Universidad de las Ciencias Informáticas

Facultad 6

SIMDECC

TÍTULO: SISTEMA DE MANEJO DE DATOS DE ENSAYOS CLÍNICOS CUBANO: DISEÑO E IMPLEMENTACIÓN DEL MÓDULO ADMINISTRACIÓN

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Richard Díaz Pompa.

Osvaldo Duniesky Sandoval Arencibia.

Tutoras:

Ing. Aidacelys López Díaz.

Ing. Linet Cobo Barreras.

Junio de 2008

"Año 50 de la Revolución"

"La clave del éxito depende sólo de lo que podamos hacer de la mejor manera posible."

Henry Wadsonrth Longfellow

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 _____.

Richard Díaz Pompa

_____

Firma del autor

Osvaldo Duniesky Sandoval Arencibia

_____

Firma del autor

Ing. Linet Cobo Barreras

_____

Firma de la tutora

Ing. Aidacelys López Díaz

_____

Firma de la tutora

Datos de Contacto

Ing. Linet Cobo Barreras

Correo: lcobo@uci.cu

La compañera Linet Cobo Barreras, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Diseño e implementación del módulo Administración, se graduó en la Universidad de las Ciencias Informáticas con el título de Ingeniera en Ciencias Informáticas en el año 2007 con honores de título de oro, más de 4.75 pts. en su carrera y por ende obteniendo la bonificación de 5 pts. en la exposición de su trabajo de diploma, actualmente se desempeña como profesora de matemática en la universidad de donde se graduó, además de analista y jefa del módulo administración en el proyecto Ensayos Clínicos al cual pertenece.

Ing. Aidacelys López Días

Correo: alopezd@uci.cu

La compañera Aidacelys López Díaz, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Diseño e implementación del módulo Administración, se graduó en la Universidad de las Ciencias Informáticas con el título de Ingeniera en Ciencias Informáticas en el año 2007, obteniendo 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 desempeña el rol de Líder del proyecto Ensayos Clínicos perteneciente a la facultad 6.

AGRADECIMIENTOS

A la Revolución por habernos dado la oportunidad de estudiar y formarnos como profesionales.

A mis padres Caridad y Osvaldo por mantenerme siempre por el buen camino y darme toda su confianza y apoyo cada vez que los necesito.

A mis 3 hermanas Lisandra, Loraine y Gisell que son las más lindas del mundo, por estar siempre a mi lado aunque nos separe la distancia la mayor parte del año.

A los Ariel, mi cuñado y mi sobrino que han sabido ser, no solo mi familia, sino mis amigos y darme siempre muy buenos consejos.

A Elaine por estar hoy aquí en este día tan importante y que es parte de mi familia también.

A Ana Lupe que siempre se preocupa por nosotros y que a ella le debemos formar parte de la familia de Ensayos Clínicos.

A Linet y Aidacelys que nos han ayudado incondicionalmente para llevar a cabo la tesis y que se han convertido en 2 amigas.

A mis dos grandes amigos Richard y Yoan que sin ellos a mi lado todos estos años, me hubiese costado mucho trabajo llegar hasta aquí hoy.

A mis compañeros de cuarto que siempre han estado ahí para mi.

A la familia de Ensayos Clínicos que constantemente nos estamos apoyando los unos a los otros.

En general a todos los que han estado presentes de una forma u otra y han contribuido a mi formación como persona y siempre han extendido una mano amiga en las buenas como en las malas.

Osvaldo Duniesky Sandoval Arencibia

A Fidel y a la Revolución por dejarme ser parte de este proyecto futuro para forjarme como profesional.

A mis queridos padres María Elena y Leo por guiarme desde pequeño por el buen camino.

A mi tía Francisca, mi hermano Frank, mi prima Aimé, mi tío Juan, Tatico y en general a toda mi familia por estar conmigo siempre en los buenos y malos momentos.

A Lupe por ser la de la idea de insertarme en el proyecto Ensayos Clínicos.

A mis queridas tutoras Aidacelys y Linet que más que tutoras han sido mis amigas por siempre preocuparse y apoyarme en todo momento así como a los profes del proyecto Ensayos Clínicos en especial a Ballester.

A mi querida Aliekna por estar conmigo día a día, por darme su amor, dedicación, comprensión y apoyo en todo momento.

A Osvaldo que más de ser mi dúo de tesis ha sido un gran amigo al igual que Yoan en estos 5 años de carrera.

A mis compañeros de aula por compartir conmigo estos 5 años que jamás podré olvidar y mis colegas del proyecto Ensayos Clínicos que tanto nos ayudamos.

En general muchas gracias a todos aquellos que de una forma u otra siempre me apoyaron y estuvieron conmigo en especial a Imara el primor.

Richard Díaz Pompa

DEDICATORIA

En especial para las dos personas que son la razón de mi existencia y para los cuales vivo, mis padres

A mis hermanas que son todo para mi y las quiero mucho.

Osvaldo Duniesky Sandoval Arencibia.

Especialmente y con gran amor a las dos personas que más quiero en la vida y que lo han dado todo por mí, mis Padres.

A mi abuela Delma, mi tía Francisca, mi hermano Frank, mi prima Aimé, mi hermanita Wendy, a Tay María mi pequeña y primera sobrinita en general a toda mi familia por ser mi inspiración en todo momento.

A mi querida Aliekna por apoyarme en todo momento.

A todos aquellos que me apoyaron y estuvieron conmigo siempre en los buenos y malos momentos.

Richard Díaz Pompa

Resumen

La investigación surge en el marco de trabajo del proyecto Ensayos Clínicos fruto de la colaboración entre el Centro de Inmunología Molecular (CIM) y la Universidad de las Ciencias Informáticas (UCI). El Centro de Inmunología Molecular se dedica desde sus inicios a la creación de biomoléculas para el tratamiento de diferentes enfermedades para lo cual requiere la realización de Ensayos Clínicos (EC), un tipo de estudio clínico que conforma una de las fases del proceso de desarrollo de fármacos. La información que en los mismos se genera es recogida en los Cuadernos de Recogida de Datos (CRD) mediante un protocolo bien definido.

La presente investigación pretende realizar el diseño e implementación de un módulo para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano, encargado de la gestión de usuarios, roles y direcciones IP desde donde trabajarán los usuarios, así como proveer una capa de seguridad a la hora de que estos decidan acceder al sistema, el trabajo se realizará basado en la utilización del framework Symfony, se desarrollarán los artefactos correspondientes a los flujos de trabajo análisis y diseño e implementación.

Palabras Claves

Ensayos Clínicos, SIMDECC, CRD.

Índice

AGRADECIMIENTOS I
DEDICATORIA III
Resumen IV
Palabras Claves IV
Índice V
Índice de Figuras VI
Índice de Tablas VII
Capítulo 1: Fundamentación Teórica. **5**
1.1 Ensayos Clínicos 5
1.2 Módulo Administración de la aplicación 6
1.3 Aplicaciones Web para los Sistemas de Manejo de Datos de Ensayos Clínicos 6
1.4 Fundamentación de las tecnologías y herramientas a utilizar 9
1.4.1 Metodología de desarrollo de software. 9
1.4.2 Lenguaje de modelado (UML, Unified Modeling Language) 12
1.4.3 Herramienta CASE para la modelación del sistema 12
1.4.4 Tecnologías de desarrollo. 13
1.4.5 Entorno de Desarrollo Integrado (IDE) 15
1.4.6 Gestor de base de datos PostgreSQL 15
1.4.7 Herramienta para el control de versiones, Subversion. 16
1.4.8 Framework para el desarrollo Web 16
Capítulo 2: Descripción y análisis de la solución propuesta. **19**
2.1 Análisis de la investigación precedente 19
2.1.1 Requisitos funcionales a automatizar 22
2.1.2 Diagrama de casos de uso del sistema. 23
2.2 Descripción de los Casos de Uso del Sistema 24
2.3 Patrones Utilizados para el desarrollo de la aplicación 24
2.3.1 Patrones de diseño 25
2.3.2 Patrones de Arquitectura 28
2.4 Paquetes del diseño 28
2.5 Modelo de diseño 29
2.5.1 Descripción de las clases del diseño 29
2.5.2 Diagrama de clases del diseño para aplicaciones Web 47

2.5.3 Paquete del modelo. 51
2.5.4 Diagramas de interacción (secuencia) para Web....52
2.6 Prototipos de Interfaz 64
2.7 Diagrama de clases persistentes....65
2.8 Modelo de Datos....66
2.9 Modelo de despliegue....67
2.10 Mapa de navegación 68
2.11 Ejemplos de código 70
Conclusiones Generales 77
Recomendaciones: 78
Referencia Bibliográfica....79
Bibliografía 80
Glosario de Términos 82
Anexos 84

Índice de Figuras

Figura 1 RUP en dos dimensiones 10
Figura 2 Diagrama de CU del módulo Administración....24
Figura 3 Diagrama paquetes del diseño del módulo Administración....29
Figura 4 Diagrama de clases del diseño CU: Autenticar 48
Figura 5 Diagrama de clases del diseño CU: Gestionar IP 49
Figura 6 Diagrama de clases del diseño CU: Gestionar Asignación 50
Figura 7 Paquete del modelo....51
Figura 8 Diagrama de secuencia CU Autenticar.....52
Figura 9 Diagrama de secuencia CU Autenticar (Continuación). 53
Figura 10 Diagrama de secuencia CU Autenticar (Cambiar contraseña).....54
Figura 11 Diagrama de secuencia CU Gestionar IP (Insertar IP).....55
Figura 12 Diagrama de secuencia CU Gestionar IP (Buscar IP).....56
Figura 13 Diagrama de secuencia CU Gestionar IP (Editar IP).....57
Figura 14 Diagrama de secuencia CU Gestionar IP (Listar IPs). 58
Figura 15 Diagrama de secuencia CU Gestionar Asignación (Insertar Asignación)....59
Figura 16 Diagrama de secuencia CU Gestionar Asignación (Buscar Asignación)....60

Figura 17 Diagrama de secuencia CU Gestionar Asignación (Editar Asignación). 61
Figura 18 Diagrama de secuencia CU Gestionar Asignación (Listar Asignaciones). 62
Figura 19 Diagrama de secuencia CU Gestionar Asignación (Eliminar Asignación).....63
Figura 20 Prototipo de interfaz CU Autenticar: Autenticarse. 64
Figura 21 Prototipo de interfaz CU Autenticar: Seleccionar Ensayo, Rol y Sitio. 65
Figura 22 Diagrama de clases persistentes.....66
Figura 23 Modelo de datos del módulo Administración 67
Figura 24 Modelo de despliegue.....68

Índice de Tablas

Tabla 2.1 Descripción de la clase del diseño: adm_principal 30
Tabla 2.2 Descripción de la clase del diseño: adm_crudIpActions..... 32
Tabla 2.3 Descripción de la clase del diseño: adm_AsignacionesActions 34
Tabla 2.4 Descripción de la clase del diseño: FachadaAdmin 37
Tabla 2.5 Descripción de la clase del diseño: ControlUsuario..... 38
Tabla 2.6 Descripción de la clase del diseño: ControlSitio 38
Tabla 2.7 Descripción de la clase del iseño: ControlEC 39
Tabla 2.8 Descripción de la clase del diseño: ControlIp 40
Tabla 2.9 Descripción de la clase del diseño: ControlAsignarIp..... 42
Tabla 2.10 Descripción de la clase del diseño: myUser 46
Tabla 2.11 Descripción de la clase del diseño: UsuarioActivo 46

Introducción

El cáncer, es una de las enfermedades más peligrosas del mundo. Esta enfermedad trae como consecuencia la transformación de las células normales del organismo en otras que se comportan de forma muy peligrosa para el cuerpo humano.

Más del 11% de las casi 56 millones de muertes que se produjeron en el mundo en los últimos años fueron debido a tumores malignos. En muchos países, más de una cuarta parte de las muertes son atribuibles al cáncer. Según el *Informe mundial sobre el cáncer*, año 2003, el análisis mundial de la morbilidad más completo realizado hasta la fecha, la incidencia del cáncer podría aumentar en un 50% hasta el año 2020, en el que habría 15 millones de nuevos casos.

Según datos estadísticos de la Organización Mundial de la Salud (OMS), de los 58 millones de muertes que se registraron en el mundo en el año 2005, 7.6 millones (13%) se debieron al cáncer. Los que más contribuyen a la mortalidad son los cánceres de:

- *pulmón (1,3 millones de muertes anuales).*
- *estómago (casi 1 millón de muertes anuales).*
- *hígado (662 000 muertes anuales).*
- *colon (655 000 muertes anuales).*
- *mama (502 000 muertes anuales).* (1)

Esta enfermedad en Cuba es la segunda causa de muerte y en los últimos diez años se aprecia un incremento del 25 por ciento en su incidencia y mortalidad.

Un informe del Registro Nacional de Cáncer de Cuba revela que en el año 2005 fallecieron 18 959 personas, (3 832 más que en 1996). Dicho informe indica que la tasa de mortalidad por cáncer en los hombres es de 194 por cien mil habitantes, mientras que en las mujeres es de 144.2.

Con el objetivo de combatir diferentes enfermedades, se crea el 5 de diciembre de 1994 el Centro de Inmunología Molecular (CIM), que desde sus inicios ha venido trabajando en la búsqueda de nuevos productos para el diagnóstico y tratamiento del cáncer y enfermedades relacionadas con el sistema inmune.

Para la aprobación e introducción de estos productos en el mercado se realizan Ensayos Clínicos, investigación llevada a cabo en seres humanos con el propósito de evaluar la eficacia de una intervención médica o quirúrgica mediante un protocolo estrictamente controlado. Las variables manejadas en este tipo de investigación son recogidas en Cuadernos de Recogida de Datos (CRD),

los cuales constituyen un formulario diseñado para anotar las variables recogidas durante un Ensayo Clínico. Dentro de los procesos por los que transcurre esta investigación se destaca el de Manejo de Datos, que no es más que la transferencia de los datos registrados en los CRD a la base de datos.

Los Ensayos Clínicos llevan asociados una gran cantidad de documentación (datos primarios, imágenes, Cuadernos de Recogida de Datos, aprobaciones, modificaciones), necesarias para cumplir con las buenas prácticas clínicas exigidas por todas las Agencias regulatorias a nivel mundial. Esta documentación debe ser almacenada no menos de 15 años posterior al cierre del estudio, de modo que pueda ser inspeccionada en cualquier momento por las Agencias Regulatorias. Como un estimado conservador se podría decir que por cada paciente de un Ensayo Clínico se genera como mínimo 1000 datos diferentes, lo cual permite fácilmente estimar el gran volumen de información que se genera y maneja en esta actividad.

En el año 1989, anterior a la inauguración del Centro de Inmunología Molecular, la Institución contaba con el anticuerpo monoclonal Murino ior t3 para el tratamiento de las crisis de rechazo en el trasplante renal, con el que se realizó un Ensayo Clínico en el Instituto de Nefrología. En la actualidad se trabaja con 15 productos diferentes, que despliegan una gama de 52 Ensayos Clínicos, alguno de ellos multinacionales. Se realizó un pronóstico en el año 2005 de incluir a 2500 pacientes, además de mantener el tratamiento y seguimiento a más de 350 incluidos durante el 2004, de los pacientes que maneja el CIM se generan no menos de 2 millones 500 mil datos, de diversos tipos, desde información referida a mediciones cuantitativas del estado de un paciente (presión sanguínea, hemoglobina y muchos otros), hasta imágenes que caracterizan el tamaño del tumor, o información referida al proceso de intercambio de mensajes entre los investigadores que conducen los estudios. En este proceso están involucrados más de 500 hospitales en el país, y si bien es cierto que la mayoría de los Ensayos Clínicos están centralizados en las provincias: Ciudad de la Habana, Pinar del Río, Matanzas, Villa Clara, Camagüey, Holguín y Santiago de Cuba, los Ensayos Clínicos post registros, fases IV de fármaco-vigilancia se encuentran en marcha en 13 de las 14 provincias del país. (2)

Actualmente, toda la información referente a los Ensayos Clínicos en el CIM y en Cuba se encuentra en papel y la transmisión de la misma entre los hospitales y los centros promotores se realiza vía correo electrónico y telefónico. El proceso de monitorización de toda la información clínica (cuyo objetivo es controlar la calidad y uniformidad de los datos recogidos) se realiza mediante visitas periódicas a cada uno de los sitios que participan en esta actividad (que se refiere a todas las provincias involucradas).

Por la importancia que tiene la realización de Ensayos Clínicos en Cuba, la compra de un sistema que se encargue del manejo de datos de Ensayos Clínicos podría pensarse como una solución; debido a los altos precios impuestos en el mercado mundial, el Polo científico decide desarrollar un Sistema de Manejo de Datos de Ensayos Clínicos Cubano (SIMDECC) en conjunto con la Universidad de las Ciencias Informáticas (UCI), el cual se encargará de la gestión de toda la información generada por los CRD. Para el desarrollo del sistema se decide dividir el trabajo en 5 módulos principales; Administración, Diseño, Validación, Publicador y Monitoreo, enmarcando la presente investigación en el módulo Administración.

En la investigación precedente se realizó un análisis del módulo Administración con el fin de desarrollar una aplicación informática para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano, este proceso mostró como resultado 31 requisitos funcionales y 9 requisitos no funcionales, traducidos en casos de uso y prototipos no funcionales.

Dado que actualmente no existe un sistema que de solución a los requerimientos anteriores, se define el siguiente **problema científico**:

¿Cómo obtener un producto funcional a partir de los requerimientos identificados en el módulo Administración para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano?

Se plantea como **objeto de estudio**:

Los procesos de gestión de información para los Sistemas de Manejo de Datos de Ensayos Clínicos.

Teniendo como **campo de Acción**:

Los procesos de gestión de información del módulo Administración para los Sistemas de Manejo de Datos de Ensayos Clínicos.

Dado el **campo de acción** se define como **objetivo general**:

Desarrollar el diseño e implementación del módulo Administración para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.

Del cual se plantean los siguientes **objetivos específicos**:

- *Diseñar el módulo Administración para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.*
- *Implementar los componentes del módulo Administración para el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.*
- *Para el cumplimiento de los objetivos se proponen las siguientes tareas:*

- *Revisión de la investigación precedente para el módulo Administración.*
- *Estudio de aplicaciones Web para los Sistemas de Manejo de Datos de Ensayos Clínicos.*
- *Estudio y selección de las herramientas a utilizar para el diseño e implementación del módulo Administración.*
- *Realización de los diagramas de clases del diseño.*
- *Elaboración de los diagramas de interacción.*
- *Confección del diagrama de clases persistentes.*
- *Confección del Modelo de Datos.*
- *Realización del diagrama de despliegue.*
- *Realización del Mapa de Navegación.*
- *Implementación del módulo Administración.*

Estructura del trabajo:

- **Capítulo 1 Fundamentación Teórica:** En este capítulo se abordan aspectos generales sobre los Ensayos Clínicos, se hace alusión a las funcionalidades de algunas aplicaciones Web existentes a nivel mundial dedicadas a la realización de Sistemas de Manejo de Ensayos Clínicos. Se lleva a cabo una caracterización de las tecnologías y herramientas utilizadas para el desarrollo de la aplicación informática.
- **Capítulo 2 Descripción y análisis de la solución propuesta:** Este capítulo contiene una valoración crítica del análisis propuesto por los analistas, un análisis del diseño a desarrollar y la construcción de los artefactos inherentes al diseño. Consta de un análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados, una breve descripción de algunos de los patrones que existen para la construcción de un software, así como la construcción de los artefactos inherentes a la implementación y la descripción de los algoritmos no triviales a implementar.

Capítulo 1: Fundamentación Teórica.

Introducción

En este capítulo se abordan aspectos generales sobre los Ensayos Clínicos, se hace alusión a las funcionalidades de algunas aplicaciones Web existentes a nivel mundial dedicadas a la realización de Sistemas de Manejo de Ensayos Clínicos. Se lleva a cabo la caracterización de las tecnologías y herramientas utilizadas para el desarrollo de la aplicación informática.

