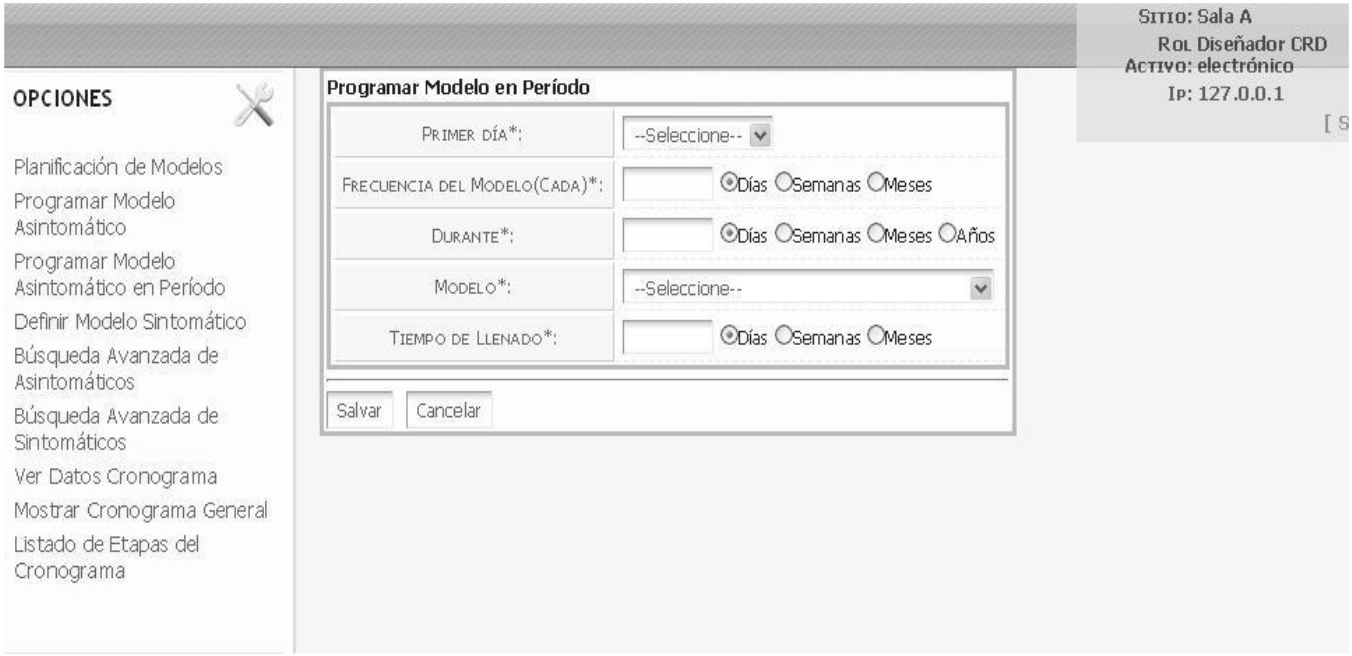

Figura 40 Programar Modelo en Período.

Mostrar los modelos Planificados.

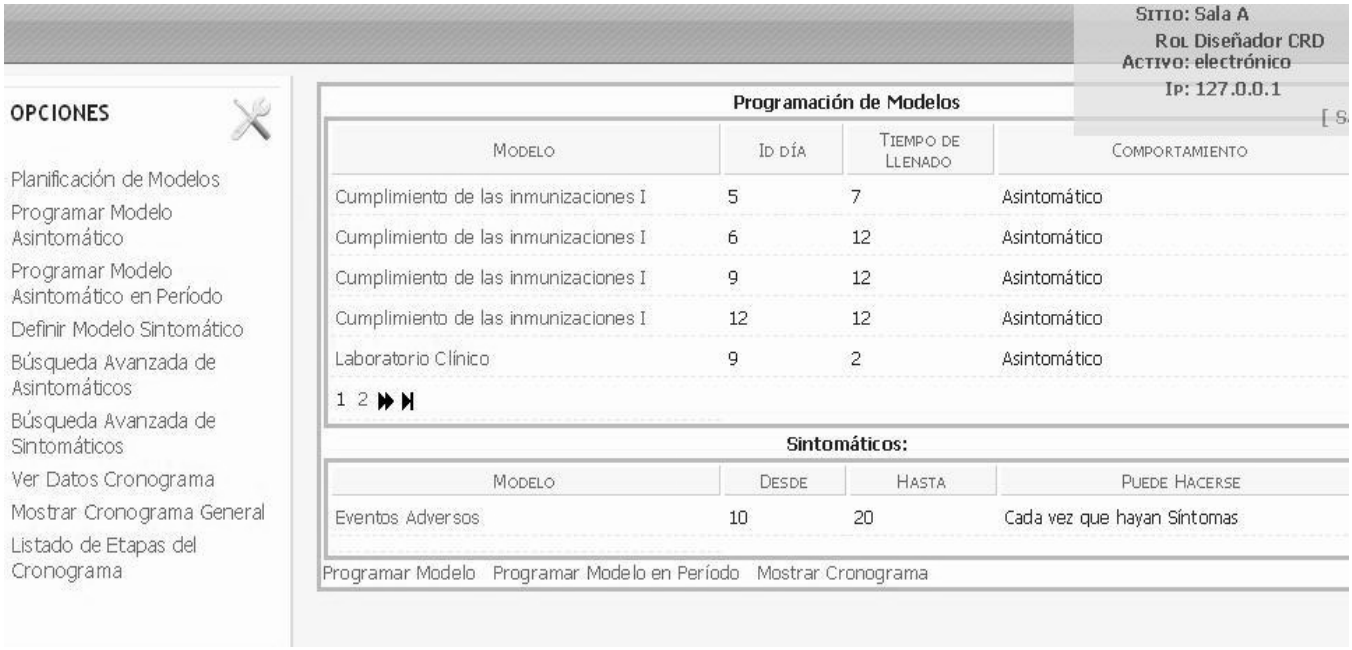

Figura 41 Mostrar los Modelos Planificados.

Buscar modelo sintomático.