1.1 Ensayos Clínicos

Desde hace muchos años la especie humana ha sido amenazada por diversas enfermedades, algunas de ellas mortales, como es el caso del cáncer; con el objetivo de combatir las mismas, científicos de Cuba y el resto del mundo se han dado a la tarea de crear productos químicos y convertirlos en medicamentos para tratamientos a seres humanos. Uno de los centros dedicados a esta labor es el CIM. Para la aplicación y exportación de estos medicamentos deben ser aprobados por las Agencias Regulatorias, para esto se realizan Ensayos Clínicos, estudio que permite a los médicos determinar si un nuevo tratamiento, medicamento o dispositivo contribuirá a prevenir, detectar o tratar una enfermedad, ayudan a los médicos a descubrir si estos nuevos tratamientos son inocuos y si son mejores que los actuales.

Se ha demostrado que los Ensayos Clínicos tienen una gran importancia en la actualidad, es la vía segura para que medicamentos, destinados a combatir el cáncer, sean lanzados al mercado sin correr el riesgo de que no cumplan con las normas requeridas para ser usados por una persona que padezca de esta enfermedad. A través de ellos se pueden prevenir enfermedades, se descubren nuevos agentes infecciosos o sencillamente se puede mejorar la calidad de vida del paciente. En el mundo existe un gran número de pacientes que padecen de cáncer que se han beneficiado con estos estudios.

Lo antes expuesto evidencia la necesidad de realizar un sistema que cumpla con las regulaciones implantadas para realizar Ensayos Clínicos, el CIM se propuso llevar a cabo esta tarea y con la colaboración de la UCI decidió desarrollar un Sistema de Manejo de Datos de Ensayos Clínicos Cubano. Este proyecto, actualmente en construcción, se divide en 5 módulos principales, Administración, Diseño, Validación, Publicador y Monitoreo, con el objetivo de realizar el módulo correspondiente a la administración de la aplicación, encargado específicamente de la gestión y

autenticación de usuarios, a la gestión de Ensayos Clínicos, Hospitales, Sitios, así como la gestión de los roles de cada uno de los usuarios y la dirección IP desde la que se conectan los mismos, se llevó a cabo una investigación anteriormente que se encargó del levantamiento del negocio y definición de los Requisitos Funcionales(RF) y no Funcionales(RNF) del sistema, descripción de los Casos de Uso (CU), el análisis de los CU identificados, así como la definición de metodologías y herramientas a utilizar, arrojando como resultado un conjunto de 31 RF y 9 RNF.

1.2 Módulo Administración de la aplicación.

Debido al muy complejo proceso de recogida de información de los Ensayos Clínicos, el sistema a desarrollar debe cumplir con algunos requisitos que serán propios de este, aunque en el mundo se han desarrollado diferentes sistemas que se encargan de la administración para el manejo de ensayos clínicos se decidió hacer un sistema propio, es decir, realizar el diseño e implementación del módulo Administración para el Sistema de Manejo de Ensayos Clínicos Cubano.
La administración debe garantizar:

- Un sistema de seguridad para usuarios, siendo la base fundamental para la capa de autenticación de la aplicación.
- La gestión de los Ensayos que se llevarán a cabo en cada unos de los sitios que intervienen en este proceso.
- La gestión de los usuarios, creando para ello los nombres de usuario y contraseñas así como el rol que desempeñan en el sistema y dentro de los Ensayos.
- La gestión de los sitios que pueden participar en los Ensayos, y que estarán distribuidos en hospitales.
- La gestión de la dirección IP por la cual tendrá acceso al sistema cada uno de los usuarios en el sitio en que se esté desarrollando un Ensayo.
- La gestión de asignación de roles y direcciones IP a un usuario desde un sitio en un ensayo específico.

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

Después de realizar varias investigaciones sobre cómo manejar el gran volumen de datos resultantes de los Ensayos Clínicos se encuentra que en el mundo se viene trabajando desde hace varios años en el desarrollo de Sistemas de Manejo de Ensayos Clínicos en versiones electrónicas. Desde 1997 se

ha trabajado en los llamados Ensayos Clínicos basados en Internet (Internet-Based Clinical Trials (IBCTs), que ofrecen la ventaja de poder trabajar con una muestra mayor de pacientes, con un costo reducido al acceso de los datos y rápida integración e informatización, posteriormente se reporta el uso de la tecnología de Internet para la adquisición de datos científicos y luego se publica no solo la adquisición de datos sino la planificación, ejecución y procesamiento de datos del Ensayo Clínico basado en Internet.

Con el objetivo de integrar al Polo a este avance científico, se ha indagado entre los distintos software que existen para el manejo y la colección de datos, entre los buscados se encuentran:

Pivotal

Sus desarrolladores se dedican al proceso de ensayos clínicos. Presentan un equipo de programadores expertos en el diseño e implementación de Ensayos Clínicos y programas de validación de ensayos clínicos. Utilizan sistemas como Clinical Trial Data Management System: Oracle Clinical.

Ofrecen servicios como:

- *Adaptación a los sistemas del cliente entrando en su intranet mediante una conexión por red privada segura y autentificada, con sistemas de emulación de terminal.*
- *Diseño de CRD en papel o electrónicos.*
- *Diseño e implementación de bases de datos de Ensayos Clínicos.*
- *Diseño y programación de rutinas de validación de datos con PL/SQL y Perl-SQL.*

Entrada de datos: doble entrada, con validación por tercera persona. (3)

CS - "Core System"

CS - "Core System" producido por la compañía eMedical en México dedicada al desarrollo de herramientas de software, propone una solución completa e inmediata a las necesidades en la administración de pacientes. Posee un amplio proceso de administración del sistema como por ejemplo:

1. Entrada al sistema.
2. Configurar el sistema:

a. Administración de perfiles.
b. Administración de asistentes.
c. Administración de catálogo.

3. Gestionar paciente: Insertar, editar, mostrar y eliminar paciente.
4. Ver Historial: ¿Cómo utilizar la información almacenada?, reportes, gráficas, etc.
5. Da la solución cliente-servidor para implementar sistemas a altos costos.

OpenClinica

OpenClinica es un Software de Manejo de Datos Clínicos (Clinical Data Management Software (CDMS)). Está pensado para administrar estudios clínicos, por lo que se apega a los estándares internacionales existentes. El producto está diseñado para el manejo de datos múltiples y estudios clínicos de diferentes ramas de la investigación.

Está basado en la plataforma de desarrollo de Java (J2EE), utilizando como base de datos PostgreSQL 8.x, pudiendo correr tanto en servidores Linux como Windows.

Dentro de los requisitos del sistema se encuentran:

- Gestionar estudio.
- Manejo de datos.
- Administración del sistema.

Luego del estudio de las soluciones que proponen estos software para los Sistemas de manejo de Ensayos Clínicos se llega a la conclusión de que:

Utilizan varias formas interesantes a la hora de administrar un sistema; pero no se adecuan de forma correcta a los procesos de negocio de los centros cubanos, ejemplo, la necesidad de que cada usuario tenga asignado un rol y más importante aún, una dirección IP desde la cual se puede conectar, que esta dirección IP pertenezca con anterioridad a un sitio que sería entonces la ubicación del usuario en un hospital, se debe tener control estricto de los ensayos que hayan sido habilitados o deshabilitados, lo cual solo puede hacer el administrador de la aplicación con autorización de alguna persona con facultades para ello, así como habilitar o deshabilitar a los usuarios en el sistema y los roles que desempeñan en el mismo; todos estos requisitos necesarios para la administración de la aplicación conjuntamente con otros para la gestión de los datos manejados por el administrador, es decir, gestión de sitios, hospitales, Ensayos Clínicos y usuarios deben ser resueltos al finalizar este producto.

Además de lo antes expuesto estas aplicaciones poseen altos precios para su adquisición, gastos en los que el país no puede incurrir, excluyendo a OpenClinica que es libre, aunque este no permite el control estricto sobre los datos que se recogen y almacenan debido a que los servidores de base de datos estarían fuera de Cuba, por todas estas razones se decidió implementar en Cuba con ayuda de la UCI un Sistema de Manejo de Datos de Ensayos Clínicos Cubano.

1.4 Fundamentación de las tecnologías y herramientas a utilizar

En este epígrafe se analizarán algunas de las tecnologías y herramientas a utilizar para la construcción de un sistema que cumpla con todos los requisitos necesarios; en estudios anteriores se hicieron comparaciones entre estas herramientas y tecnologías llegando a hacerse una propuesta final para la utilización de una u otra, en el presente epígrafe se argumentará el ¿por qué? de las herramientas y tecnologías que se utilizarán para el desarrollo de la aplicación.

1.4.1 Metodología de desarrollo de software.

Actualmente existe diversidad en cuanto a metodologías para el desarrollo de un software, citando algunos ejemplos podemos mencionar Extreme Programming (XP), Microsoft Solution Framework (MSF) y Racional Unified Process (RUP), cada una de ellas responde a diferentes clasificaciones, ágiles, flexibles y robustas.

Una metodología representa un conjunto de actividades a tener en cuenta en el marco de la Ingeniería de Software con el objetivo de desarrollar un software de alta calidad que cumpla con las necesidades del usuario (cliente).

En la investigación precedente se decidió utilizar como guía para el proceso de desarrollo de la solución informática, Rational Unified Process (RUP).

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

RUP representa la metodología más rigurosa y la que mejor se adapta a la realización de proyectos grandes en cuanto a tamaño, duración y a sistemas Orientados a Objetos. Cuenta con diferentes elementos de planificación (plan de desarrollo, plan de iteración, plan de calidad, etc.) con los que se controla el desarrollo del software, es iterativo e incremental lo que ayuda a una mejor realización del software y aplicación de nuevas versiones.

RUP divide el proceso en 4 fases de desarrollo:

- ***Inicio:*** *Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto.*
- ***Elaboración:*** *Se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.*
- ***Construcción:*** *Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.*
- ***Transición:*** *Se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.* (4)

RUP define 9 flujos de trabajo los cuales tienen lugar en cada uno de las fases de desarrollo, algunos de estos flujos de trabajo tienen actividad en todas las fases en mayor o menor medida. La siguiente figura muestra a RUP en dos dimensiones y evidencia lo antes dicho.

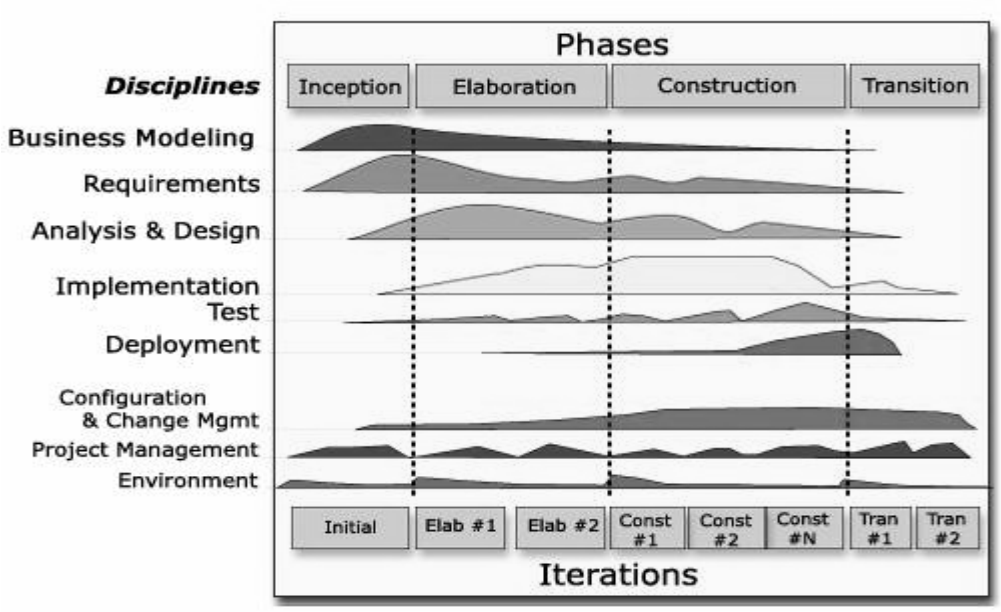

Figura 1 RUP en dos dimensiones

Dentro de los flujos de trabajo definidos por RUP los desarrolladores desempeñan diferentes roles y artefactos, los que se implementarán en los flujos de trabajo de análisis y diseño e implementación para el desarrollo de la presente aplicación son:

1- Análisis y diseño

Roles

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

Artefactos

- Realización de los Casos de Uso del Sistema
- Clases del diseño
- Paquetes de diseño
- Diagrama de despliegue
- Prototipo de interfaz de usuario
- Mapa de navegación
- Modelo de datos

2- Implementación

Roles

- Programador

Artefactos

- Prototipo ejecutable

1.4.2 Lenguaje de modelado (UML, Unified Modeling Language)

El Lenguaje Unificado de Maquetación (UML) ayuda a especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no determina qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente le ayuda a visualizar el diseño y a hacerlo más accesible para otros. UML representa un estándar de descripción de esquemas de software, está diseñado para su uso con software orientado a objetos.

1.4.3 Herramienta CASE para la modelación del sistema

Las Herramientas CASE (*Computer Aided Software Engineering*, del español "Ingeniería de Software Asistida por Ordenador") representan diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como la realización del diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras.

Con vista a la utilización de una herramienta CASE para la modelación del sistema, la investigación precedente tomó la decisión de utilizar el Rational Rose. Debido al carácter propietario de la misma y otros aspectos que serán expuestos posteriormente, la presente investigación se ha decidido por el uso del Visual Paradigm.

Visual Paradigm

Este producto está creado no para el uso personal, de la familia o de la casa; se pensó exclusivamente para uso profesional. Visual Paradigm es una herramienta UML que soporta la última notación UML 2.0, generación de código, importación desde Rational Rose, soporta el ciclo de vida completo del desarrollo de software y ofrece:

- *Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.*
- *Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.*
- *Capacidades de ingeniería directa (versión profesional) e inversa.*
- *Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.*

- *Disponibilidad de múltiples versiones, para cada necesidad.*
- *Disponibilidad de integrarse en los principales IDE.*
- *Disponibilidad en múltiples plataformas.* (5)

1.4.4 Tecnologías de desarrollo.

Lenguaje de programación Web, PHP5

El lenguaje de programación permite crear las funcionalidades que sustentan a cada una de las interfaces gráficas accedidas por el usuario y así llevar a cabo las actividades de gestión del sistema. PHP5 es uno de los mejores lenguajes para la creación de páginas Web dinámicas, es orientado a objetos, haciendo uso así de todas las ventajas que este paradigma nos ofrece. Obtiene su consolidación en cuanto al paradigma orientado a objetos a partir de esta versión, motivo fundamental por el cual se decidió trabajar sobre PHP5, permite lectura de archivos XML de forma sencilla, utilización de bases de datos como PostgreSQL o la implementación de servicios Web. La portabilidad de PHP le permite migrar a un sistema sin problemas y sin cambios desde Linux o Unix a plataforma Windows, y viceversa.

En investigaciones precedentes resultó como lenguaje de programación para la implementación del sistema PHP; luego de un análisis a fondo del mismo, el presente trabajo decide utilizar PHP en su versión 5 por las características antes mencionadas, orientado a objetos, utilización de bases de datos como PostgreSQL, muy portable y fácil de entender e implementar.

Servidor Web Apache

Los servidores Web se utilizan para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Es software libre, actualmente el servidor Web más utilizado en el mundo. Tiene gran potencia, fiabilidad y sencillez de configuración. Puede funcionar en la más amplia variedad de plataformas y entornos.

Apache 2.2 la versión a utilizar extiende su diseño modular hasta las funciones más básicas de un servidor Web. El servidor viene con una serie de Módulos de Multiprocesamiento que son responsables de conectar con los puertos de red de la máquina, aceptar las peticiones, y generar los

procesos hijos que se encargan de servirlas. La licencia Apache es descendiente de las licencias BSD, no es GPL. Esta licencia permite hacer lo que quieras con el código fuente.

Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto. Tiene una alta configurabilidad en la creación y gestión de Logs. Permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (6)

Se escoge Apache por las características antes mencionadas, además de ser multiplataforma, lo que lo hace prácticamente universal. Además es una tecnología gratuita de código fuente abierta. Esto le da gran transparencia de manera que si se quiere ver que es lo que se está instalando como servidor se puede hacer sin ningún problema.

Editor para los prototipos de interfaz, Kompozer

En la investigación precedente se decidió utilizar como herramienta para editar los prototipos de interfaz el Dreamweaver8, debido a que no es de uso libre, se decide hacer un cambio de herramienta y utilizar el Kompozer como editor de código HTML, esta decisión se tomó debido a que es una herramienta de creación de páginas Web en la que no se requiere ningún conocimiento de HTML por parte de los usuarios. Es un software gratuito y tiene muchas características parecidas al Dreamweaver.

- *Opciones especiales para la inserción de imágenes, tablas y formularios.*
- *Generador automático de tablas de contenido basado en los niveles de encabezado.*
- *Editor CSS avanzado, con capacidad de crear y usar tanto archivos CSS externos como hojas incrustadas en el archivo HTML mediante etiquetas <style>.*
- *Posibilidad de definir y usar plantillas.*
- *Admite etiquetas PHP sin alterar su contenido.*
- *Limpiador de código HTML.*
- *Enlace directo con el validador HTML de W3C.*
- *Ayuda completa incorporada en el programa.* (7)

1.4.5 Entorno de Desarrollo Integrado (IDE)

Programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Eclipse PDT (PHP Development Tool)

En investigaciones precedentes se decidió usar como entorno de desarrollo para la construcción de la aplicación, Zend Studio, debido a la necesidad de utilizar un IDE que se integre fácilmente al framework Symfony y que tenga un cliente de Subversion que facilite el control de versiones, se realizó un estudio más exhaustivo de la propuesta anterior y se llegó a la conclusión de utilizar como entorno de desarrollo Eclipse.

Eclipse como plataforma de desarrollo de código abierto es usado para múltiples aplicaciones basadas en framework así como para la construcción de entornos de desarrollo (IDEs) que puedan ser utilizados para la construcción de aplicaciones Web, editor sensible al contexto. Provee asistencia y autocompletado de código, además ofrece un autocompletado de código para el desarrollo con el framework Symfony, Soporte para el debug incremental del código de PHP. Es un IDE de desarrollo multiplataforma ejecutándose en varios sistemas operativos como Windows y Linux.

1.4.6 Gestor de base de datos PostgreSQL

Los sistemas de gestión de base de datos se utilizan para el almacenamiento de información donde posteriormente se puede acceder a los datos de forma rápida y estructurada, aseguran la integridad, confidencialidad y seguridad de los datos almacenados.

PostgreSQL es un gestor de código abierto y es uno de los mejores gestores de base de datos del mundo por su rapidez, capacidad de almacenamiento, robustez. Posee características muy propias de él que otros sistemas no tienen bien definidos como disparadores, herencia, vistas y lenguajes procedulares.

*Postgre es Full **ACID** compliant (Atomicity, Consistency, Isolation and Durability)*

- **Atomicity (Atomicidad (Indivisible))** Es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

- **Consistency (Consistencia)** Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- **Isolation (Aislamiento)** Es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
- **Durability (Durabilidad)** Es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. (8)

Debido a las características antes planteadas, adicionándole que PostgreSQL corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, tiene documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios y consta con comunidades muy activas, varias de ellas en castellano, se ha decidido desde investigaciones anteriores utilizarlo como gestor de base de datos.

1.4.7 Herramienta para el control de versiones, Subversion.

Las herramientas de control de versiones se utilizan para gestionar los cambios que se realizan en el producto en desarrollo, lo que hace que facilite la administración de las versiones del producto.

En investigaciones precedentes se realizó una comparación entre Concurrent Version System (CVS) y Subversion y se decidió aplicar este último para el desarrollo de la aplicación debido a que:

- Es un software para el control de versiones diseñado para reemplazar a CVS.
- Tiene una fuerte integración con apache lo que posibilita definir controles de acceso avanzados y navegación vía Web para consultar el depósito de archivos.
- Posee alta transparencia al eliminar y cambiar nombres de archivos sin intervención manual como lo haría CVS,
- Sistemas de acceso URL.
- Puede Trabajar sin conexión permanente.

1.4.8 Framework para el desarrollo Web

Framework, estructura de soporte compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

Los framework son diseñados con el intento de facilitar el desarrollo de software para los diseñadores y programadores. Sus objetivos principales son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Por tanto se puede decir que un framework para Web es un conjunto de componentes (por ejemplo clases en java o php, descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. (9)

A continuación se hace referencia a las características más importantes del framework symfony y al final se justifica su utilización.

Symfony

*Symfony es un framework desarrollado completamente en PHP5, es compatible con la mayoría de los gestores de base de datos como son MySQL, PostgreSQL, Oracle y SQL Server de Microsoft, se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.*

Es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web utilizando para esto el patrón de arquitectura Modelo Vista Controlador (MVC). Además proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. (10)

¿Por qué utilizar Symfony?