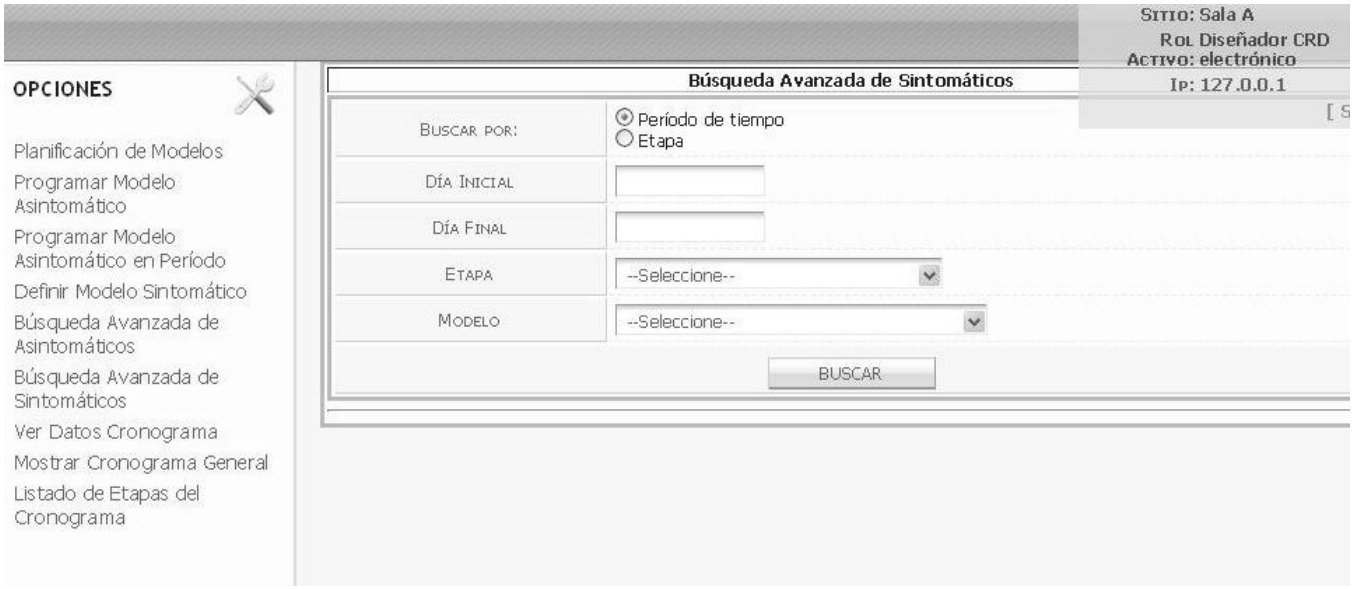

Figura 42 Buscar Modelos Sintomáticos.

Craer modelo sintomático.

Figura 43 Crear Modelo Sintomático.

Mostrar modelo sintomático.

Figura 44 Mostrar Modelo Asintomático.

Crear modelo asintomático.

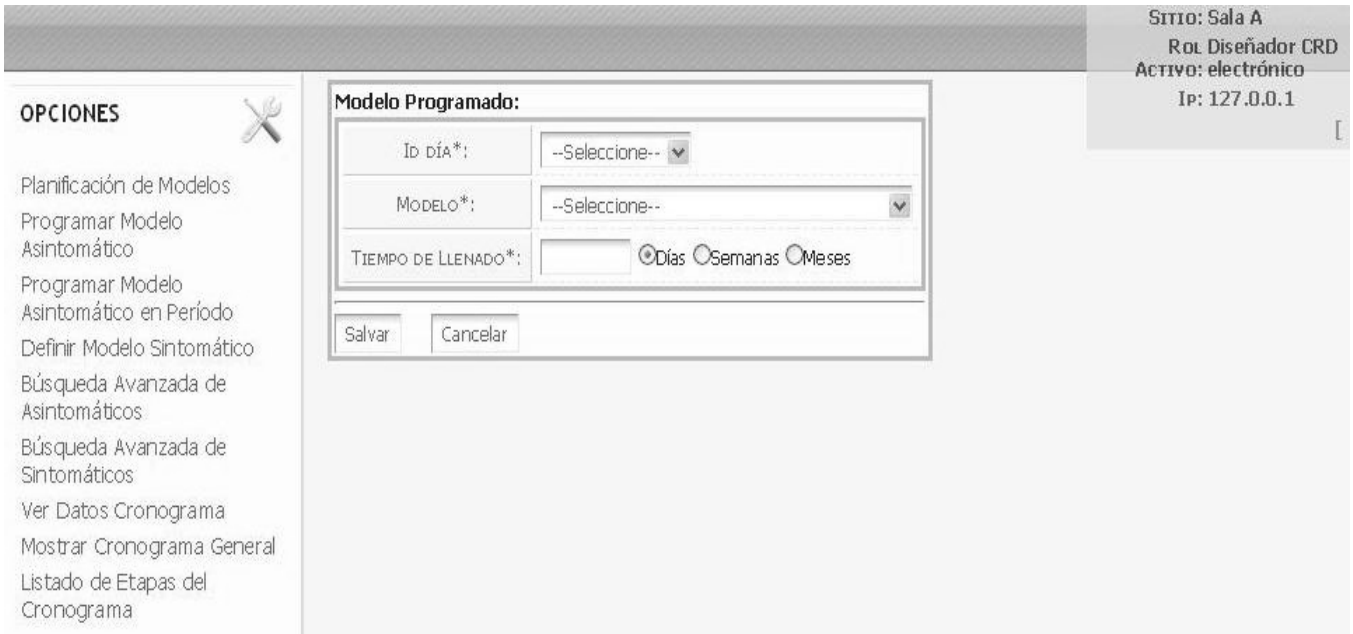

Figura 45 Craer Modelo Asintomático.

Buscar modelo asintomático.

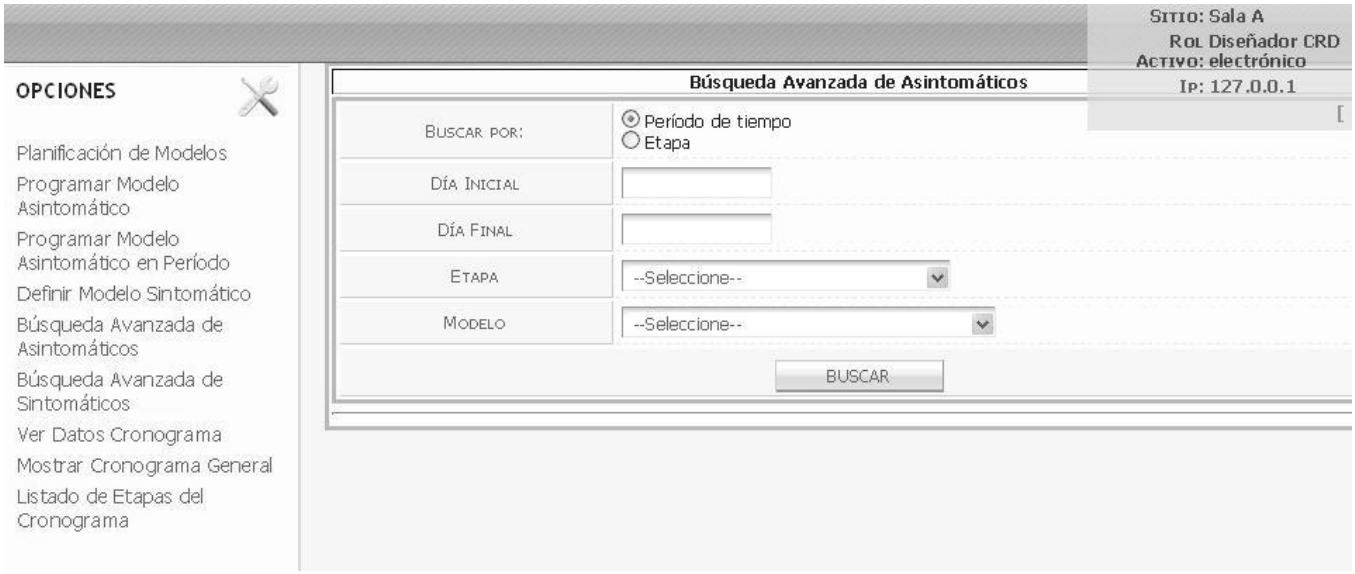

Figura 46 Buscar Modelos Asintomáticos.

Mostrar modelo asintomático.

Figura 47 Mostrar Modelo Asintomático.

2.2.6 Mapa de navegación.

Un Mapa de navegación define la estructura jerárquica de páginas lógicas de la aplicación y los niveles de los usuarios en la navegación. Cada página lógica es candidata a convertirse en una interfaz de usuario.

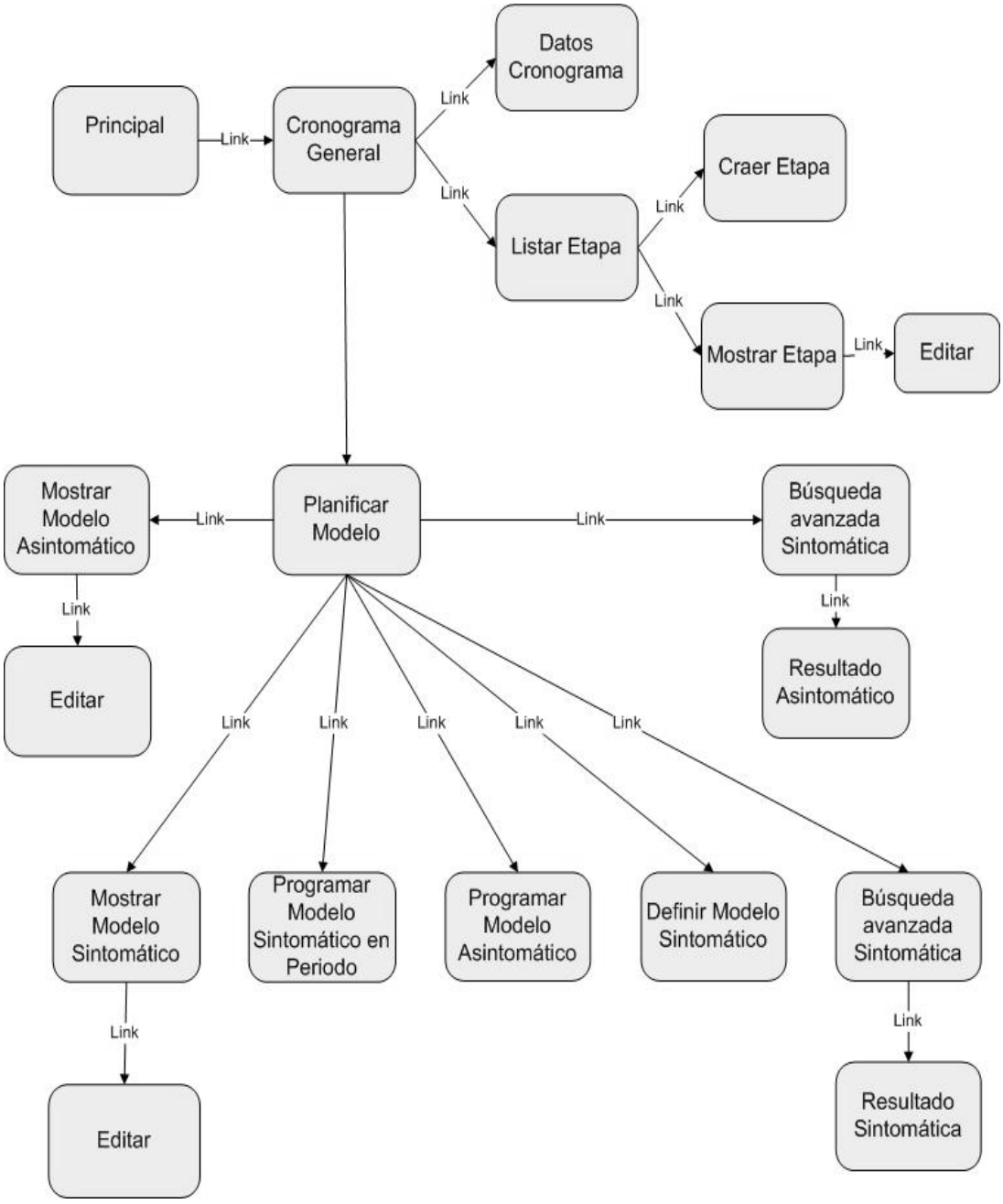

Figura 48 Mapa de navegación.