En la presente investigación se ha decidido utilizar como framework de desarrollo Symfony debido a su robustez, se adapta perfectamente al sistema operativo con que se trabaja (GNU/Linux), al lenguaje que se utilizará (PHP5), así como al gestor de base de datos (PostgreSQL), una de las características que lo diferencia de otros framework es la amplia documentación en español que alrededor del mismo se ha desarrollado, su evolución está en constante desarrollo por parte de sus desarrolladores y la comunidad que se ha creado a nivel internacional, teniendo una característica casi única ya que las compañías o empresas, así como la comunidad de desarrolladores extendida por todo el mundo, que lo utilizan, como por ejemplo Yahoo, hacen un reporte sistemático a los creadores del framework

donde se exponen posibles actualizaciones o cambios necesarios para el framework a través de las experiencias adquiridas, otra característica que lo distingue es que no trata de reinventar la rueda, reutiliza componentes y librerías externas de muy buena calidad y que ya han sido ampliamente probadas. El Sistema de Manejo de Datos de Ensayos Clínicos Cubano está dividido en módulos, característica que lo hace muy complejo, Symfony permitirá la integración de cada uno de estos módulos, así como la seguridad entre los mismos.

Conclusiones

El presente capítulo recogió el resultado de todo un análisis de los conceptos fundamentales relacionados con los Ensayos Clínicos. Se detallaron las condiciones y problemas que rodean el objeto de estudio y el flujo actual de los procesos haciendo una breve referencia a algunos de los sistemas automatizados que gestionan Sistemas de Manejo de Ensayos Clínicos, concluyendo que el desarrollo en Cuba de un sistema que se encargue del manejo de ensayos clínicos es la solución más factible debido a los requerimientos específicos que este presenta para la administración de la aplicación, Se llevaron a cabo estudios sobre posibles tecnologías y herramientas a utilizar para el desarrollo de la aplicación, Apache como servidor Web, PHP5 como lenguaje de programación, Symfony como marco de trabajo(framework), Eclipse como IDE de desarrollo, Kompozer y como metodología de desarrollo RUP.

Capítulo 2: Descripción y análisis de la solución propuesta.

Introducción

Este capítulo contiene una valoración crítica del análisis propuesto por los analistas, un análisis del diseño a desarrollar y la construcción de los artefactos inherentes al diseño. Consta de un análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados, una breve descripción de algunos de los patrones que existen para la construcción de un software, así como la construcción de los artefactos inherentes a la implementación y la descripción de los algoritmos no triviales a implementar.

2.1 Análisis de la investigación precedente

En la investigación precedente se realizó un análisis, obteniéndose los siguientes requisitos no funcionales, variando algunos por el cambio al sistema operativo Linux:

1- RNF de apariencia o interfaz externa

- Las páginas no tendrán muchas imágenes y poseerán pocos colores.
- Las páginas principales tendrán información que servirá de guía al usuario.
- Cada rol tendrá una interfaz diferente con las funciones que le corresponden.
- Se hará uso de simbología mediante íconos para indicar el estado de los elementos utilizados en el diseño.

2- RNF de usabilidad

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

- El sistema contendrá un manual de usuario, que será usado como ayuda para el trabajo con la aplicación.

3- RNF de Soporte

- Una vez terminada la aplicación se instalará en el CIM para realizar pruebas piloto del software y pruebas de despliegue.
- Una vez aprobada la aplicación y lista para comenzar su ejecución se instalará en los centros del polo científico y hospitales donde se realice la conducción de Ensayos Clínicos.
- Cada cierto tiempo previsto por los administradores de la aplicación se realizará el mantenimiento del software.
- La capacidad de mantenimiento deberá ser adecuada, documentando cuidadosamente todas las actividades realizadas en el desarrollo de la aplicación informática.
- Se debe facilitar la posibilidad de actualización y cambios sobre la base de un diseño escalable y robusto.

4- RNF de Seguridad

- El acceso a cualquier manipulación del sistema, tanto entrada como análisis de datos debe estar sometido a un proceso de autentificación del usuario donde será especificado el usuario y la contraseña pasando de ahí a otro nivel donde escogerá el ensayo, el rol y el sitio para acceder al sistema.
- Las contraseñas deberán tener más de 7 caracteres de longitud y tener una fortaleza media.
- Los usuarios estarán obligados a cambiar la contraseña cada 60 días como máximo.
- Cada usuario tendrá asignado uno o varios roles en el sistema. Cada rol definido tendrá niveles de acceso al Software.
- Solo podrán acceder a la aplicación los clientes a través de direcciones IP específicas bien controladas.
- Todo cambio o modificación en el sistema debe ser atribuible a un usuario particular según su autenticación.
- Se debe garantizar comunicaciones seguras entre los clientes y el servidor, encriptando todo el tráfico de información usando llaves negociadas, algoritmos y protocolos.

5- RNF de Software

- Para la instalación de la aplicación se debe disponer del sistema operativo Windows o GNU/Linux.
- En las computadoras de los clientes también deberán existir las mismas restricciones de los sistemas operativos incluyendo un navegador asociado al sistema operativo finalmente escogido para la visualización de las interfaces Web.

6- RNF de Hardware

- Para el funcionamiento de la aplicación son imprescindibles un navegador y conectividad.
- El Servidor Web debe tener alta disponibilidad y un rendimiento adecuado, garantizado al menos un procesador Dual Intel Xeon 3 GHz o similar y RAM suficiente (4 GB a 8 GB).
- Los servidores de almacenamiento de datos deben tener de 1 a 3 TB disponibles pues el volumen de información es bastante grande y perdura en el tiempo hasta 15 años.

7- RNF Restricciones en el Diseño y la Implementación

- El diseño e implementación de la aplicación será basado en la Metodología RUP haciendo uso del lenguaje de modelado UML.
- Se usará como herramienta CASE Visual Paradigm para el modelado de los artefactos que se generan en cada uno de los flujos de trabajo definidos por RUP.
- Para el diseño de las interfaces se utilizará Kompozer.
- Como plataforma de desarrollo integrado se utilizará Eclipse PDT (PHP Development Tool).
- Como framework de desarrollo Web se utilizará Symfony.
- Se usará como lenguaje de programación PHP5.
- Como gestor de base de datos PostgreSQL.
- Podrán ser utilizados varios estándares como HTTP, HTML, XML, SOAP, UDDI.

8- RNF de Extensibilidad

- Se debe lograr un diseño adaptable, con la capacidad de poder soportar funcionalidades adicionales o modificar las funcionalidades existentes sin impactar el resto de los requerimientos contemplados en el sistema.

9- RNF de Disponibilidad

- Se garantizará que la aplicación se mantenga funcionando las 24 horas del día y los siete días de la semana con el menor tiempo posible de recuperación de fallos.
- El servidor de aplicación debe soportar un aumento de usuarios concurrentes por minuto de 1 a 400.

2.1.1 Requisitos funcionales a automatizar

- R1 Autenticar Usuario.
- R2 Cambiar contraseña.
- R3 Registrar sitio.
- R4 Modificar datos del sitio.
- R5 Eliminar sitio.
- R6 Mostrar listado de los sitios.
- R7 Buscar sitio.
- R8 Registrar hospital.
- R9 Modificar hospital.
- R10 Eliminar hospital.
- R11 Mostrar listado de los hospitales.
- R12 Buscar hospitales.
- R13 Registrar usuario.
- R14 Modificar datos del usuario.
- R15 Eliminar un usuario.
- R16 Mostrar listado de los usuarios.
- R17 Buscar usuario.

- R18 Asignar rol y direcciones IP a un usuario.
- R19 Modificar asignación de rol y direcciones IP a usuarios.
- R20 Eliminar asignación de rol y direcciones IP a usuarios.
- R21 Mostrar listado de asignaciones de rol y direcciones IP a usuarios.
- R22 Buscar asignación de rol y direcciones IP a usuarios.
- R23 Registrar Ensayo Clínico.
- R24 Modificar datos del Ensayo Clínico.
- R25 Mostrar listado de los Ensayos Clínicos.
- R26 Buscar Ensayo Clínico.
- R27 Eliminar Ensayo Clínico.
- R28 Asignar dirección IP y/o rango a un sitio.
- R29 Mostrar dirección IP asignados.
- R30 Modificar datos de una dirección IP asignada.
- R31 Buscar dirección IP asignada.

2.1.2 Diagrama de casos de uso del sistema.

Un diagrama de casos de uso representa gráficamente parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones, muestra los distintos requisitos funcionales que se esperan de una aplicación y cómo se relacionan con los usuarios u otras aplicaciones.

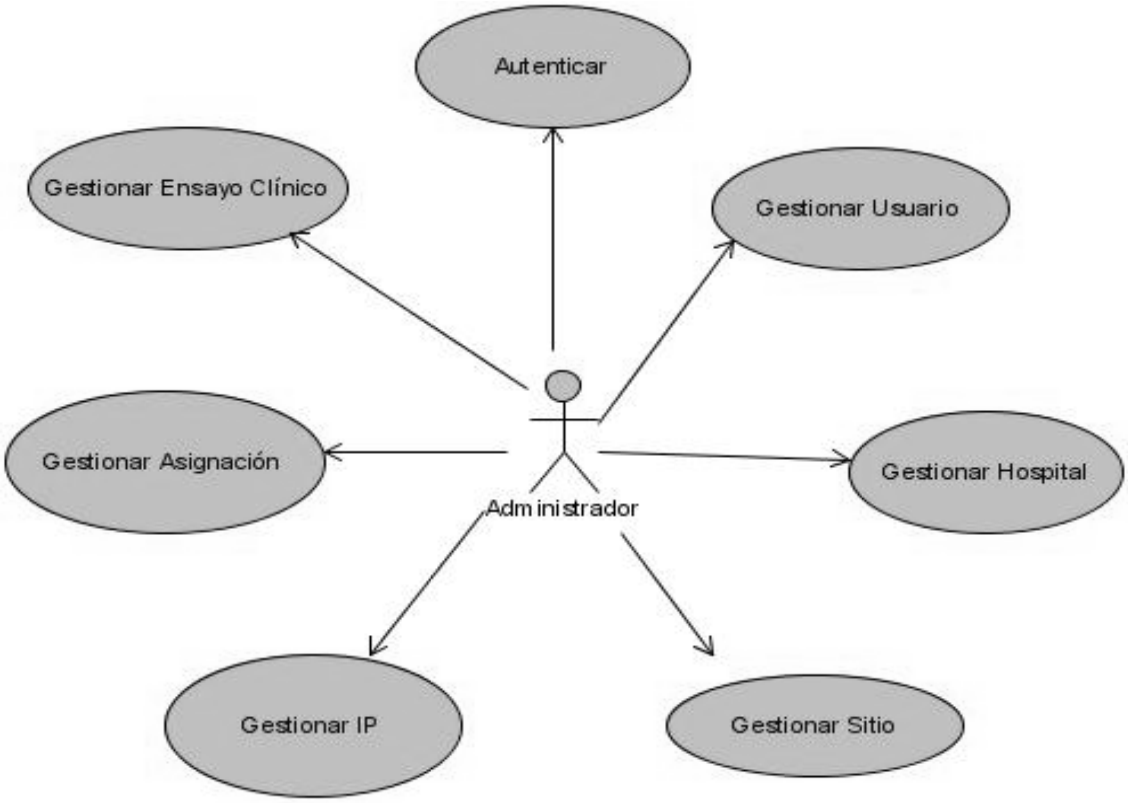

Figura 2 Diagrama de CU del módulo Administración

2.2 Descripción de los Casos de Uso del Sistema.

No solo con los diagramas de casos de uso se logra entender plenamente las funcionalidades que están asociadas a cada uno de ellos, con este objetivo se lleva a cabo las descripciones de casos de uso que no son más que reseñas textuales del caso de uso. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el mismo. Para realizar un análisis exhaustivo de los casos de uso a través de su descripción, referirse al expediente de proyecto donde encontrará la descripción extendida de cada uno de los CU.

2.3 Patrones Utilizados para el desarrollo de la aplicación.

Un patrón representa una descripción detallada de una solución adecuada a un problema concreto, es un modelo a seguir para la construcción de un software. Los patrones conducen a arquitecturas

simples y comprensibles, los mismos ayudan al desarrollador a seguir un modelo orientado a objeto ya que una arquitectura orientada a objeto está llena de patrones.

2.3.1 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos donde existe la comunicación entre los mismos para resolver un problema de diseño en algún contexto.

Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

Un patrón de diseño es:

- *Una solución estándar para un problema común de programación.*
- *Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.*
- *Un proyecto o estructura de implementación que logra una finalidad determinada.*
- *Un lenguaje de programación de alto nivel.*
- *Una manera más práctica de describir ciertos aspectos de la organización de un programa.*
- *Conexiones entre componentes de programas.*
- *La forma de un diagrama de objeto o de un modelo de objeto.* (11)

Patrones Gof

Patrón Decorator:

Define un layout que no es más que una plantilla global que almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página, el contenido de la plantilla se integra en el layout, o si se mira desde otro punto de vista el layout decora la plantilla.

Patrón Singlenton:

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.* (12)

Patrón Facade (Fachada):

Este patrón se basa en proporcionar al programador una clase sencilla con un grupo de métodos, uno para cada operación permitida y de modo que sean estos métodos los que internamente hagan las operaciones con el fin de que otros módulos lo puedan utilizar sin necesidad de repetir el código ya implementado, o sea sería una clase que servirá de información a otros módulos.

Patrón Front-Controller

Este patrón obliga a que todas las peticiones hechas a la aplicación pasen por un controlador.

- El controlador proporciona un punto de entrada único que controla y gestiona las peticiones Web realizadas por los clientes.
- Teniendo este único punto de entrada se evita tener que repetir la misma lógica de control en todos los archivos.

Patrones de asignación de responsabilidades (GRASP).

Patrón Experto:

Asignar una responsabilidad al experto en la información, se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide, esto provee un bajo nivel de acoplamiento.

Patrón Creador:

La clase ‘B’ crea las instancias de ‘A’ si:

- ‘B’ agrega los objetos de ‘A’
- ‘B’ contiene los objetos de ‘A’
- ‘B’ registra las instancias de los objetos de ‘A’.
- ‘B’ tiene los datos de inicialización que serán enviados a ‘A’ cuando este objeto sea creado.

Patrón Controlador:

Asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase que represente alguna de las siguientes opciones:

- El sistema global.
- La empresa u organización global.
- Algo activo en el mundo real que pueda participar en la tarea.
- Un manejador artificial de todos los eventos del sistema de un caso de uso (controlador de casos de uso).

Alta Cohesión:

Asigna una responsabilidad de modo que la cohesión siga siendo alta.

Patrón Bajo Acoplamiento:

Una clase con bajo acoplamiento no depende de “muchas otras” clases. Las clases con alto acoplamiento recurren a muchas otras y no es conveniente, son más difíciles de mantener, entender y reutilizar.

Polimorfismo:

Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán mediante operaciones polimórficas a los tipos en el que el comportamiento presenta variantes.

Indirección:

Se asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, y estos no terminen directamente acoplados. El intermediario crea una Indirección entre el resto de los componentes o servicios.

Fabricación Pura:

Asignar un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema para dar soporte a una alta cohesión, un bajo acoplamiento y reutilización.

No hables con Extraños:

Se asigna la responsabilidad a un objeto directo del cliente para que colabore con un objeto indirecto, de modo que el cliente no necesite saber nada del objeto indirecto.

2.3.2 Patrones de Arquitectura.

Un patrón de arquitectura describe un problema particular y recurrente del diseño, propone una solución o una vía de desarrollar cierto producto que surge en un contexto específico, y presenta un esquema genérico y probado de su solución.

Patrón Modelo Vista Controlador:

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. La principal ventaja de esta separación reside en la facilidad para realizar cambios en la aplicación puesto que cuando realizamos un cambio de base de datos, programación o interfaz de usuario solo tocaremos uno de los componentes, podemos modificar los componentes sin conocer cómo funcionan los otros.

- **Modelo:** Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos) para almacenar los datos.

Los patrones utilizados para el desarrollo de la aplicación fueron los de diseño, GRASP, es decir, asignación de responsabilidades, como patrones de diseño Gof se utilizaron Decorator, Front-Controller, Facade y Singleton que son utilizados por el framework Symfony, también se utilizó el patrón de arquitectura Modelo Vista Controlador que separa el desarrollo de la aplicación en 3 capas facilitando la realización de cambios en la aplicación.

2.4 Paquetes del diseño

Un paquete de diseño es una colección de clases relaciones, realizaciones de CU, diagramas y otros paquetes, es usado para estructurar el modelo de diseño mediante su división en partes más pequeñas. Son usados para agrupar elementos del modelo de diseño relacionados con propósitos organizacionales y frecuentemente para administración de configuración. (13)

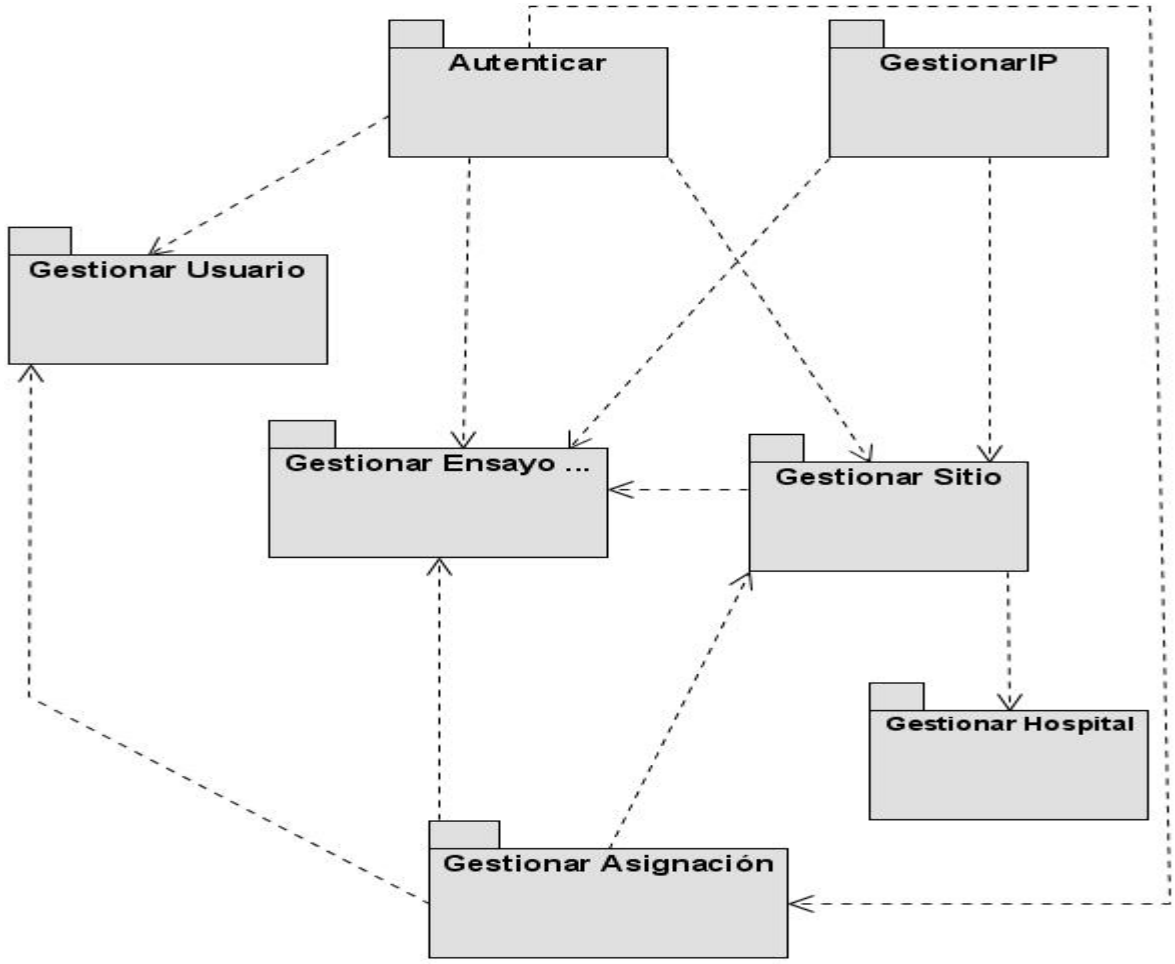

Figura 3 Diagrama paquetes del diseño del módulo Administración

2.5 Modelo de diseño

De acuerdo al diagrama de caso de uso del sistema obtenido en la investigación precedente se procede a la realización del modelo de diseño, es una representación de lo que se va a construir, aquí se modela el sistema de manera que soporte todos los requisitos, tanto funcionales como no funcionales, creándose así una entrada apropiada para las actividades de implementación.

2.5.1 Descripción de las clases del diseño

A continuación se describen las clases utilizadas en la modelación de los diagramas de diseño correspondientes a los casos de uso Autenticarse, Gestionar Asignación y Gestionar IP, proporcionando así una mayor comprensión de estas clases, se hace referencia a las dependencias

con otras y se describen los atributos de cada una y que hacen sus funciones. Las descripciones de los CU restantes pueden ser encontradas en el anexo del documento.

Tabla 2.1 Descripción de la clase del diseño: adm_principal

Nombre: adm_principal	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Autenticar, llama a la Clase FachadaAdmin, ControlUsuario y myUser para utilizar algunas de sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeAutenticar()	Muestra mensaje de error cuando se desea acceder a una zona sin estar autenticado y en caso de estarlo pero venció la sesión por inactividad.
executeErrorPrivilegios()	Captura por URL una variable y según sus datos muestra diferentes mensajes cuando hay algún error al loguearse.
executeErrorRol()	Muestra mensaje cuando se loguea e intenta acceder a una página con un rol sin privilegios para hacerlo.
executeLogIn()	Llama a las funciones verificar_autenticar() de la clase FachadaAdmin y signIn() de la clase myUser y en caso de que no haya errores al loquearse le da acceso al usuario al sistema, sino lanza un excepción, también llama a la función CalcularFecha() de la clase ControlUsuario para determinar el tiempo de caducidad de la contraseña del usuario logueado.
handleErrorLogIn()	Redirecciona a la página principal en caso de dejar campos vacios al loquearse.
executeLogOut()	Llama a la función signOut de la clase myUser() le retira las credenciales de acceso al sistema y redirecciona a la página principal.
executeAdmPrincipal()	Página principal para el rol de administrador.
executeCambiarContraseña()	Muestra utilizando una función AJAX un formulario para que el usuario pueda cambiar su contraseña.
executeCambiar()	Llama a la función Buscar_usuario_cont() de la clase ControlUsuario que busca un usuario con un id y una contraseña

	específico y crea un objeto usuario validando a la vez que la contraseña anterior esté correcta y muestra en la página principal una notificación haciendo saber si se pudo o no cambiar la contraseña.
executeDireccionar()	Según el usuario logueado al sistema llama a la función verificarRolHabilitado() de la clase FachadaAdmin, verifica que su rol esté habilitado en el sistema, en caso positivo, redirecciona a las páginas a las que tenga acceso.

executeUpdate()	Llama a la función DevolverPorPK() de la clase ControlSitio, modifica los datos y redirecciona a la página donde muestra los datos del sitio insertado.
executeDelete()	Verifica a través de la función VerificarSiHablilitado(), de la clase ControlEC, si el ensayo al que pertenece el hospital está habilitado o no, si lo está, lanza una excepción notificando que el ensayo está actualmente en uso y no se pueden eliminar los datos que pertenezcan a él, en caso contrario llama a la función DevolverPorPK() de la clase ControlSitio, elimina los datos y redirecciona a la página donde se muestra la lista de sitios que existen en la base de datos.
executeBuscar()	Función que crea un arreglo con las provincias de Cuba y muestra en la Client page correspondiente un formulario para hacer una búsqueda avanzada de sitios a criterio del administrador.
executeMostrarSitio()	Crea objetos ensayo y hospital y llama a la función MostrarSitios() de la clase ControlSitio y muestra en la Client Page correspondiente a executeBuscar(), utilizando AJAX, los sitios de acuerdo al criterio seleccionado por el administrador.
executeAutocompletar()	Función para completar una palabra o frase en un edit según las letras que se conozcan de la misma, en este caso para el nombre de un sitio, de un hospital o de un Ensayo Clínico.

Tabla 2.2 Descripción de la clase del diseño: adm_crudIpActions

Nombre: adm_crudIpActions	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Gestionar IP, llama a la Clase ControlIp, ControlEC y ControlSitio para utilizar sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de direcciones IP existentes en la base de datos.
executeList()	Llama a la función getIpPager() de la clase ControlIp y muestra en la Client Page correspondiente la lista de direcciones IP registrados en la base de datos.
executeShow()	Llama a la función DevolverPorPK() de la clase ControlIp y muestra en la Client Page correspondiente los datos de una dirección IP seleccionada por el administrador; pero no puede modificarlos.
executeInsert()	Llama a la función Buscar_Habilitados() de la clase ControlEC y BuscarSitioSinIP() de la clase ControlSitio, de donde obtiene los ensayos que existen y los sitios que aun no se le hayan asignado direcciones IP y luego muestra en la Client Page correspondiente un formulario para insertar en la base de datos la dirección IP asignada..
executeInsertar()	Llama a las funciones a SitioPerteneceHospital() y DameEnsayo() de la misma clase para saber a que ensayo se está asignando la dirección IP y a que hospital, en caso de no haber errores inserta los datos y redirecciona a la página donde se muestran todos las direcciones IP que existen en la base de datos sino lanza una excepción, según la acción del usuario puede llamar o no a Eliminar_ip() o Eliminar_rango() de la misma clase.
executeEdit()	Función que crea un objeto IP a través de la función DevolverPorPK() de la clase ControlIp y muestra en la Client Page

	correspondiente los datos de una dirección IP seleccionada por el administrador con opción de modificarlos.
executeUpdate()	Llama a las funciones validar_ip(), validar_rango() en ControlIP, modifica los datos y redirecciona a la página donde muestra los datos de la dirección IP modificada según la acción del usuario puede llamar a Eliminar_ip() o Eliminar_rango() de la misma clase.
executeDelete()	Llama a la función DevolverPorPK() de la clase ControlIp, elimina los datos y redirecciona a la página donde se muestra la lista de direcciones IP que existe en la base de datos.
executeBuscar()	Muestra en la Client Page correspondiente un formulario para hacer una búsqueda avanzada de direcciones IP a criterio del administrador.
executeMostrarIps ()	Llama a la función MostrarIps() de la clase ControlIp y muestra en la Client Page correspondiente a executeBuscar(), utilizando AJAX, las direcciones IP de acuerdo al criterio seleccionado por el administrador.
executeAnadir_ip()	A la hora de adicionar una dirección IP y asignársela a un sitio, esta función guarda en una variable dichas direcciones IP y automáticamente las muestra en un ListBox para visualización del administrador.
executeBuscarSitio()	Función que es llamada desde insertSuccess mediante AJAX y dado el id de un Ensayo Clínico hace una búsqueda de todos los sitios que pertenezcan a este y que aun no tengan direcciones IP asignadas y los muestra en un select.
executeAutocompletar()	Función para completar una palabra o frase en un edit según las letras que se conozcan de la misma, en este caso para el ensayo y el sitio a que está asignado dicha dirección IP.
executeLlenarIp()	Desde la función executeInsert() se hace una llamada a esta función a través de AJAX y se muestra un formulario para introducir direcciones y rangos IP, según los datos introducidos por el administrador, se hacen llamadas a esta misma función para que valide los datos introducidos mediante las funciones Validar_ip() y Validar_rango() de la clase ControlIp, el

	administrador puede eliminar los datos introducidos, en caso de que haya habido algún error, para esto se llama a esta misma función y esta a su vez llama a eliminar_ip() o eliminar_rango(), según sea el caso, de la clase ControlIp.

Tabla 2.3 Descripción de la clase del diseño: adm_AsignacionesActions

Nombre: adm_AsignacionesActions	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Gestionar Asignación, llama a las Clases ControlAsignarIP, ControlEC, ControlUsuario, ControlSitio para utilizar sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de las asignaciones de roles y direcciones IP existentes en la base de datos.
executeList()	Llama a la función getAsignarPager() de la clase ControlAsignarIP y muestra en la Client Page correspondiente el listado de asignación de roles y direcciones IP.
executeShow()	Llama a la función DevolverPorPK() de la clase ControlAsignarIP y muestra los datos de la asignación seleccionada por el administrador, sin opción de modificarlos.
executeInsert()	Llama a la función Buscar() de la clase ControlUsuario y Buscar_habilitados() de la clase ControlEC, de donde obtiene los usuarios y los ensayos que existen, hace una consulta para buscar los roles y luego muestra en la Client Page correspondiente un formulario para insertar en la base de datos la asignación.

executeInsertar()	A través del sitio seleccionado por el usuario busca el hospital al que pertenece ya que es un parámetro necesario para insertar una asignación, los demás parámetros a insertar se encuentran en las sesiones de donde son obtenidos por el método, en caso de no haber errores, inserta los datos y redirecciona a la página donde muestra los datos de dicha asignación, sino lanza una excepción. En el epígrafe “ejemplos de código” de este capítulo se representa el código de la función debido a su importancia y relativa complejidad.
executeEdit()	Llama a la función DevolverPorPK() de la clase ControlAsignarIP obteniendo así un objeto AsignarRolIp, guarda en sesiones los datos de la asignación y las direcciones IP y los rangos que tiene el usuario antes de editarse y así tener control sobre ellos y luego muestra en la Client Page correspondiente los datos de la asignación seleccionada por el administrador con opción de modificarlos.
executeUpdate()	Antes de todo se encarga de validar las direcciones IP y los rangos, tanto al almacenarlos en una variable como a la hora de guardarlos en la base de datos, se auxilia de funciones como ValidarIpBDEditar(), ValidarIpVariable(), ValidarRangoBDEditar() y ValidarRangoVariable(), antes de enviar la información da la opción de eliminar las direcciones IP en caso de algún error al teclearlos a través de eliminar_ip() y eliminar_rango(), todas estas funciones de la clase ControlAsignarIp, luego y cuando todo esté correcto modifica los datos y redirecciona a la página donde muestra los datos de la asignación modificada.
executeDelete()	Llama a la función DevolverPorPK() de la clase ControlAsignarIP, elimina los datos y redirecciona a la página donde se muestran las asignaciones de roles y direcciones IP.
executeLlenarips()	Desde la función executeInsert() se hace una llamada a esta función a través de AJAX y se muestra un formulario para introducir direcciones y rangos IP, según los datos introducidos por el administrador, se hacen llamadas a esta misma función para

	que valide los datos introducidos mediante las funciones ValidarIPBD() y ValidarIpVariable() de la clase ControlAsignarIP, el administrador puede eliminar los datos introducidos en caso de que haya habido algún error, para esto se llama a esta misma función y esta a su vez llama a eliminar_ip() o eliminar_rango(), según sea el caso, de la clase ControlAsignarIp. En el epígrafe "ejemplos de código" de este capítulo se representa el código de la función debido a su magnitud, importancia y complejidad.
executeBuscarSitio()	Función que es llamada desde insertSuccess mediante AJAX y dado el id de un Ensayo Clínico hace una búsqueda de todos los sitios que pertenezcan a este y los muestra en un select.
executeBuscarRoles()	Al sitio Centro Promotor se le pueden asignar algunos roles que a los demás sitios no, por lo que en esta función se almacenan en una variable los nombres de los roles para el centro promotor y en otra variable los roles para los demás sitios, se verifica cual sitio se seleccionó, de acuerdo a los roles que puede tener se buscan en la base de datos por el nombre y se devuelve el id de los mismos, luego la función devuelve un arreglo, en el que la posición coincide con el id de cada rol y almacena en cada una los nombres de los roles de acuerdo al sitio escogido.
executeDameSitio()	Guarda en una sesión sitio el id de un sitio.
executeBuscar()	Muestra en su Client Page correspondiente un formulario para hacer una búsqueda a criterio del administrador.
executeMostrarAsignacion()	Es llamada mediante AJAX y según los criterios seleccionados por el administrador hace una búsqueda en la base de datos de alguna asignación que cumpla estos criterios, esto lo hace a través de la función MostrarAsignacion() de la clase ControlAsignarIP y los muestra en la Client Page correspondiente a executeBuscar().
executeAutocompletar()	Función para completar una palabra o frase en un edit según las letras que se conozcan de la misma, en este caso para el ensayo, el sitio, el login de usuario y el nombre del rol para una asignación determinada.

Tabla 2.4 Descripción de la clase del diseño: FachadaAdmin

Nombre: FachadaAdmin	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de autenticarse)	
Atributo	**Tipo**
$instancia	privado
Responsabilidades	
Nombre:	**Descripción:**
getInstancia()	Devuelve una instancia de la clase FachadaAdmin
rolesUsuario()	Devuelve los roles que tiene un usuario.
verificar_autenticar()	Verifica que el nombre de usuario y la contraseña existan en la base de datos, que el usuario tenga permiso para acceder al sistema en cuanto a rol y en cuanto a la dirección IP desde donde se conecta, llevando esto consigo comparaciones entre la dirección IP de la PC cliente y las posibles direcciones IP que pudiera tener el usuario en la base de datos y que indican desde que PC tiene acceso el mismo al sistema.
crearUsuario ()	Crea un usuario y devuelve un objeto del mismo.
Verificar_bandera()	Restringe el acceso a aquellos usuario que intente loguearse con una cuenta que ya está en el sistema, cada vez que un usuario intente esto se verifica en la tabla sesiones de la base de datos que la sesión esté activa, en caso de que el usuario ya logueado esté más de 10 min inactivo dentro del sistema la próxima acción que se realice con esta misma cuenta va a desactivar la sesión inactiva y requerir volver a loguearse.
verificarRolHabilitado()	Comprueba que exista la asignación usuario, rol, ensayo en la base de datos, en caso afirmativo recoge las direcciones y el rango IP que tiene asignado el usuario, comprueba que coincida con la dirección IP desde donde está conectado y que el rol que tiene para ese ensayo esté habilitado en la base de datos, devuelve verdadero o falso según sea el caso.

Tabla 2.5 Descripción de la clase del diseño: ControlUsuario

Nombre: ControlUsuario	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar un usuario, utiliza la clase AUsuarioPeer)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getUsuarioPager()	Devuelve los usuarios almacenados en la base de datos y en caso de que sean muchos hace una paginación de los mismos.
DevolverPorPK()	Llama a la función retriveByPK() de la clase AUsuarioPeer y devuelve un objeto usuario según la llave primaria que se le pasó por parámetro.
Insertar()	Hace una consulta a la base de datos para comprobar si ya existe o no el usuario que se desea insertar devolviendo verdadero o falso según sea el caso.
Buscar()	Devuelve un arreglo con todos los usuarios almacenados en la base de datos.
MostrarUsuario()	Según el criterio de búsqueda hace una consulta a la base datos y devuelve un arreglo de objetos con los usuarios encontrados.
Mostrar_usuario_cont()	Busca en la base de datos un usuario que coincida con el usuario y la contraseña especificada por parámetro y devuelve un objeto usuario.
CalcularFecha()	Dado el id de un usuario por parámetro, se busca la fecha en que se modificó la contraseña por última vez, se busca la fecha actual, se convierte a segundos, se resta esta con la almacenada en la base de datos y la diferencia se vuelve a llevar a días y se devuelve esta cantidad.

Tabla 2.6 Descripción de la clase del diseño: ControlSitio

Nombre: ControlSitio
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar

un sitio, utiliza la clase ASitioPeer)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getSitioPager()	Devuelve los sitios almacenados en la base de datos y en caso de que sean muchos hace una paginación de los mismos.
DevolverPorPK()	Llama a la función retriveByPK() de la clase ASitioPeer y devuelve un objeto sitio según las llaves primarias que se le pasó por parámetro.
Insertar()	Hace una consulta a la base de datos para comprobar si ya existe o no el sitio que se desea insertar devolviendo verdadero o falso según sea el caso.
Buscar()	Devuelve un arreglo de sitios con todos los sitios almacenados en la base de datos.
MostrarSitios()	Según el criterio de búsqueda hace una consulta a la base datos y devuelve un arreglo de objetos con los sitios encontrados.
BuscarPorId()	Busca un sitio en la base de datos según el id especificado por parámetros, devuelve un objeto de sitio.
BuscarSitioSinIp()	Hace una búsqueda de todos los sitio de la tabla sitio, luego comprueba si cada uno de ellos está en la tabla IP, si no está, lo cual indica que no tiene direcciones IP asignadas, los almacena en un arreglo de objetos y los devuelve.
BuscarSitioPorEnsayo()	Dado un id de ensayo se buscan en la tabla sitios los que pertenezcan a dicho ensayo y se verifica en la tabla IP que estos sitios tengan asignado alguna dirección IP, luego se devuelve un arreglo con los nombres de estos sitios.

Tabla 2.7 Descripción de la clase del diseño: ControlEC

Nombre: ControlEC
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar un Ensayo Clínico, utiliza la clase AEnsayoClinicoPeer)

Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getEnsayoPager()	Devuelve los ensayos almacenados en la base de datos y en caso de que sean muchos hace una paginación de los mismos.
DevolverPorPK()	Llama a la función retriveByPK() de la clase AEnsayoClinicoPeer y devuelve un objeto ensayo según la llave primaria que se le pasó por parámetro.
Insertar()	Hace una consulta a la base de datos para comprobar si ya existe o no el ensayo que se desea insertar devolviendo verdadero o falso según sea el caso.
Buscar()	Devuelve un arreglo de objetos de ensayos con todos los almacenados en la base de datos.
MostrarEnsayos()	Según el criterio de búsqueda hace una consulta a la base datos y devuelve un arreglo de objetos de ensayos con los encontrados.
Buscar_habilitados()	Se buscan todos aquellos ensayos que estén habilitados, es decir, que el campo Habilitar_ensayos de la tabla Ensayo_Clinico esté en verdadero, devuelve un arreglo de objetos ensayos habilitados.
VerificarSiHabilitado()	Recibe un id de ensayo como parámetro y verifica en la base de datos si el mismo está o no habilitado, devuelve verdadero o falso según sea el caso.

Tabla 2.8 Descripción de la clase del diseño: ControlIp

Nombre: ControlIp	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar una dirección o un rango IP, utiliza la clase AIpPeer)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**

getIpPager()	Devuelve las direcciones IP almacenadas en la base de dato y en caso de que sean muchas hace una paginación de las mismos.
DevolverPorPK()	Llama a la función retriveByPK() de la clase AIpPeer y devuelve un objeto IP según las llaves primarias que se le pasó por parámetro.
SitioPerteneceHospital()	Dado el id de un sitio se busca el mismo en la base de datos y con el objeto creado se busca y devuelve el nombre del hospital a que pertenece.
Buscar()	Devuelve un arreglo de objetos de ensayos con todos los almacenados en la base de datos.
MostrarEnsayos()	Según el criterio de búsqueda hace una consulta a la base datos y devuelve un arreglo de objetos de ensayos con los encontrados.
Buscar_habilitados()	Dado un id de ensayo, se buscan todos aquellos que estén habilitados, es decir, que el campo Habilitar_ensayos de la tabla Ensayo_Clinico este en verdadero, devuelve un arreglo de objetos ensayos.
DameEnsayo()	Dado el id de un sitio se busca el mismo en la base de datos y con el objeto creado se busca y devuelve el id del ensayo al que está asignado.
validar_ip ()	Función que recibe 3 parámetros, el primero es la dirección IP que se desea almacenar en una sesión luego de comprobar que no coincida con el 2do parámetro, el cual puede ser nulo que es la dirección IP guardada con anterioridad en la sesión, y comprobar que no se encuentre además dentro del tercer parámetro que es un rango IP en caso de que ya se haya guardado un rango anteriormente, devuelve verdadero o falso según sea el caso, esto garantiza que no se entren direcciones o rangos IP iguales.
Validar_rango()	Función que recibe 4 parámetros, los 2 primeros son el rango IP que se desea almacenar en una sesión luego de comprobar que no coincida con el tercer parámetro, el cual puede ser nulo y es un rango guardado con anterioridad en la sesión y luego de comprobar que el cuarto parámetro que puede ser una dirección IP no se encuentre entre los dos primeros, devuelve verdadero o falso según sea el caso, esto garantiza que no se entren

	direcciones o rangos IP iguales.
eliminar_ip()	Dado una posición y una variable con las direcciones IP separadas por “;” se dividen formando las direcciones IP correctamente y se elimina según la posición especificada, luego se vuelven a unir separados por “;” y se devuelve otra variable actualizada.
eliminar_rango()	Dado una posición y una variable por parámetros con los rangos IP separados por “;” se dividen formando los rangos correctamente y se eliminan según la posición especificada, luego se vuelven a unir separados por “;” y se devuelve otra variable actualizada.
guardar_ip()	A la hora de editar una asignación de IP a un sitio y se elimina una dirección IP de las que tenia asignadas, se llama a esta función para volver a almacenar las direcciones IP que quedaron en la base de datos.
guardar_rango()	A la hora de editar una asignación de IP a un sitio y se elimina un rango IP de los que tenia asignados, se llama a esta función para volver a almacenar los rangos que quedaron en la base de datos.
ValidarEnsayoSitio()	Según los dos enteros que se le pasan por parámetro si uno de los dos es igual a -1 retorna falso, si los dos son diferentes de -1 retorna verdadero.
DameSitioSinIp()	Dado un id de ensayo busca en la tabla sitio los que pertenezcan a este ensayo y luego comprueba en la tabla IP que estos sitios existan allí, lo cual significa que tienen direcciones IP asignadas, devuelve un arreglo con los nombres de los sitios que cumplan con estos dos criterios.