2.2.7 Diagrama de despliegue.

Un diagrama que muestra la configuración de los nodos de proceso y las instancias de componentes y objetos que residen en ellos. [13]

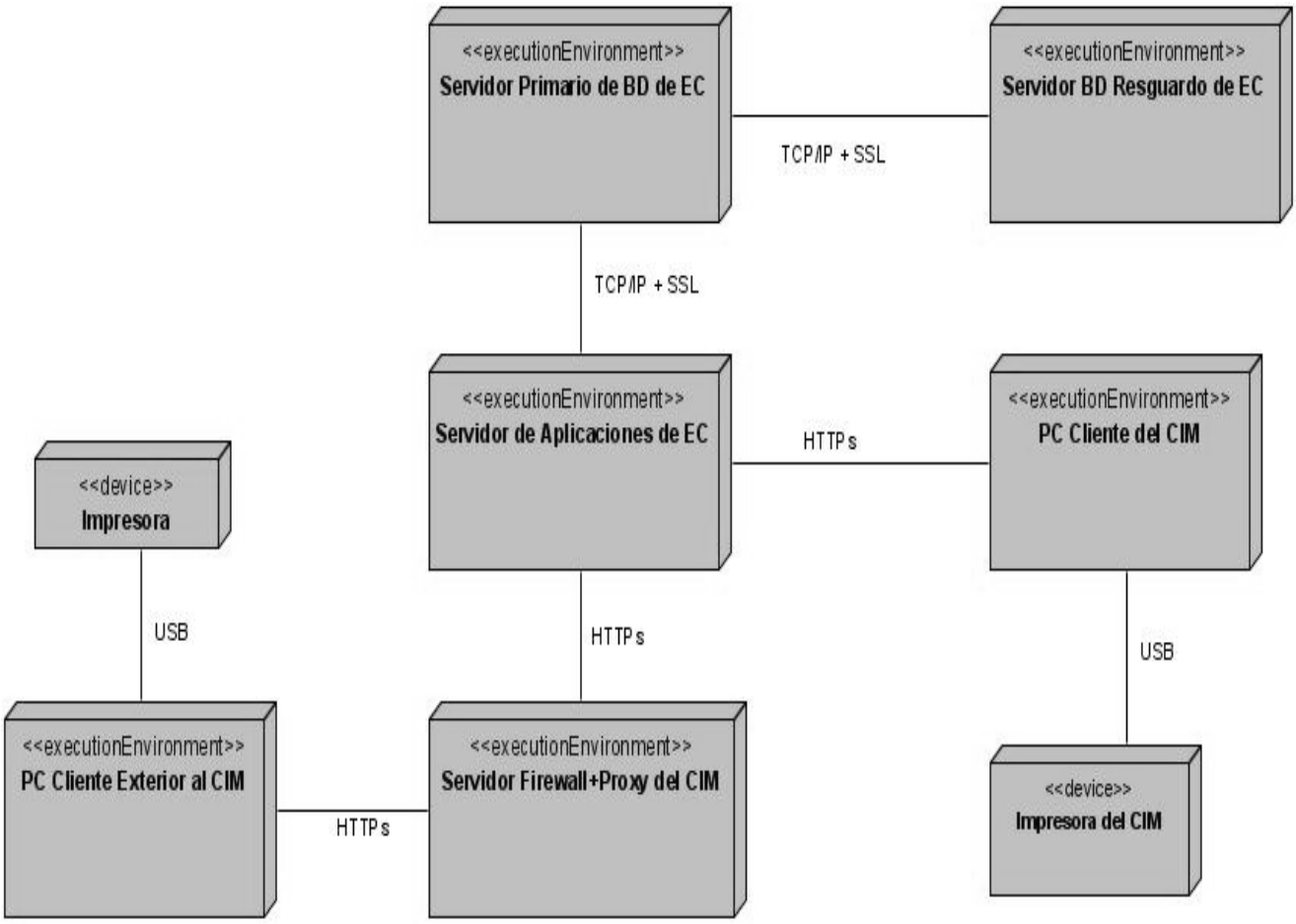

Figura 49 Diagrama de despliegue.

2.3 Modelo de datos.

El modelo de datos describe la representación lógica y física de los datos persistentes, este modelo es el artefacto resultante de la actividad de diseñador de base de datos.

2.3.1 Diagrama de clases persistentes.

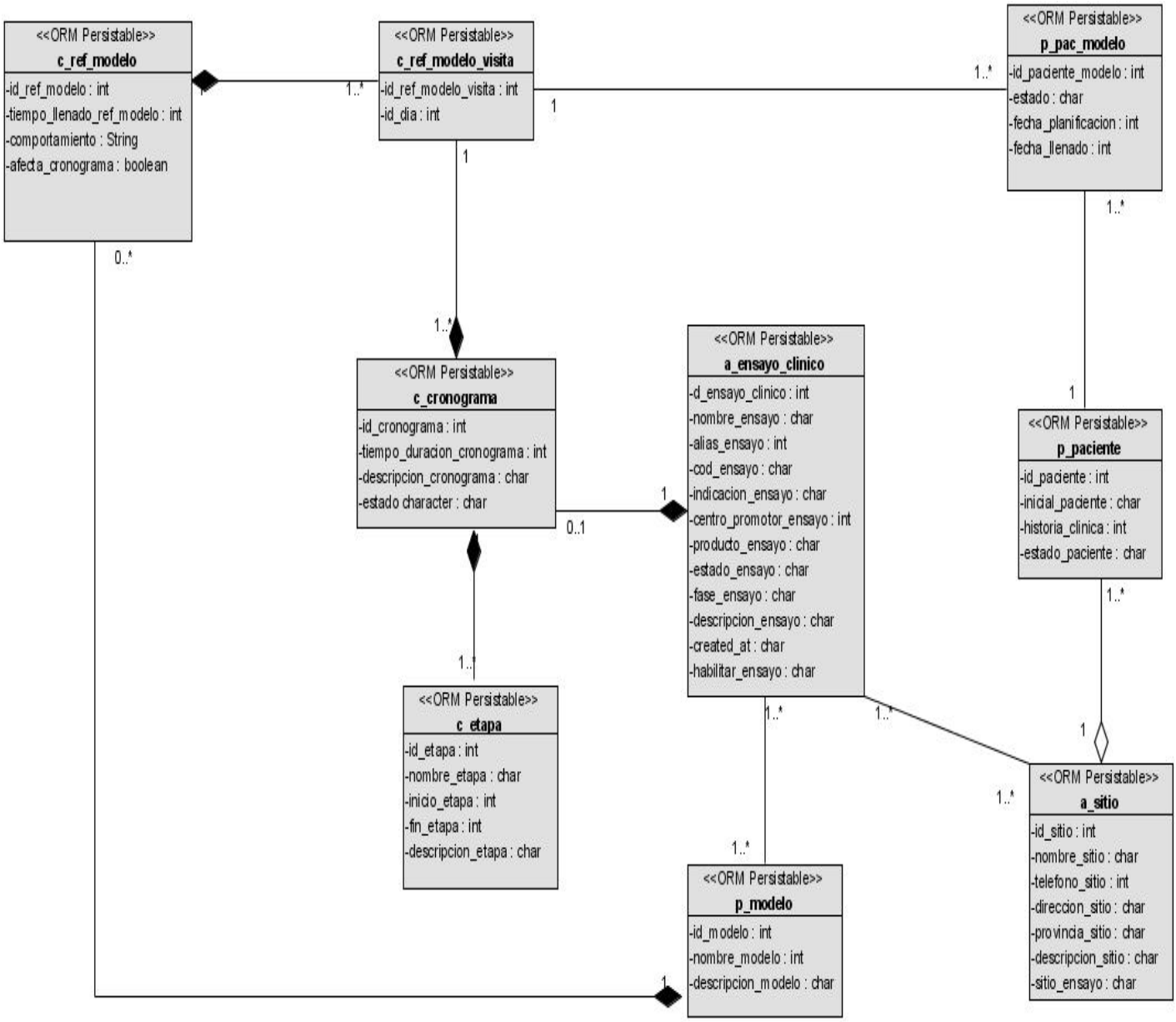

Figura 50 Diagrama de clases persistentes.