Tabla 2.9 Descripción de la clase del diseño: ControlAsignarIp

Nombre: ControlAsignarIp	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar una asignación de roles y direcciones IP a un usuario en un ensayo, utiliza la clase ARolUseIpPeer)	
Atributo	**Tipo**
Responsabilidades	

Nombre:	**Descripción:**
getIpPager()	Devuelve las asignaciones de roles y direcciones IP de cada usuario que se encuentran almacenadas en la base de dato y en caso de que sean muchas hace una paginación de las mismas.
DevolverPorPK()	Llama a la función retriveByPK() de la clase ARolUseIpPeer y devuelve un objeto de la asignación según las llaves primarias que se le pasó por parámetro.
validarIpBd()	Función que recibe 3 parámetros, la dirección IP que se desea asignar al usuario, las direcciones IP que ya tiene el sitio y los rangos del mismo, pudiendo ser uno de estos dos últimos nulo; pero no los dos, se verifica que la dirección IP coincida con algunas de las que pudiese tener el sitio y que se encuentre dentro del rango permisible, es decir, dentro del rango pasado con el tercer parámetro. Se auxilia de la función de php explode(), la cual divide cadena por cadena según un parámetro delimitador.
validarIpVariable()	Función que recibe 3 parámetros, el primero es la dirección IP que se desea almacenar en una sesión luego de comprobar que no coincida con el 2do parámetro, el cual puede ser nulo que es la dirección IP guardada con anterioridad en la sesión y que no se encuentre dentro del tercer parámetro que es un rango IP en caso que ya se haya guardado un rango anteriormente, devuelve verdadero o falso según sea el caso, esto garantiza que no se entren direcciones o rango IP iguales.
validarRangoVariable()	Función que recibe 4 parámetros, los 2 primeros son el rango IP que se desea almacenar en una sesión luego de comprobar que no coincida con el tercer parámetro, el cual puede ser nulo y es un rango IP guardado con anterioridad en la sesión y comprobar también que el cuarto parámetro que puede ser una dirección IP no se encuentre entre los dos primeros, devuelve verdadero o falso según sea el caso, esto garantiza que no se entren direcciones o rangos IP iguales.
validarRangoBd()	Función que recibe 3 parámetros, primera y segunda dirección IP que son los que forman el rango que se desea asignar al usuario,

	como tercero los rangos del sitio, pudiendo ser este último nulo; se verifica que el rango a asignar coincida o esté dentro de los rangos del sitio, es decir, dentro del tercer parámetro. Se auxilia de la función de php explode(), la cual divide cadena por cadena según un parámetro delimitador.
MostrarAsignacion()	Se busca, el sitio, el ensayo, el usuario y el rol en sus respectivas tablas en la base de dato, se obtiene el id de cada uno y se verifica que se encuentren en la tabla RolUseIp como una asignación, se devuelve un objeto de la misma.
validarIpBdEditar()	Función que a la hora de editar una asignación, según el id del sitio que se le pasa por parámetro, busca las direcciones y los rangos IP del mismo y verifica que la dirección IP que se le quiere asignar pertenezca a este sitio, es decir, asignar una dirección IP dentro del rango permisible. Se auxilia de la función de php explode(), la cual divide cadena por cadena según un parámetro delimitador.
validarRangoBdEditar()	Función que a la hora de editar una asignación, según el id del sitio que se le pasa por parámetro, busca los rangos IP del mismo y verifica que el rango IP que se le quiere asignar pertenezca a este sitio, es decir, asignar un rango dentro de los que ya tiene el sitio. Se auxilia de la función de php explode(), la cual divide cadena por cadena según un parámetro delimitador.
guardarCambios()	Función que a la hora que se desee habilitar o deshabilitar una asignación, busca en la tabla RolUseIp de la base de dato a través de retriveByPK, crea un objeto de la misma y guarda el nuevo cambio, es decir, pone en true o false, según se haya deseado, el campo Habilitado.
validarAsignacion()	Función que busca en la tabla RolUseIp una asignación que coincida con los datos pasados por parámetros, si encuentra alguna retorna false, lo cual indica que esa asignación ya existe, sino retorna verdadero.
eliminar_ip()	Dado una posición y una variable con las direcciones IP separadas por “;” se dividen formando las direcciones IP correctamente y se

	elimina según la posición especificada, luego se vuelven a unir separados por “;” y se devuelve otra variable actualizada.
eliminar_rango()	Dado una posición y una variable pasados por parámetros con los rangos IP separados por “;” se dividen formando los rangos correctamente y se eliminan según la posición especificada, luego se vuelven a unir separados por “;” y se devuelve otra variable actualizada.
guardarIp()	A la hora de editar una asignación de una dirección IP a un sitio y se elimina una dirección IP de las que tenia asignadas, se llama a esta función para volver a almacenar las direcciones IP que quedaron en la base de datos.
guardarRango()	A la hora de editar una asignación de direcciones IP a un sitio y se elimina un rango IP de los que tenia asignados, se llama a esta función para volver a almacenar los rangos que quedaron en la base de datos.
BuscarRol()	Dado el id de un rol lo busca en la base de datos y devuelve un objeto rol
BuscarRolesPorSitios()	Los roles para el sitio Centro Promotor son diferentes que para los otros sitios, por tanto esta función dado el id de un sitio, a través de constantes definidas se verifica si el sitio es centro promotor o no, de acuerdo al caso que sea se devuelve un arreglo con los roles que tiene cada tipo de sitio.
VerificarRolesPorUsuario()	Dado un usuario, un rol y una dirección IP se verifica que el rol sea monitor, auditor o gerente de datos, en caso de ser uno de esos se verifica que dicho usuario tenga alguno de esos roles en ese ensayo, en caso que los tenga se devuelve falso, y se devuelve verdadero en caso de que el rol pasado por parámetro no sea ninguno de los tres roles mencionados o que el usuario no tenga esos roles en el ensayo

Tabla 2.10 Descripción de la clase del diseño: myUser

Nombre: myUser	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Autenticar, llama a la Clase FachadaAdmin, ControlUsuario y myUser para utilizar algunas de sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
signIn()	Esta función pone autenticado en true y crea una sesión 'UsuarioActivo' con el usuario que entró al sistema.
signOut()	Esta función pone autenticado en false, destruye la sesión del usuario y limpia las credenciales que tenía el mismo.
LimpiarCredencial()	Esta función limpia las credenciales del usuario, en este caso elimina el rol del usuario que estaba guardado en la credencial.
getUsuarioActivo()	Esta función devuelve un objeto del usuario que está activo en la sesión.
setUsuarioActivo()	Esta función cambia el usuario activo que está en la sesión.
limpiraIPs()	Esta función limpia las variables utilizadas para guardar en la sesión del usuario algunas direcciones IP que se le asignan.

Tabla 2.11 Descripción de la clase del diseño: UsuarioActivo

Nombre: UsuarioActivo	
Tipo: Clase(Es la clase que contiene los atributos y funciones correspondientes al usuario que está activo en la sesión.)	
Atributo	**Tipo**
idUsuario	public
nombreUsuario	public
ip	public
idRolActivo	public
rolAcivo	public

idEnsayoClinico	public
idSitio	public
nombreSitio	public
Responsabilidades	
Nombre:	**Descripción:**
__consruct()	Constructor de la clase.
getUsuario()	Esta función devuelve un objeto usuario de la tabla usuario pasándole como parámetro el id del usuario.
setSitioActivo()	Esta función cambia el valor de la variable idSitio e idEnsayoClinico según el sitio que se le pase como parámetro.
getSitioActivo()	Esta función devuelve un objeto del sitio de la base de datos pasándole como parámetro el id del sitio.
setRolActivo()	Esta función cambia el valor de la variable idRolActivo y rolActivo según el rol que se le pase como parámetro.
getRolActivo()	Esta función devuelve un objeto del rol de la base de datos pasándole como parámetro el id del rol.

2.5.2 Diagrama de clases del diseño para aplicaciones Web

Para la construcción de estos diagramas se utilizó la extensión que posee UML para el modelado de aplicaciones Web propuesta por Conallen. Usando los estereotipos <<Client Page>>, <<HTML Form>> y <<Server Page>>. A continuación se muestran los diagramas de clases del diseño para aplicaciones Web de los casos de uso Autenticarse, Gestionar Asignación y Gestionar IP, los diagramas de casos de uso restantes pueden ser vistos en el anexo del documento.

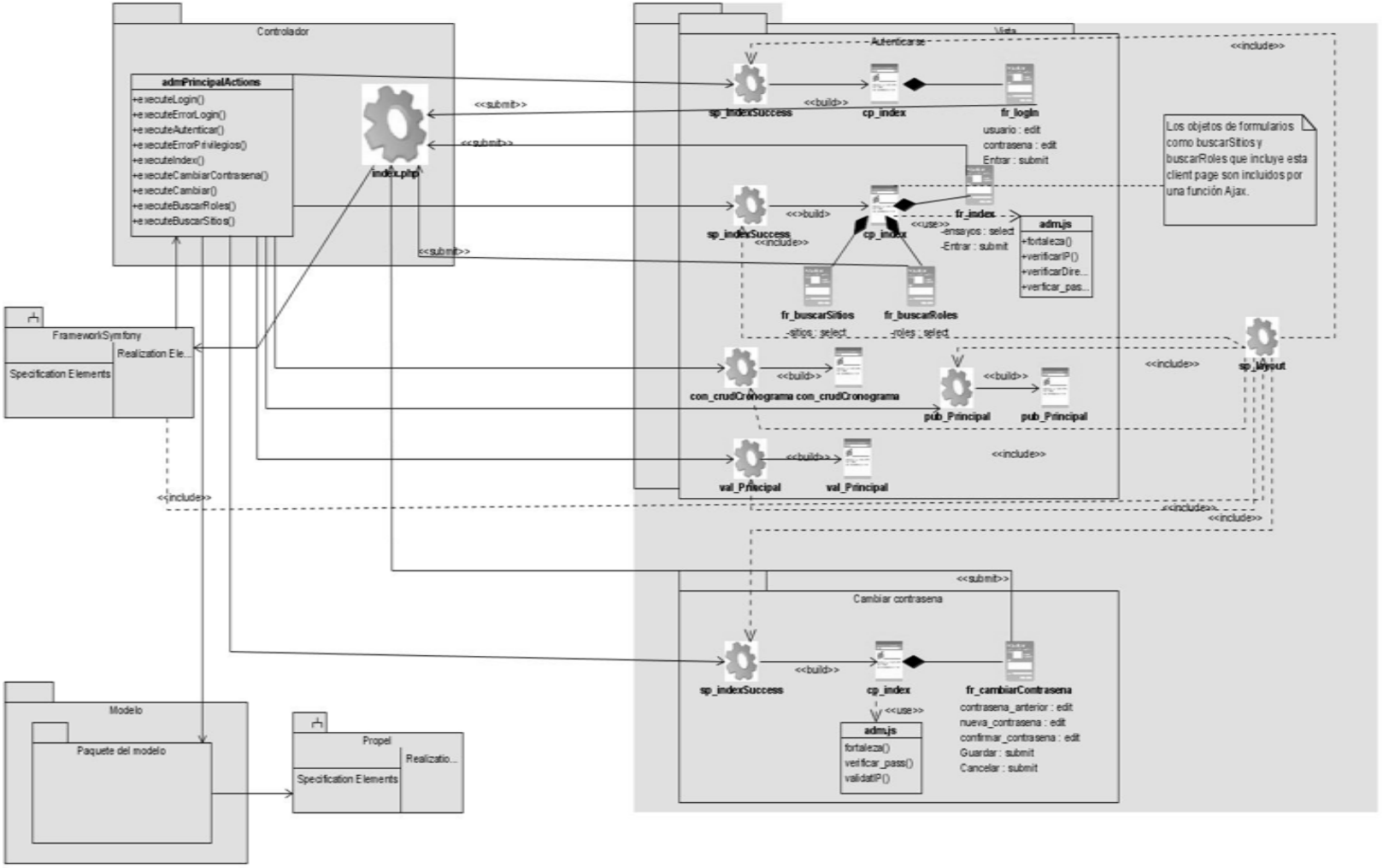

Figura 4 Diagrama de clases del diseño CU: Autenticar

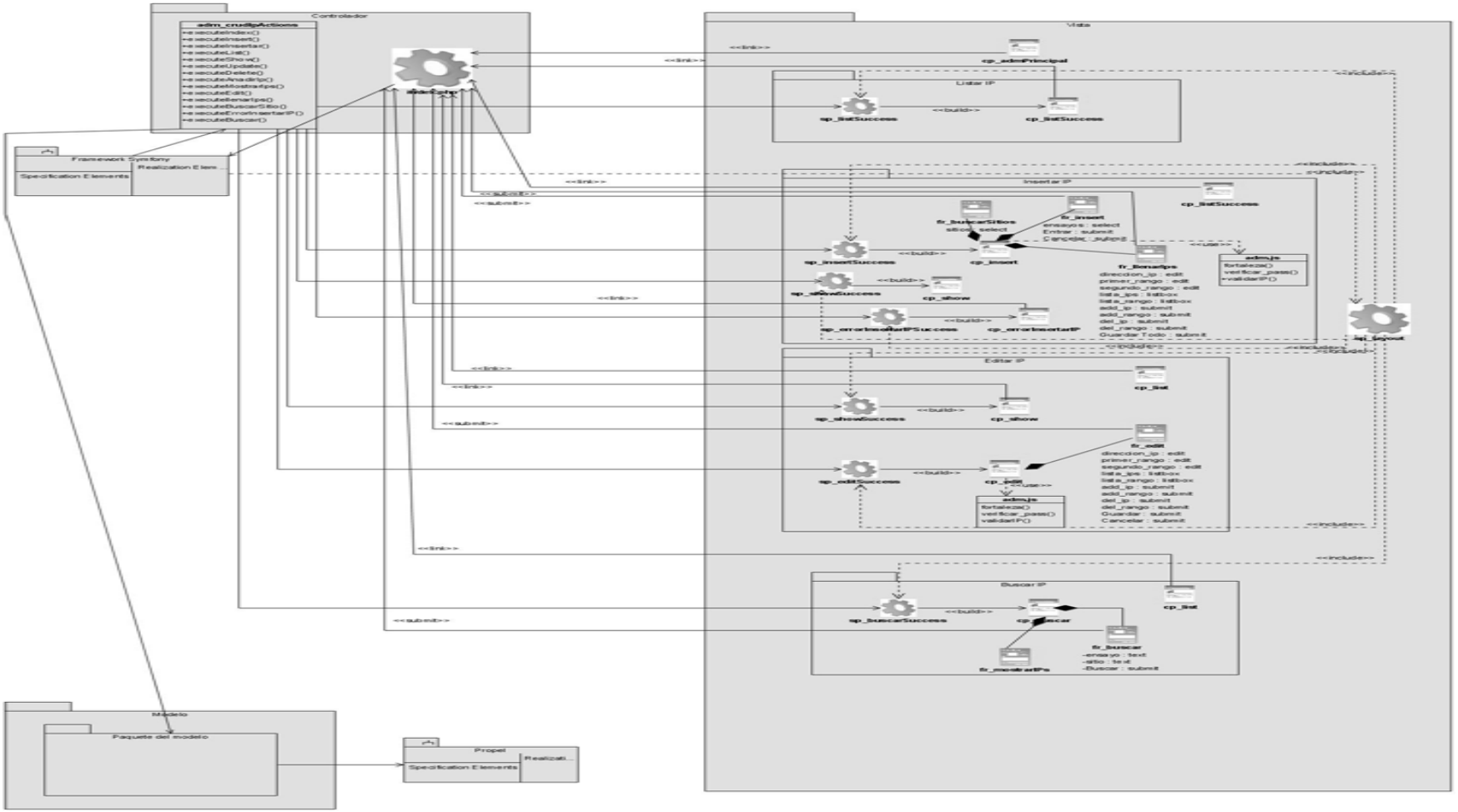

Figura 5 Diagrama de clases del diseño CU: Gestionar IP

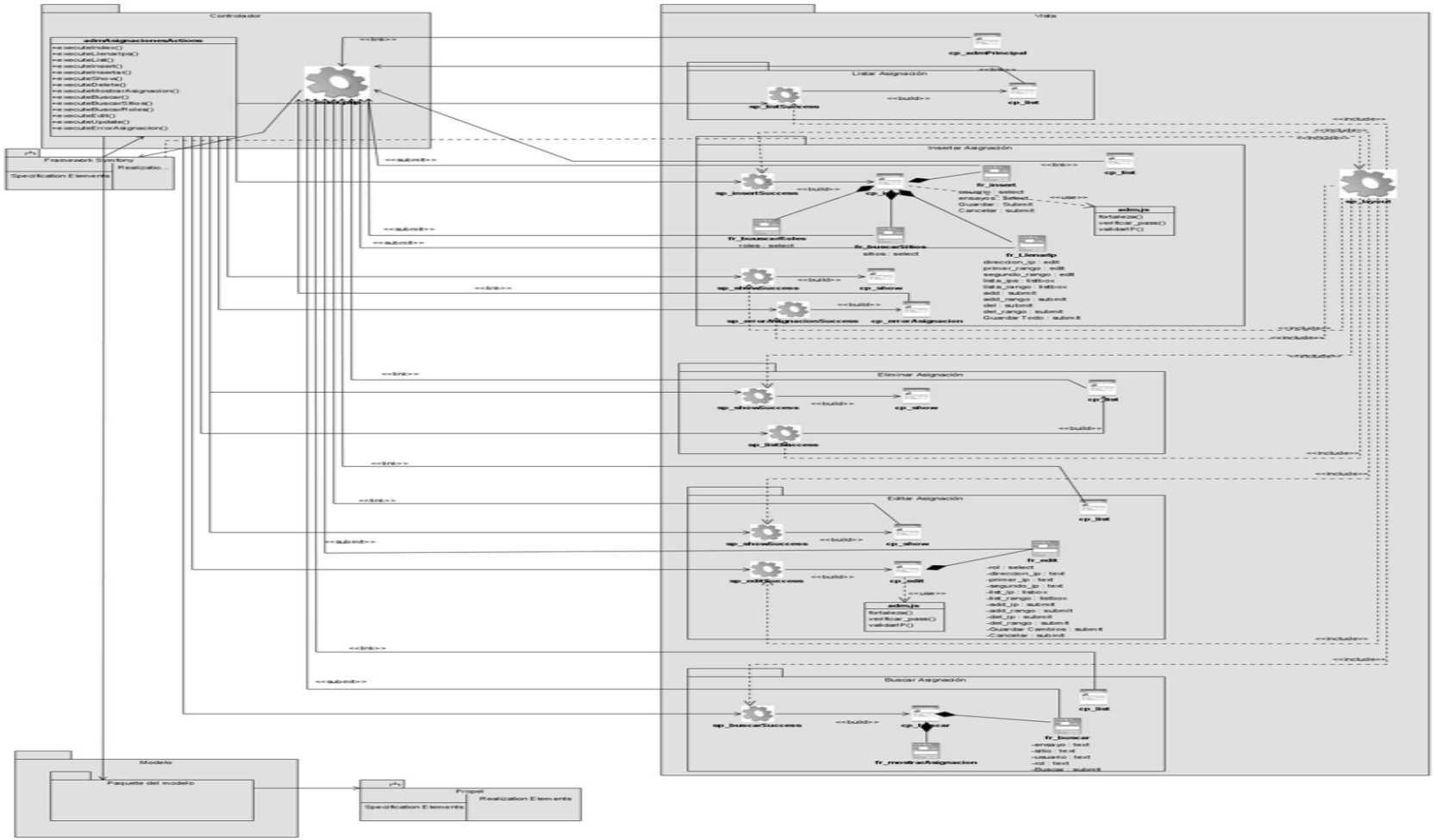

Figura 6 Diagrama de clases del diseño CU: Gestionar Asignación

2.5.3 Paquete del modelo.

En este diagrama se muestra la relación entre las clases del modelo.