2.3.2 Modelo físico.

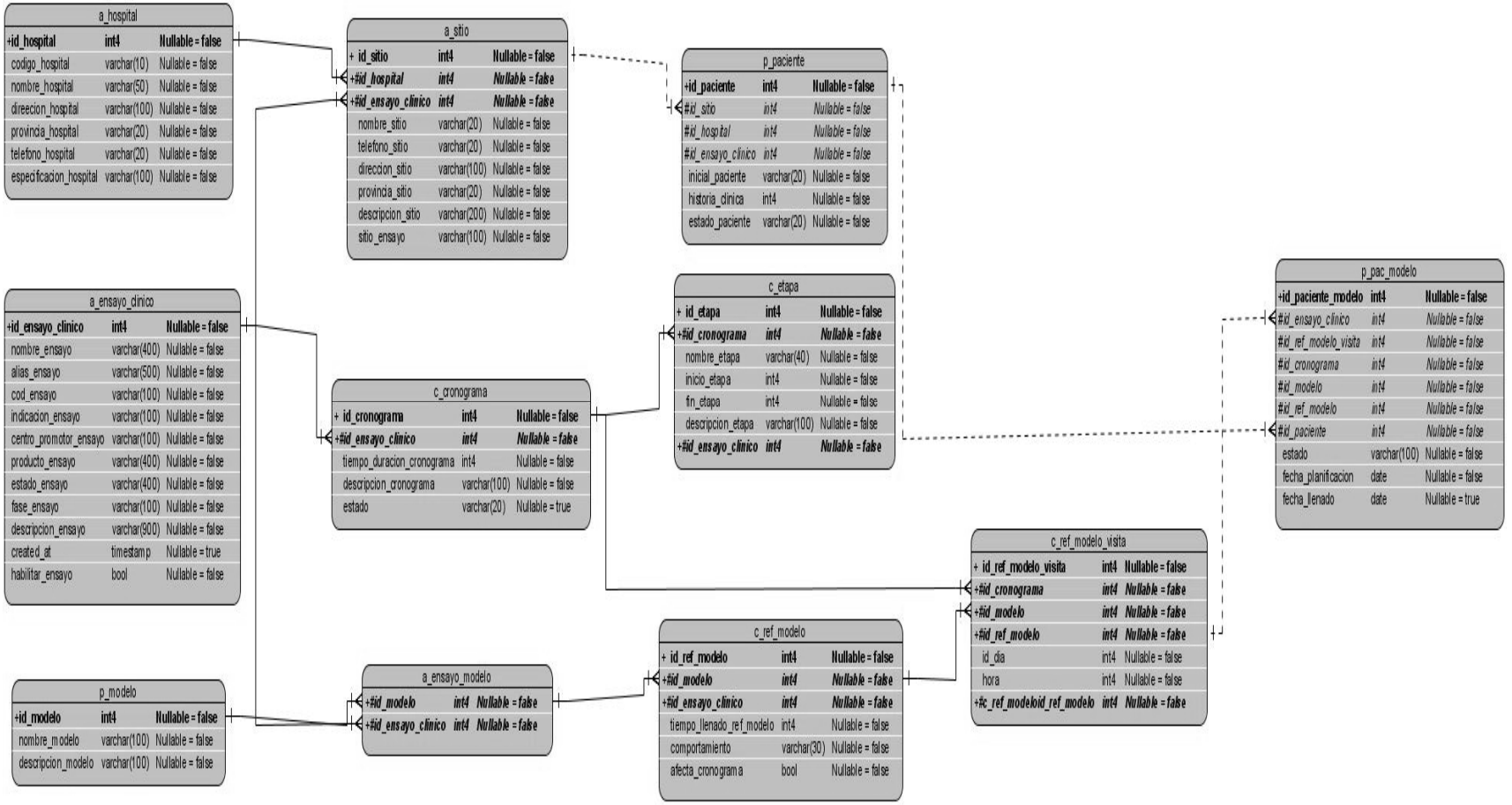

Figura 51 Modelo físico

2.3. 3 Descripción de las tablas de la base de datos.

Tabla 13. Descripción de las tablas: a_ensayo_clinico.

Nombre: a_ensayo_clinico		
Descripción: Contiene todo los datos relacionados con el ensayo clínico.		
Atributo	**Tipo**	**Descripción**
id_ensayo_clinico	integer	Identificador de la tabla a_ensayo_clínico.
nombre_ensayo	varying(400)	Nombre del ensayo clínico a tratar
alias_ensayo	varying(500)	Diminutivo que se le asigna al nombre del ensayo clínico.
cod_ensayo	varying(100)	Código del ensayo clínico
Indicacion_ensayo	varying(100)	Indicaciones asociadas al ensayo clínico
centro_promotor_ensayo	varying(100)	Nombre del centro que llevará a cabo la investigación.
poducto_ensayo	varying(400)	Nombre del producto a evaluar en el ensayo clínico.
estado_ensayo	varying(400)	Se tiene el estado del ensayo que puede ser habilitado o deshabilitado.
fase_ensayo	varying(100)	Recoge las distintas fases del ensayo clínico.
descripcion_ensayo	varying(900)	Se da una breve explicación sobre el ensayo clínico.
created_at	times	Guarda la fecha del momento exacto de creado, el ensayo clínico.
habilitar_ensayo	boolean	Variable que retorna un valor de verdadero (si está habilitado) o falso (si está deshabilitado)

Tabla 14. Descripción de las tablas: a_ensayo_modelo.

Nombre: a_ensayo_modelo		
Descripción: Contiene el id de los modelos para determinados ensayos.		
Atributo	**Tipo**	**Descripción**
id_modelo	integer	Identificador de la tabla modelo

Tabla 15. Descripción de las tablas: c_cronograma.