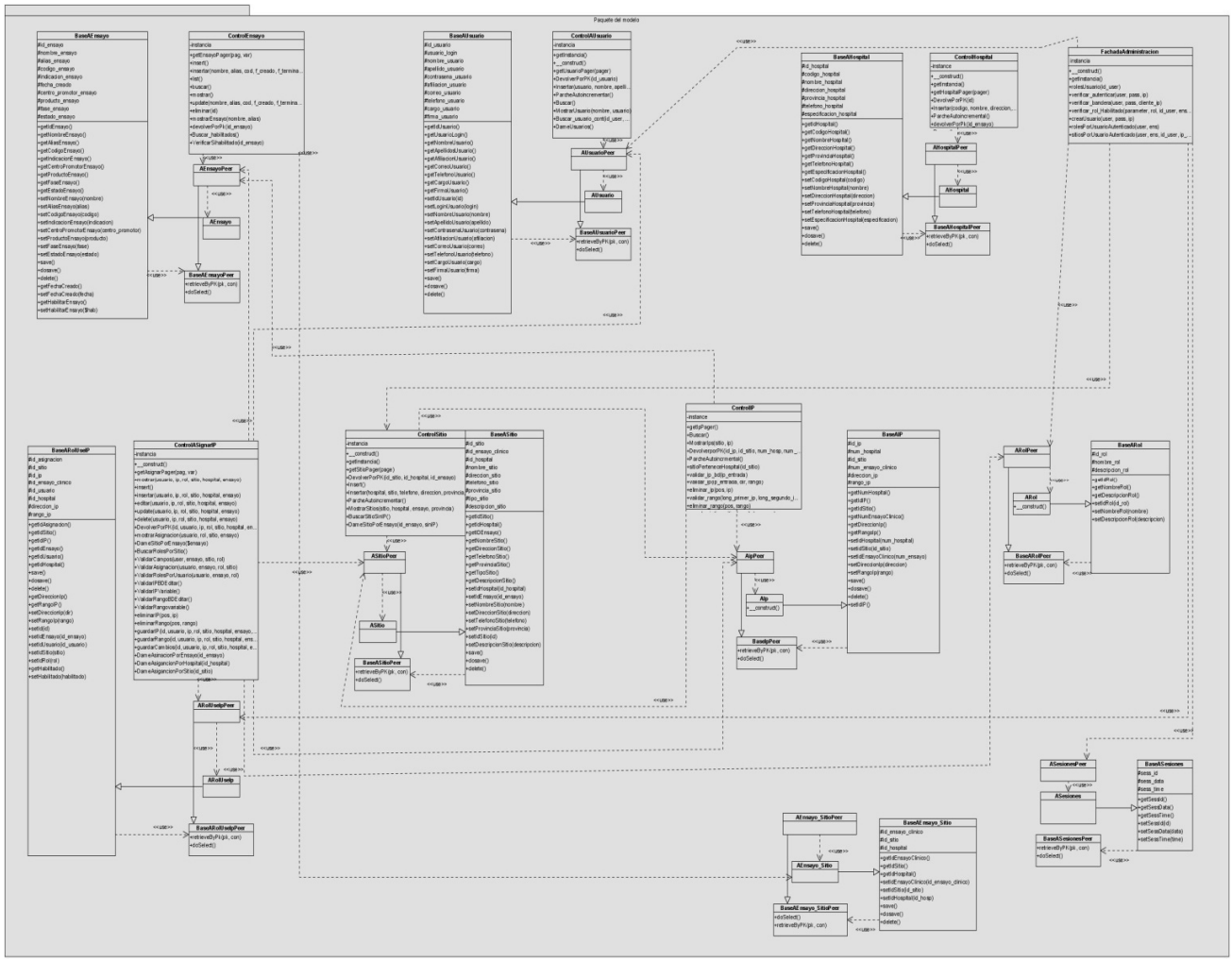

Figura 7 Paquete del modelo

2.5.4 Diagramas de interacción (secuencia) para Web.

A continuación se muestran los diagramas de secuencia correspondientes a los casos de uso Autenticar, Gestionar IP y Gestionar Asignación, mostrando la interacción de un conjunto de objetos en la aplicación a través del tiempo. Los diagramas de secuencia de los casos de uso restantes se pueden encontrar en los anexos del documento junto a los escenarios de Gestionar IP y Gestionar Asignación

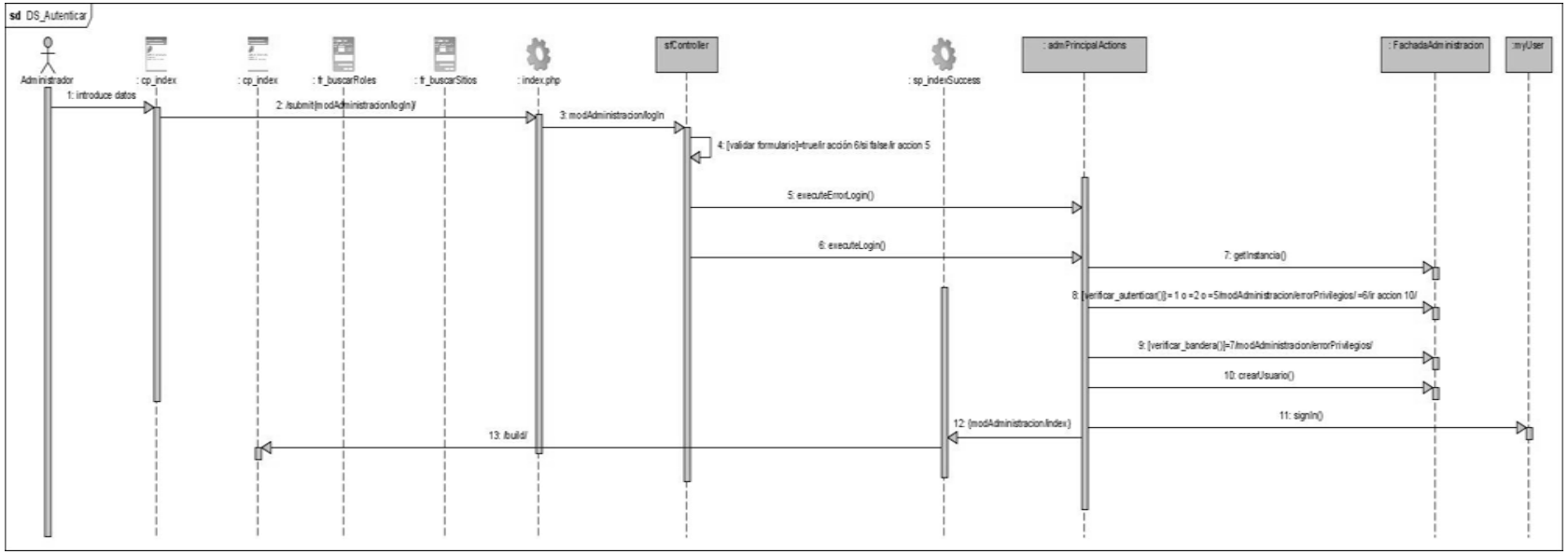

Figura 8 Diagrama de secuencia CU Autenticar.

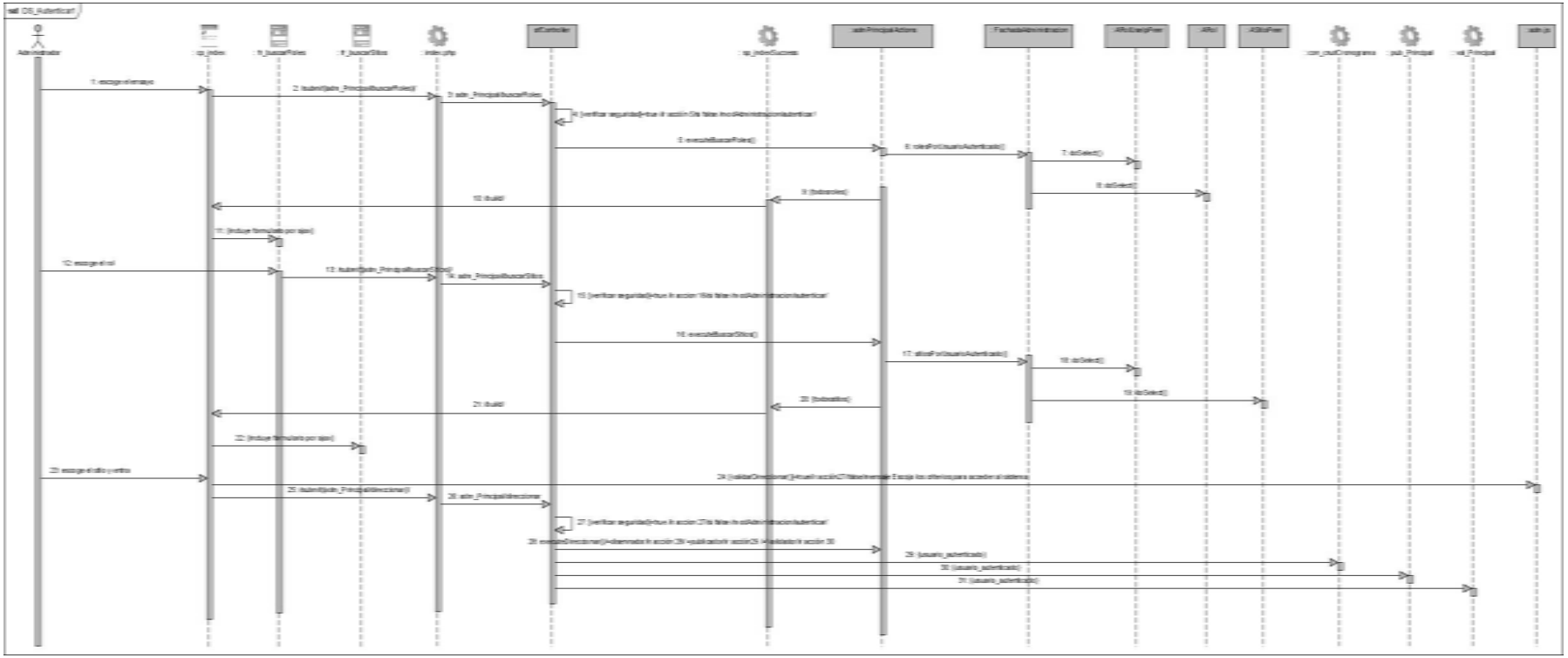

Figura 9 Diagrama de secuencia CU Autenticar (Continuación).

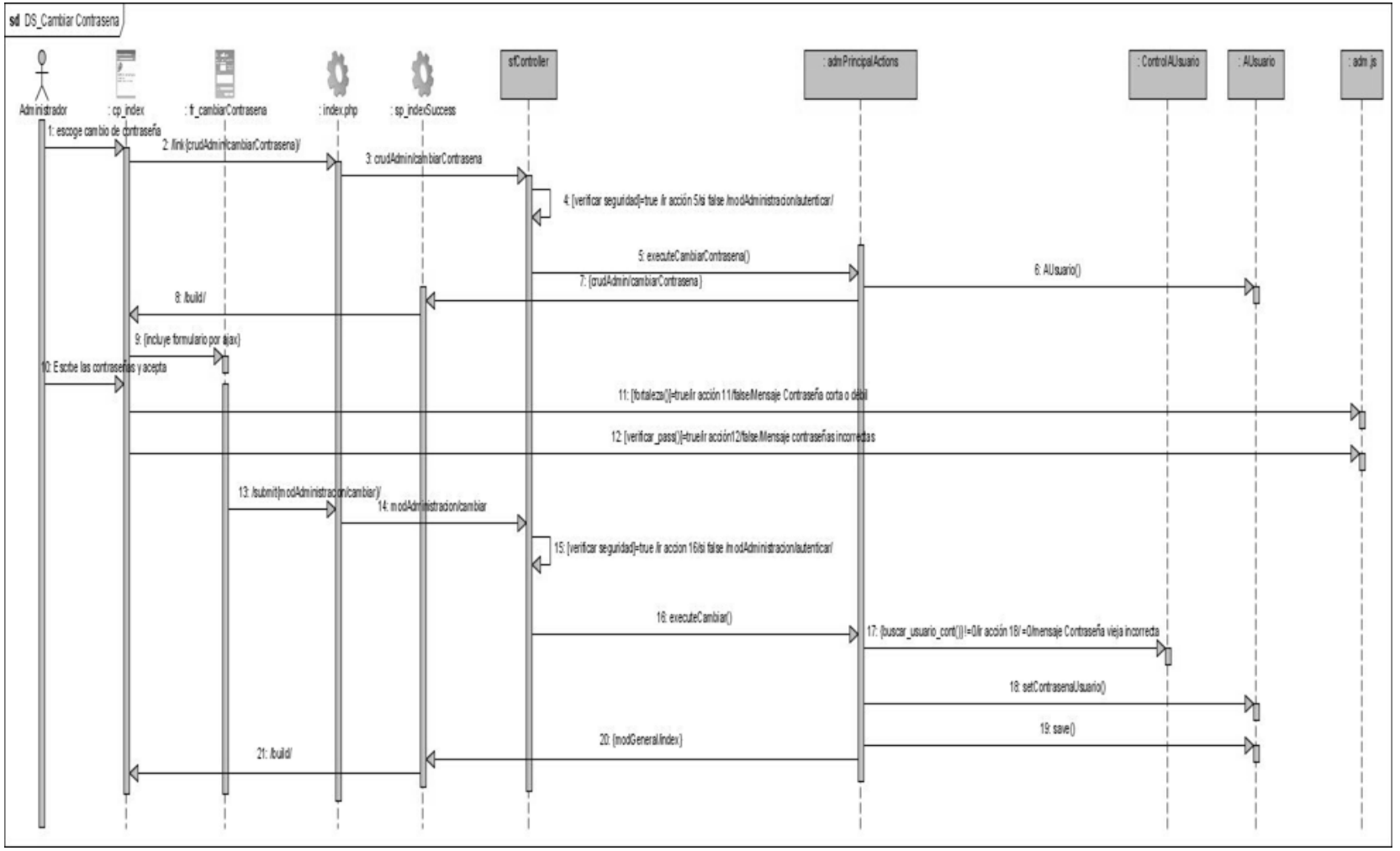

Figura 10 Diagrama de secuencia CU Autenticar (Cambiar contraseña).

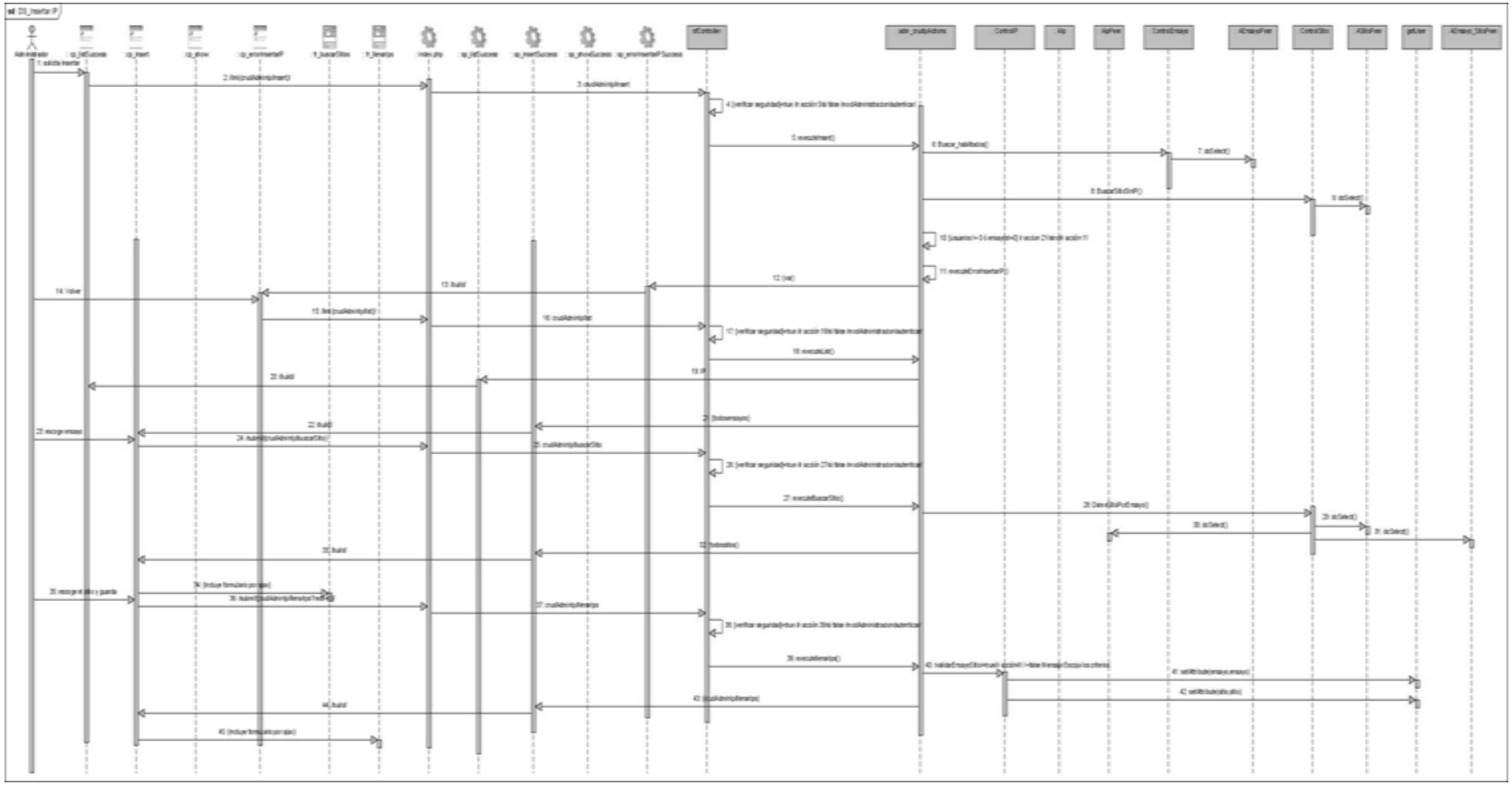

Figura 11 Diagrama de secuencia CU Gestionar IP (Insertar IP).

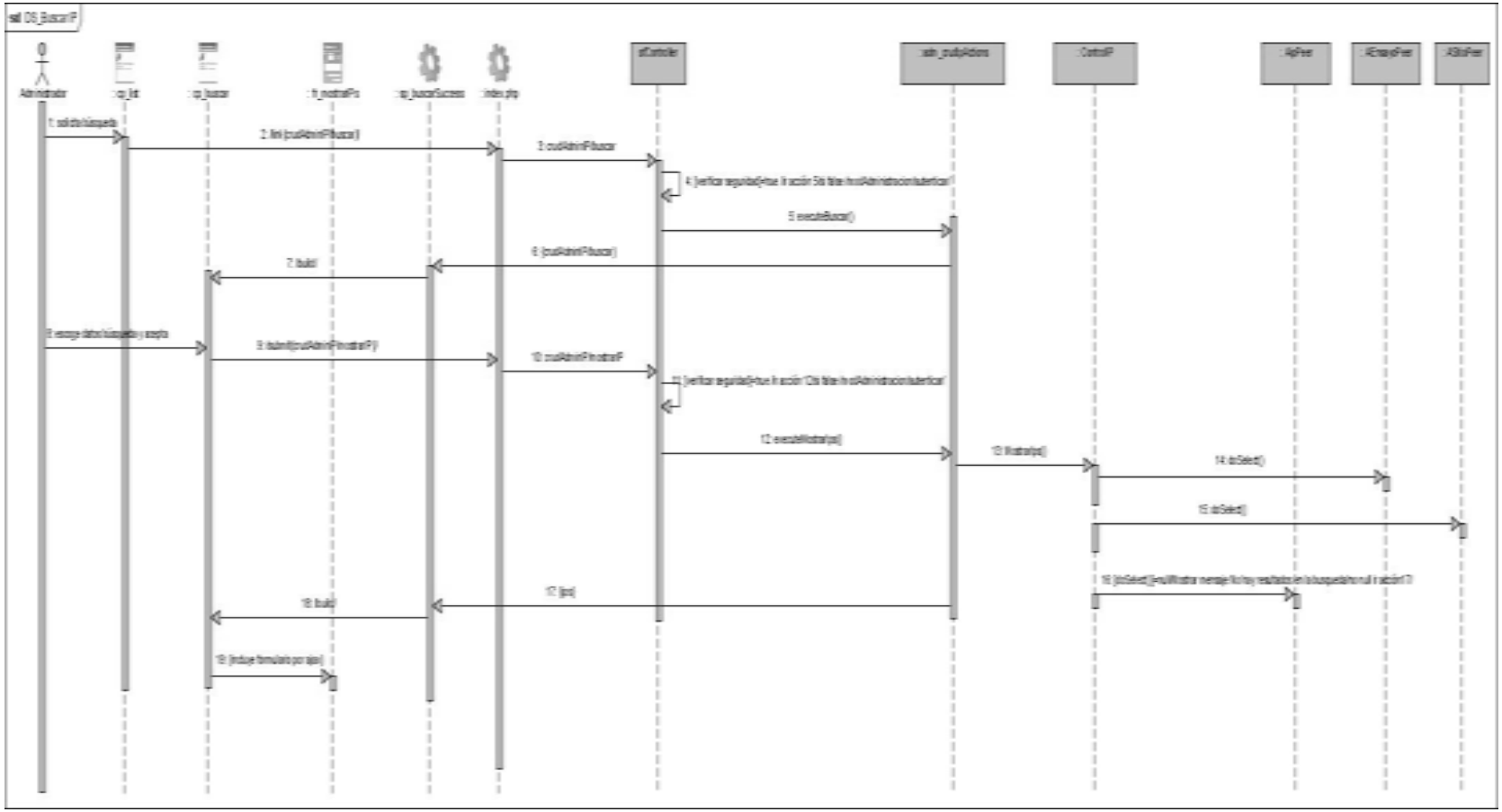

Figura 12 Diagrama de secuencia CU Gestionar IP (Buscar IP).

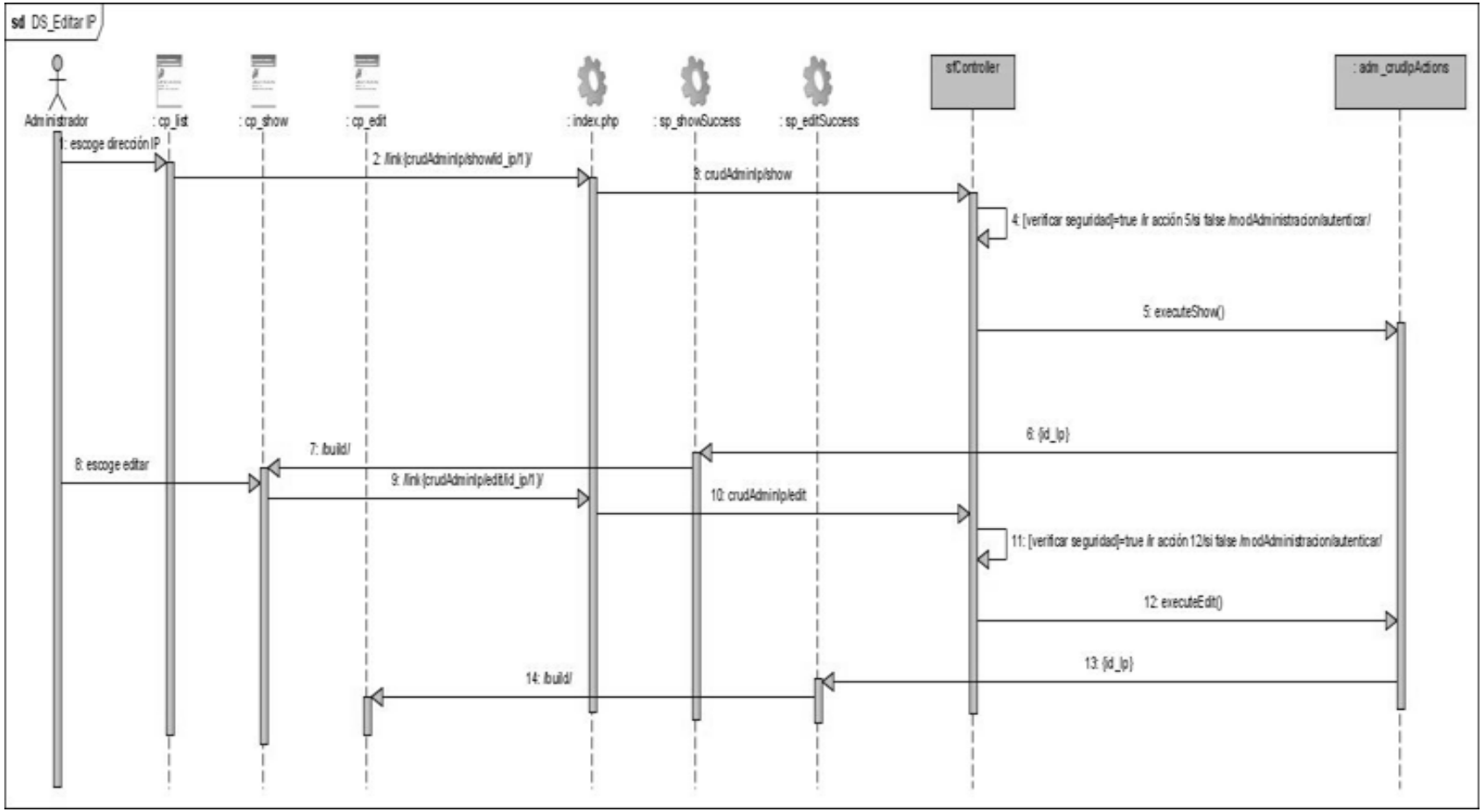

Figura 13 Diagrama de secuencia CU Gestionar IP (Editar IP).

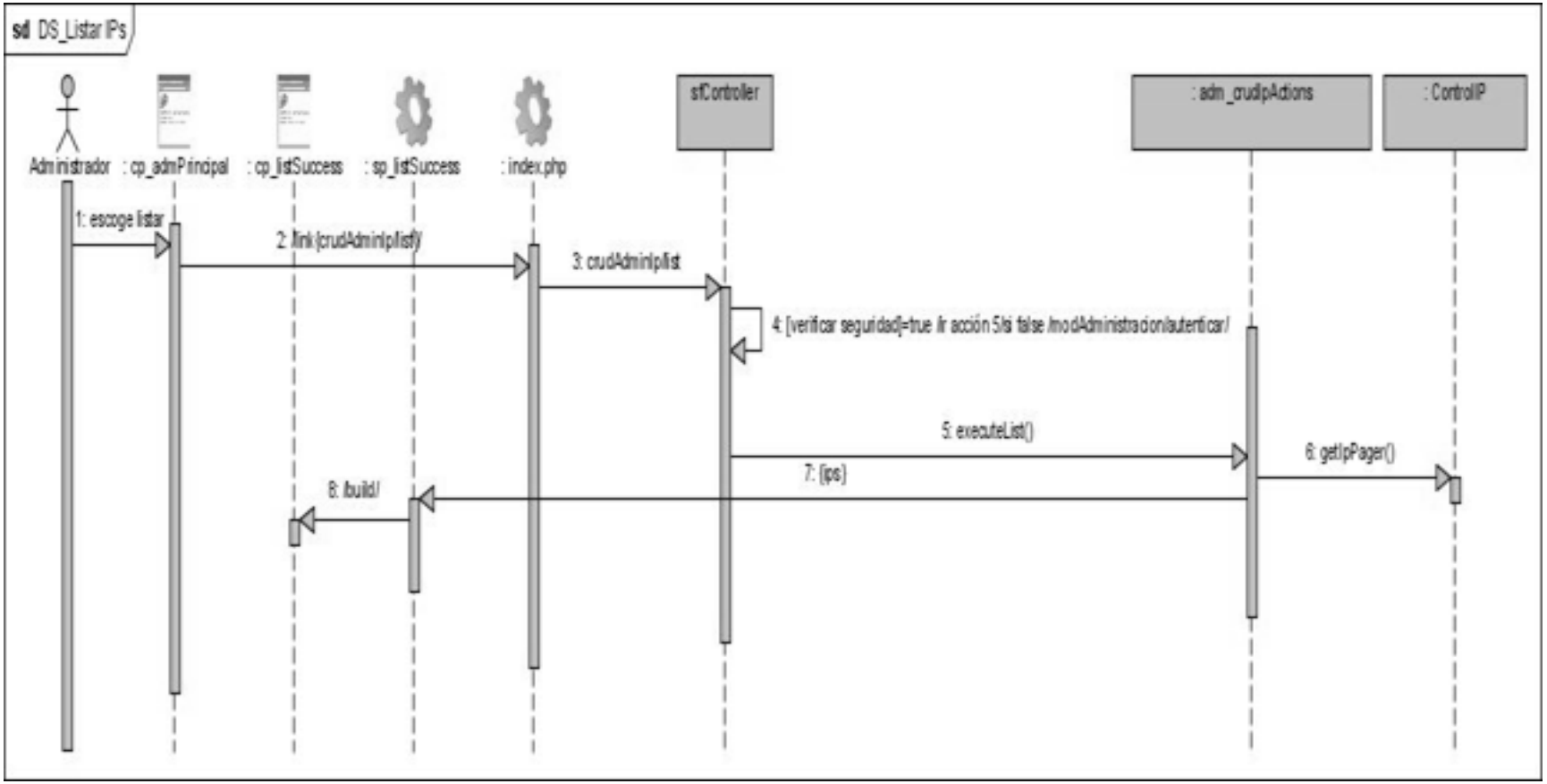

Figura 14 Diagrama de secuencia CU Gestionar IP (Listar IPs).

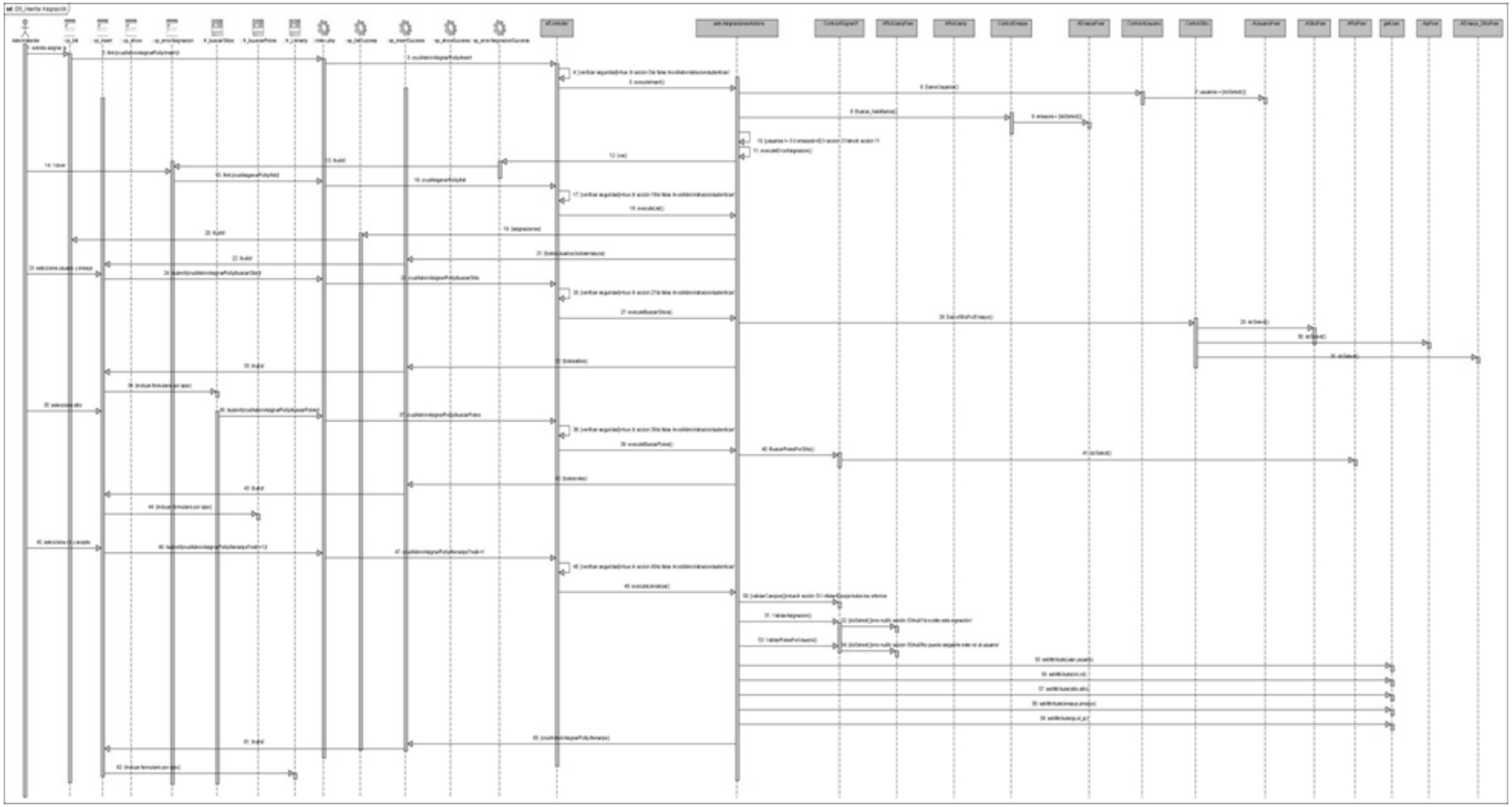

Figura 15 Diagrama de secuencia CU Gestionar Asignación (Insertar Asignación).

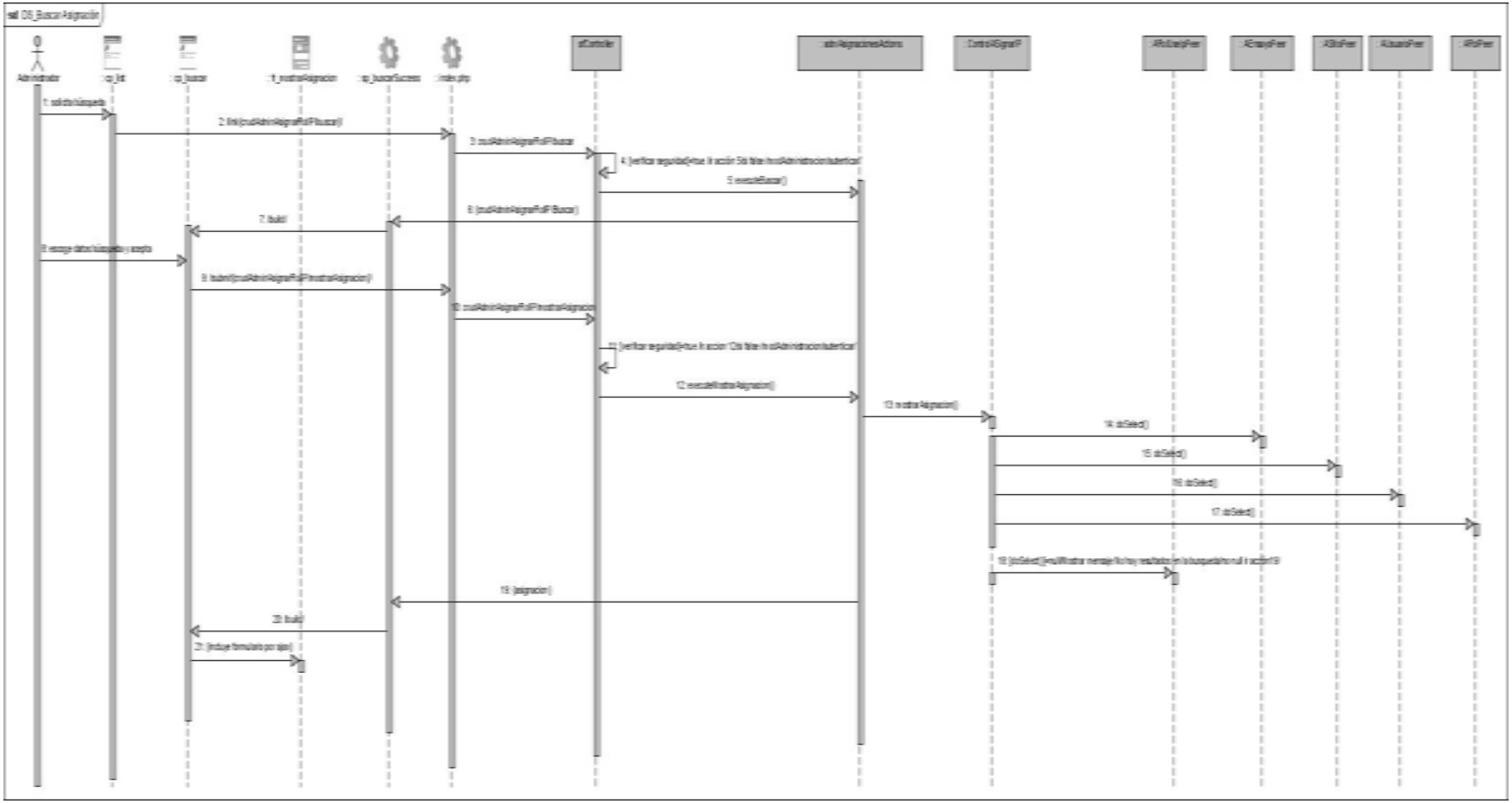

Figura 16 Diagrama de secuencia CU Gestionar Asignación (Buscar Asignación).

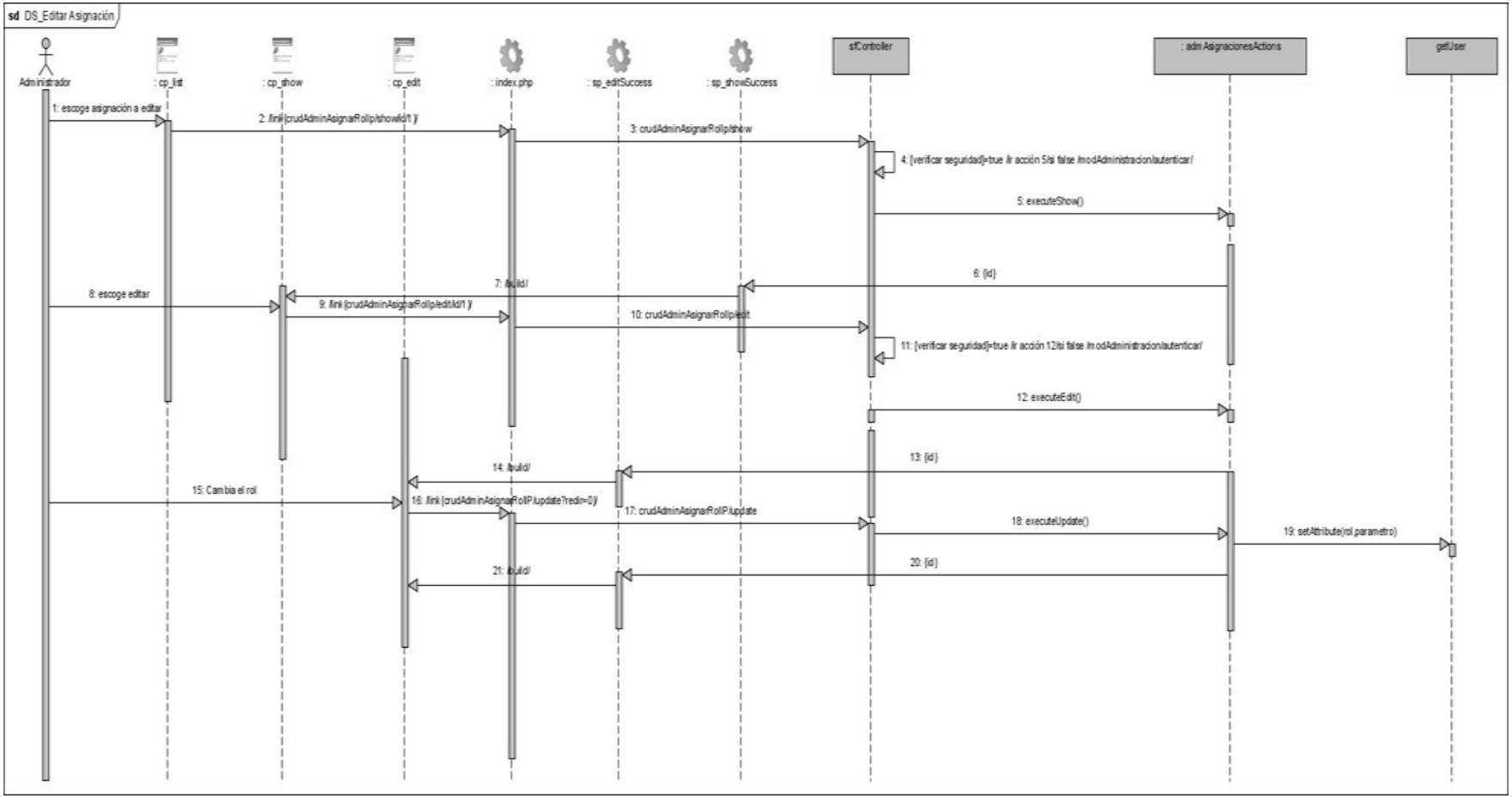

Figura 17 Diagrama de secuencia CU Gestionar Asignación (Editar Asignación).