Nombre: c_cronograma		
Descripción: Contiene todo los datos relacionados con cronograma.		
Atributo	Tipo	Descripción
id_cronograma	integer	Identificador de la tabla c_cronograma
tiempo_duracion_cronograma	integer	Guarda el tiempo que durará el ensayo clínico
descripcion_cronograma	varying(100)	Se da una breve explicación sobre el cronograma.
estado	varying(20)	Recoge el estado del cronograma este puede ser creado y aprobado.

Tabla 16. Descripción de las tablas: c_etapa.

Nombre: c_etapa		
Descripción: Contiene todo los datos relacionados con las etapas.		
Atributo	**Tipo**	**Descripción**
id_etapa	integer	Identificador de la tabla c_etapa
nombre_etapa	varying(40)	Nombre que recibirá la etapa
inico_etapa	integer	Día de inicio de la etapa
fin_etapa	integer	Día en que concluirá la etapa
descripcion_etapa	varying(100)	Se da una breve explicación sobre la

		etapa.

Tabla 17. Descripción de las tablas: c_ref_modelo.

Nombre: c_ref_modelo		
Descripción: Contiene todo los datos relacionados con las referencia modelo existente.		
Atributo	**Tipo**	**Descripción**
id_ref_modelo	integer	Identificador de la tabla c_ref_modelo
tiempo_llenado_ref_modelo	integer	Tiempo de holgura para el modelo
comportamiento	varying(30)	El modelo puede ser sintomático Planificado o Sintomático no planificado o asintomático.
afecta_cronograma	boolean	Esta variable retorna verdadero (modelo afecta el cronograma) o falso (modelo no afecta el cronograma)

Tabla 18. Descripción de las tablas: c_ref_modelo_visita.

Nombre: c_ref_modelo_visita		
Descripción: Contiene todo los datos relacionados con las referencias modelos existente.		
Atributo	**Tipo**	**Descripción**
id_ref_modelo_visita	integer	Identificador de la tabla c_ref_modelo_visita
Id_dia	integer	Identificador del día
hora	integer	Hora en la que se debe realizar el examen.

Tabla 19. Descripción de las tablas: p_pac_modelo.

Nombre: p_pac_modelo		
Descripción: Contiene todo los datos relacionados con los modelos ya aplicado a cada paciente.		
Atributo	**Tipo**	**Descripción**
Id_paciente_modelo	integer	Identificador de la tabla p_pac_modelo
estado	varying(100)	Recoge el estado del modelo puede ser vacio, completo, incompleto, firmado, monitoreado e interrumpido.
fecha_planificacion	Date	Fecha en la que es planificado el modelo.
fecha_llenado	Date	Fecha en la que realmente se hace el modelo.

Tabla 20. Descripción de las tablas: p_paciente.

Nombre: p_paciente		
Descripción: Contiene todo los datos relacionados con los pacientes.		
Atributo	**Tipo**	**Descripción**
Id_paciente	integer	Identificador de la tabla paciente
Inicial_paciente	varying(20)	Se ponen las iniciales del paciente
historia_clinica	integer	Tiene el número de la historia clínica
estado_paciente	varying(20)	Recoge el estado del paciente puede ser incluido o mal incluido.

2.4 Código fuente.

2.4.1 Método guardarModeloPeriodo().

Este método pertenece a la clase CCronograma, recibe el modelo, el primer día en que se quiere planificar, la frecuencia y el tiempo durante el cual se desea planificar dicho modelo así como el tiempo de llenado del mismo, y registra en la base de datos el modelo durante el período de tiempo definido, obteniendo la CRefModelo adecuada en caso de existir y si no, la crea. Valida la integridad de los datos haciendo uso de transacciones.

```
public function guardarModeloPeriodo( $tiempo_llenado, $frecuencia, $duracion, $dia_inicial,
$id_modelo, $comportamiento, $afecta_cronograma )
{
      if( $frecuencia==0)
            $frecuencia=1;
      $con = Propel::getConnection();
      try
      {
            $con->begin();
            $c_ref_modelo = CCronogramaPeer::getRefModeloCorrecta( $id_modelo,
            $this->id_ensayo_clinico, $tiempo_llenado, $comportamiento, $afecta_cronograma );

            $cantV = intval( $duracion / $frecuencia) +1;
            if( $duracion == 0)
                  $cantV = 0;
            $contVisitas=0;
            while( $dia_inicial <= $this->getTiempoDuracionCronograma() &&
                                                            $contVisitas < $cantV)
            {
                  $ref_modelo_visita=new CRefModeloVisita();
                  $ref_modelo_visita->setIdRefModeloVisita(
 $ref_modelo_visita->getNuevoId( $this->getIdCronograma(),  $this->getIdEnsayoClinico() ) );

                  $ref_modelo_visita->setIdRefModelo($c_ref_modelo->getIdRefModelo() );
                  $ref_modelo_visita->setIdCronograma($this->getIdCronograma());
                  $ref_modelo_visita->setIdEnsayoClinico($this->getIdEnsayoClinico());
                  $ref_modelo_visita->setIdModelo($id_modelo);
                  $ref_modelo_visita->setIdDia($dia_inicial);
                  $ref_modelo_visita->setHora( 0);
                  $ref_modelo_visita->save();
                  $dia_inicial += $frecuencia;
                  $contVisitas++;
            }

            $con->commit();
      }
      catch (Exception $e)
      {
            $con->rollback();
            throw $e;
      }
}
```

2.4.2 Método getCriteria()

Método de la clase CCronogramaPeer, recibe el id_modelo, día, id_cronograma, id_ensayo_clinico, comportamiento y si el comportamiento es igual. Construye y devuelve un objeto Criteria con los parámetros de búsqueda señalados, si alguno de estos parámetros se pasa con valor NULL no se agrega dicho criterio; si comportamientoIgual tiene valor verdadero se buscan CRefModeloVisita que tengan el comportamiento pasado (si este no es NULL), sino, se buscan las que tengan comportamiento distinto al indicado por parámetro.

Esta flexibilidad permite al método que pueda ser reutilizado desde diferentes contextos, tanto si se desea listar usando un paginador, como si se necesitan los objetos CRefModeloVisita.

```
public static function getCriteria ($id_modelo, $dia, $id_cronograma, $id_ensayo,
$comportamiento, $comportamientoIgual)
{
	$critRMV= new Criteria();
	if( $id_cronograma !=NULL && $id_cronograma !="")
		$critRMV->add( CRefModeloVisitaPeer::ID_CRONOGRAMA, $id_cronograma);
	$critRMV->add( CRefModeloVisitaPeer::ID_ENSAYO_CLINICO, $id_ensayo );

	if( $dia != NULL && $dia !="")
		$critRMV->add( CRefModeloVisitaPeer::ID_DIA, $dia );

	if( $id_modelo !=NULL && $id_modelo!="" && $id_modelo!='--Seleccione--')
		$critRMV->add(CRefModeloVisitaPeer::ID_MODELO, $id_modelo);

	$critRMV->addJoin( CRefModeloVisitaPeer::ID_REF_MODELO, CRefModeloPeer::ID_REF_MODELO);
	$critRMV->addJoin( CRefModeloVisitaPeer::ID_ENSAYO_CLINICO,
	CRefModeloPeer::ID_ENSAYO_CLINICO);
	$critRMV->addJoin( CRefModeloVisitaPeer::ID_MODELO, CRefModeloPeer::ID_MODELO);

	if( $comportamientoIgual )
		$critRMV->add( CRefModeloPeer::COMPORTAMIENTO, $comportamiento);
	else
		$critRMV->add( CRefModeloPeer::COMPORTAMIENTO, $comportamiento,
	Criteria::NOT_EQUAL);

	$critRMV->addAscendingOrderByColumn(CRefModeloVisitaPeer::ID_DIA);

	return $critRMV;
}
```

2.5 Conclusiones.

En este capítulo se definió el estilo arquitectónico a utilizar así como los diagramas de clases Web del diseño, los diagramas de secuencia y diagrama de despliegue. Cumpliendo con el rol de diseñador de Base de datos se realiza el modelo de clases persistentes, el diagrama Entidad-Relación y la descripción de las tablas, todo esto con el objetivo de lograr una programación libre de imprecisiones. Queda reflejado en este capítulo fragmentos de código asociado al resultado de la investigación.

CONCLUSIONES

- Se realizó un estudio de las aplicaciones Web existentes para el Manejo de Datos de Ensayos Clínicos.
- Se realizó el diseño del submódulo cronograma de ejecución para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.
- Se implementó el submódulo cronograma de ejecución para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.

RECOMENDACIONES

- Profundizar en el estudio de herramientas de planificación con el objetivo de generalizar la aplicación para cualquier sistema con necesidad de realizar Cronogramas de ejecución.

- Implementar la Ayuda del submódulo con el objetivo de informarle al usuario cómo trabajar con el mismo y de mostrar información importante referente al protocolo del ensayo.

REFERENCIA BIBLIOGRÁFICA

1. Organización Mundial de la Salud. *Organización Mundial de la Salud.* [En línea] [Citado el: *noviembre 14, 2007.] http://www.who.int/research/es/.*

2. Diccionario de cáncer . Diccionario de cáncer . [Online] [Cited: noviembre 2007, 2007.] http://www.cancer.gov/templates/db_alpha.aspx?lang=spanish&CdrID=45333.

3. Diccionario de cáncer . Diccionario de cáncer . [En línea] [Citado el: 14 de noviembre de 2007.] http://www.cancer.gov/templates/db_alpha.aspx?lang=spanish&CdrID=45961.

4. Hipócrates. *Hipócrates.* [En línea] [Citado el: noviembre 21, 2007.] http://www.hipocrates.com/.

5.Biblioteca. *UML y Patrones Introducción al análisis y diseño Orientado a Objeto.* [En línea] [Citado el: 21 de Noviembre de 2007.] http://bibliodoc.uci.cu/pdf/reg03050.pdf.

6. Entorno Virtual de Aprendizaje. *Introducción a la Ingeniería de Software.* Ciudad Habana: s.n., 2007-2008.

7. Eclipse. *Eclipse.* [En línea] [Citado el: enero 18, 2008.] http://www.eclipse.org/home/newcomers.php.

8. garbage collector. *garbage collector.* [En línea] [Citado el: enero 24, 2008.] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

9. Biblioteca. UML y Patrones Introducción al análisis y diseño Orientado a Objeto. [En línea] [Citado el: Noviembre 21, 2007.] http://bibliodoc.uci.cu/pdf/reg03050.pdf.

10. **rodríguez, David y rodríguez, Daniel.** *Herramienta para la generación de código.* 2007. Ciudad de la Habana.

11. **Universidad de Sevillas.** Departamento de lenguaje y sistemas informáticos. *Departamento de lenguaje y sistemas informáticos.* [En línea] 2005/2006. [Citado el: febrero 24, 2008.] http://www.lsi.us.es/docencia/pagina_asignatura_doc.php?id=13&cur=2005.

12. Ayuda extendida del Rational Rose Enterprise. Edition 2003

13. Biblioteca. UML y Patrones Introducción al análisis y diseño Orientado a Objeto. [En línea] [Citado el: Noviembre 21, 2007.] http://bibliodoc.uci.cu/pdf/reg03050.pdf.

BIBLIOGRAFÍA

1. Ingeniería de Software. *Ingeniería de Software.* [En línea] [Citado el: 15 de Diciembre de 2007.] http://teleformacion.uci.cu/mod/resource/view.php?id=6655.

2. Texas Heart Institute. *Texas Heart Institute.* [En línea] [Citado el: 11 de Octubre de 2007.] http://www.texasheartinstitute.org/HIC/Topics_Esp/FAQ/clinical_trials_span.cfm.

3. Centro de Inmunología Molecular. *Centro de Inmunología Molecular.* [En línea] [Citado el: 11 de Octubre de 2007.] http://www.cim.sld.cu/.

4. Farmacología clínica. *Farmacología clínica.* [En línea] [Citado el: 11 de Octubre de 2007.] http://www.se-fc.org/0301.php.

5. BLOG de SOFTWARE GENEBETA. *BLOG de SOFTWARE GENEBETA.* [En línea] [Citado el: 17 de Octubre de 2007.] http://www.genbeta.com/2006/07/26-aptana-ide-para-aplicaciones-ajax.

6. Maestros del Web. *Maestros del Web.* [En línea] [Citado el: 17 de Octubre de 2007.] http://www.maestrosdelweb.com/editorial/zendstudio/.

7. Difunde Firefox Redescubre la Web. *Difunde Firefox Redescubre la Web.* [En línea] [Citado el: 23 de octubre de 2007.] http://www.difundefirefox.com/kompozer.

8. **Ramiro Lago Bagués.** Proactiva. *Proactiva.* [En línea] [Citado el: 23 de Octubre de 2007.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

9. Introducción a la computación. *Introducción a la computación.* [En línea] [Citado el: 15 de Diciembre de 2007.] http://entren.dgsca.unam.mx/introduccion/so_carac.html.

10. Oracle. *Oracle.* [En línea] [Citado el: 20 de Enero de 2008.] http://www.oracle.com/global/es/products/database/index.html.

11. ANT. *ANT.* [En línea] [Citado el: 18 de febrero de 2008.] http://www.ant.org.ar/cursos/curso_intro/sistop.html.

12. Pergaminovirtual. *Pergaminovirtual.* [En línea] [Citado el: 19 de enero de 2008.] http://www.pergaminovirtual.com.ar/revista/cgi-bin/hoy/archivos/00000210.shtml.

13. OdesarrolloWeb.com. *OdesarrolloWeb.com.* [En línea] [Citado el: 22 de enero de 2008.] http://www.desarrolloweb.com/directorio/bases_de_datos/oracle/.

14. **Domínguez, Arodys E. Vaillant y Miranda, Duniel Gutiérrez.** *Sistema de Manejo de Datos de Ensayos Clínicos: Diseño e implementación de la Base de Datos.* 2007.

15. **López, Aidacelys Díaz y Rodríguez, Lucia García.** *Sistema de Manejo de Datos de Ensayos Clínicos: Módulo de diseño.* 2007.

16. Eclipse. *Eclipse.* [En línea] [Citado el: 18 de febrero de 2008.] http://plataformaeclipse.com/noticias/2/.

17. **Jabson, Ivar, Booch, Grady y Rumbagh, James.** *El proceso unificado de desarrollo de software.* La Habana: Félix Varela, 2004.

18. UNIFIED MODELING LANGUAGE. UNIFIED MODELING LANGUAGE. [En línea] [Citado el: 19 de enero de 2008.] http://www.uml.org/ .

19. PostgreSQL. PostgreSQL. [En línea] [Citado el: 2007 de octubre de 2007.] http://www.postgresql.org/docs/index.html.

20. Organización Mundial de la Salud. *Organización Mundial de la Salud.* [En línea] [Citado el: *noviembre 14, 2007.] http://www.who.int/research/es/.*

21. Entorno Virtual de Aprendizaje. *Introducción a la Ingeniería de Software.* Ciudad Habana: s.n., 2007-2008.

22. Ayuda extendida del Rational Rose Enterprise. Edition 2003

21. **rodríguez, David y rodríguez, Daniel.** *Herramienta para la generación de código.* 2007. Ciudad de la Habana.

22. Biblioteca. *UML y Patrones Introducción al análisis y diseño Orientado a Objeto.* [En línea] [Citado el: 21 de Noviembre de 2007.] http://bibliodoc.uci.cu/pdf/reg03050.pdf.

23. **Rumbauch, James, Jacobson, Ivar y Booch, Grady.** *EL PROCESO UNIFICADO DE DESARROLLO SOFTWARE Volumen 1.* La Habana : Félix Varela, 2004.

24. **Rumbauch, James, Jacobson, Ivar y Booch, Grady.** *EL PROCESO UNIFICADO DE DESARROLLO SOFTWARE Volumen 2.* La Habana : Félix Varela, 2004

25. **Larman, Craig.** *UML Y Patrones Introducción al análisis y diseño Orientado a Objeto Volumen 1.* La Habana : Félix Varela, 2004.

26. **Larman, Craig.***UML Y Patrones Introducción al análisis y diseño Orientado a Objeto Volumen 2.* La Habana : Félix Varela, 2004.

27. **Pressman, Roger S.** *Ingenieria del software. Un enfoque práctico Parte 1.* La Habana : Félix Varela, 2004.

28. **Pressman, Roger S.** *Ingenieria del software. Un enfoque práctico Parte 2.* La Habana : Félix Varela, 2004.

ANEXOS

Anexo 1- Requisitos Funcionales

RF1- Iniciar cronograma.

RF2- Calcular días

2.1 Convertir los datos entrados en años, meses o semanas a días.

RF3-Mostrar datos del cronograma.

RF4-Editar el cronograma.

RF5-Crear etapa.

RF6-Mostrar etapa.

RF7-Editar etapa.

RF8-Listar etapa.

RF9-Eliminar etapa.

RF10-Planificar modelo asintomático para un día determinado

RF11-Planificar modelo asintomático para un período de tiempo.

RF12-Eliminar modelo asintomático planificado.

RF13-Listar modelos planificados.

RF14-Editar modelo asintomático planificado.

RF15-Mostar la planificación de un modelo determinado.

RF16- Buscar modelos asintomáticos planificados.

16.1 Permitir definir filtros de búsquedas: por modelo, por período de tiempo o por etapa.

RF17-Planificar modelo sintomático para un día determinado

RF18-Eliminar modelo sintomático planificado.

RF19-Editar modelo sintomático planificado.

RF20-Mostar la planificación de un modelo determinado**.**

RF21- Buscar modelos sintomáticos planificados.

16.1 Permitir definir filtros de búsquedas: por modelo, por período de tiempo o por etapa.

RF22- Mostrar Cronograma General.

RF23-Registrar inserción de paciente.

RF24-Generar Cronograma Específico.

RF25-Notificar inserción de modelo a un paciente.

RF26-Listar los modelos asintomáticos que puede completar un paciente.

RF27-Listar los modelos sintomáticos que puede completar un paciente.

Anexo 2- Casos de Uso

1. **Gestionar_Cronograma**
2. **Gestionar_Modelo_Asintomático**
3. **Gestionar_Modelo_Sintomático**
4. **Evento_Periódico**
5. **Gestionar_Etapas**
6. **Mostrar_Cronograma_General**
7. **Gestionar_Cronograma_Específico**

GLOSARIO DE TÉRMINOS

A

Agencia regulatoria: Es la agencia que se encarga de aprobar el protocolo y el diseño de los Cuadernos de Recogida de Datos presentado para un estudio clínico, con el objetivo de autorizar la realización del mismo.

Asintomático: Calificativo que se le da a un modelo al que se le puede definir el día en que se va a llenar.

B

BSD: Licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. El autor, bajo esta licencia, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación.

C

CE: Cronograma de Ejecución.

CIM: Centro de Inmunología Molecular.

CRD: Cuaderno de Recogida de Datos.

D

DOM (Document Object Model): A través de el los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

E

EC: Ensayo Clínico.

I

IDE: Integrated Development Environment (Entorno de Desarrollo Integrado)

S

Sintomático: Calificativo que se le da a un modelo al que no se le puede definir el día en que se debe llenar, pues su realización está sujeto a un síntoma determinado.

Síntoma: Cualquier fenómeno anormal, funcional o sensitivo, percibido por el enfermo, indicativo de enfermedad.