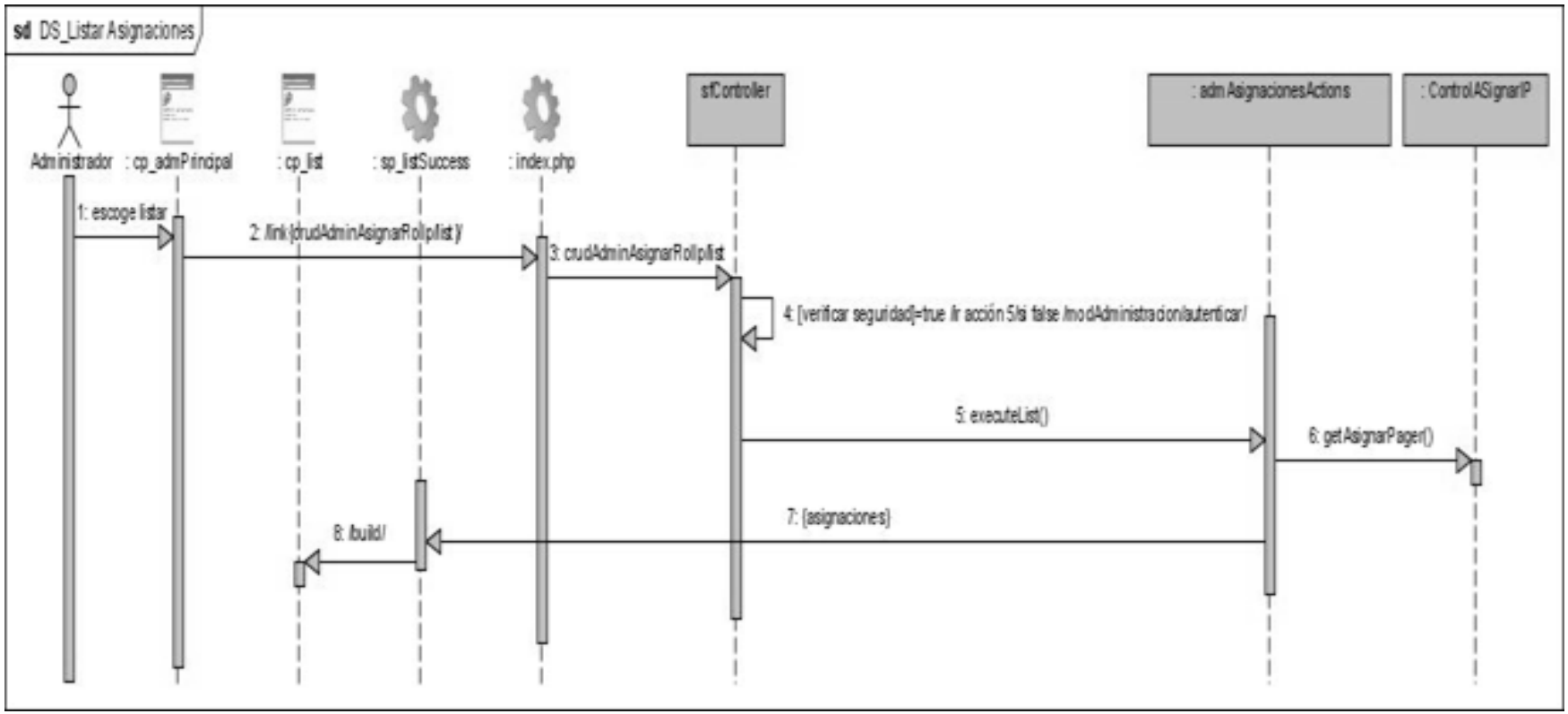

Figura 18 Diagrama de secuencia CU Gestionar Asignación (Listar Asignaciones).

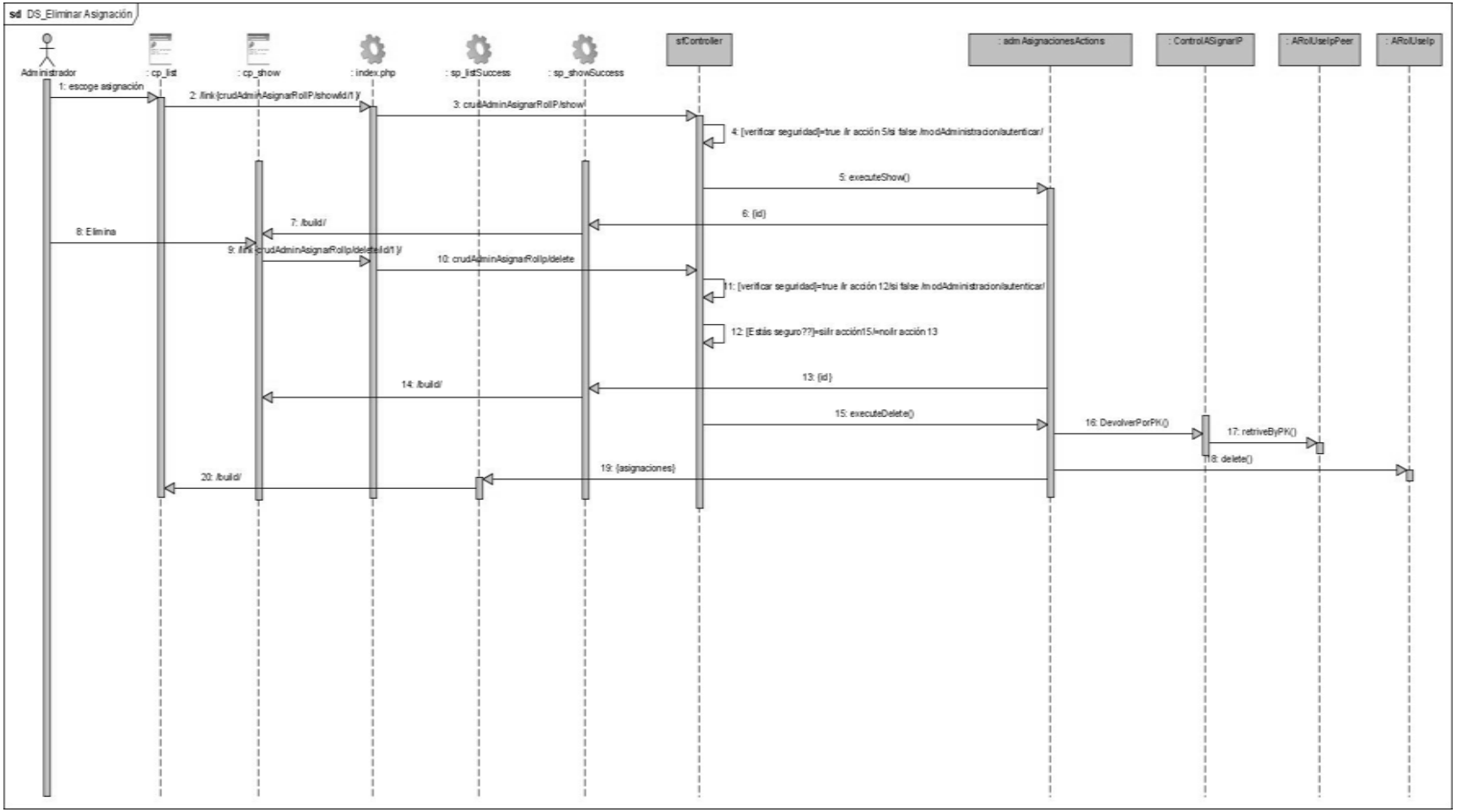

Figura 19 Diagrama de secuencia CU Gestionar Asignación (Eliminar Asignación).

2.6 Prototipos de Interfaz

Los prototipos de interfaz representan páginas que interactúan con el usuario en el intercambio de información, realizándose los mismos por cada escenario del caso de uso, a continuación se representan los referentes al caso de uso Autenticar.

Figura 20 Prototipo de interfaz CU Autenticar: Autenticarse.

Figura 21 Prototipo de interfaz CU Autenticar: Seleccionar Ensayo, Rol y Sitio.

2.7 Diagrama de clases persistentes.

A continuación se muestra la interacción entre las clases persistentes de la aplicación mediante un diagrama de clases persistentes.

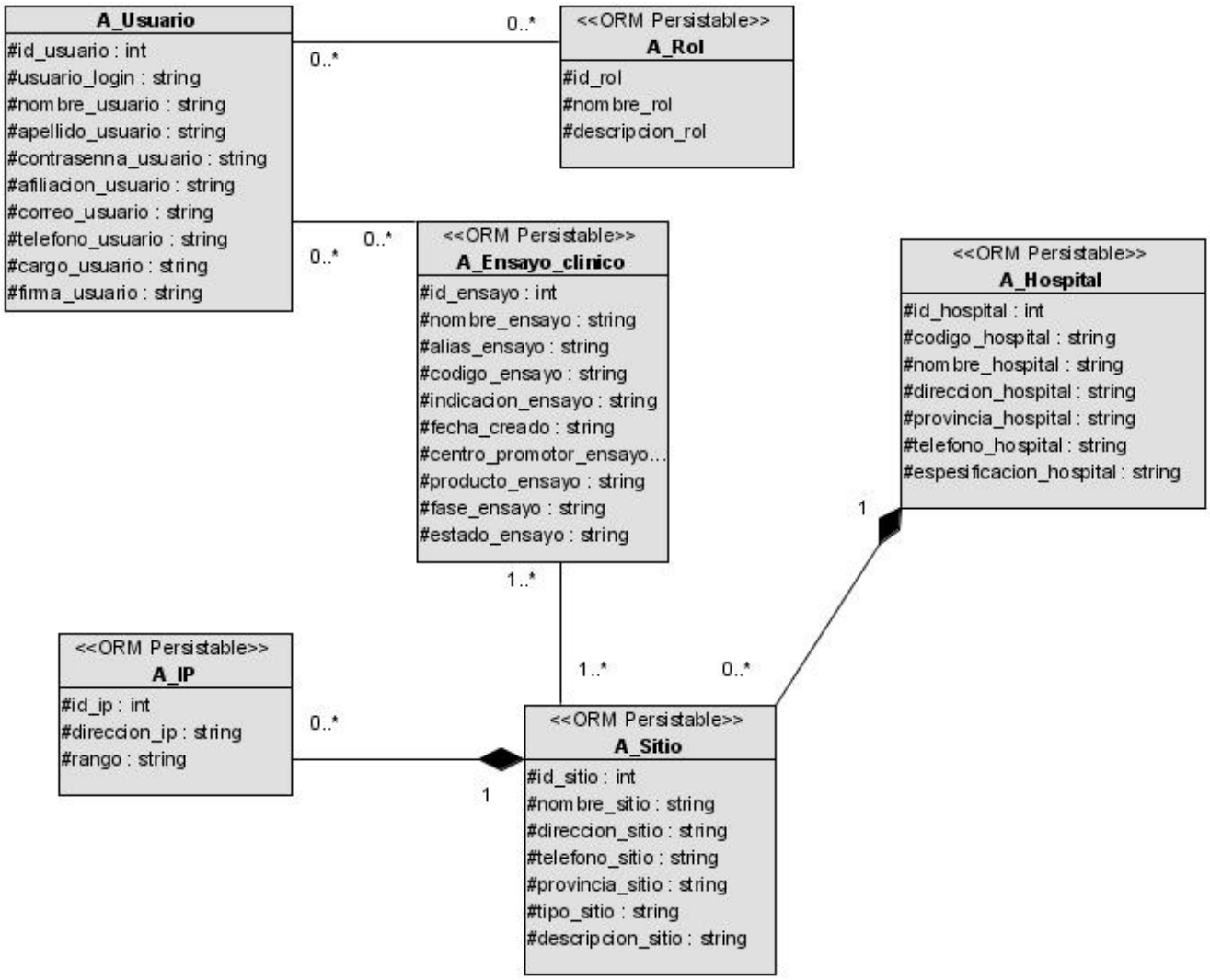

Figura 22 Diagrama de clases persistentes del módulo Administración.

2.8 Modelo de Datos

Este modelo representa a la realidad a través de un esquema gráfico empleando las terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas atributos, el enlace que rige la unión de las entidades está representada por la relación del modelo.

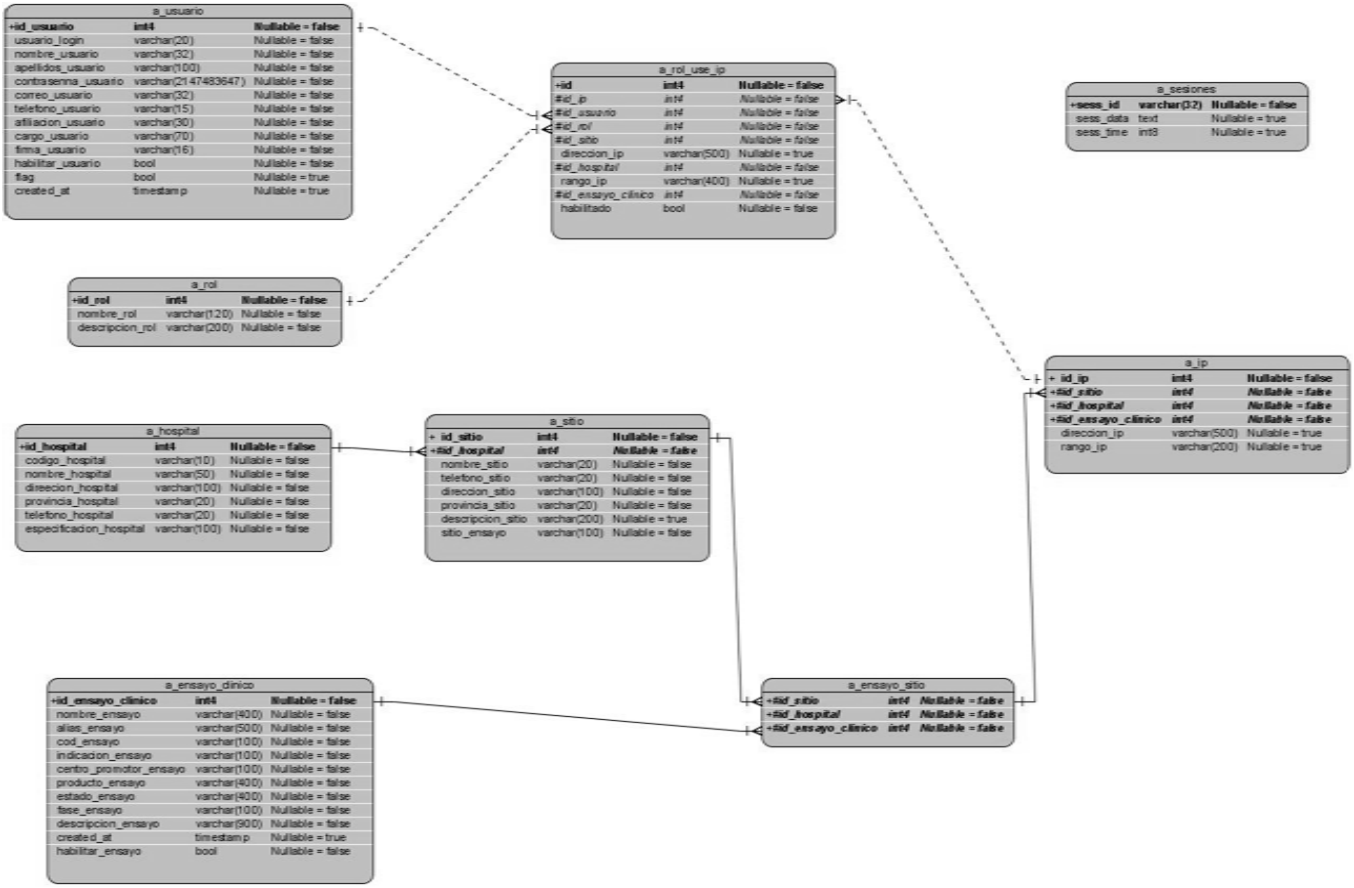

Figura 23 Modelo de datos del módulo Administración

2.9 Modelo de despliegue

El modelo de despliegue representa un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación.

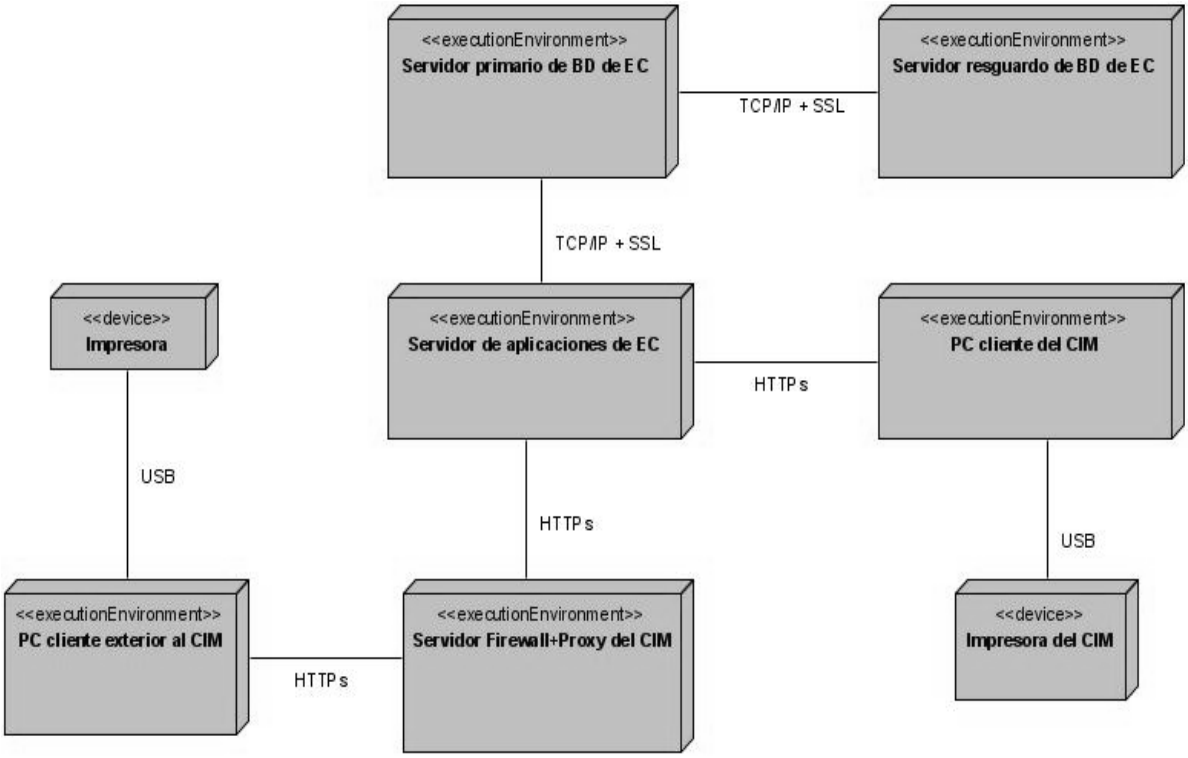

Figura 24 Modelo de despliegue

2.10 Mapa de navegación

El mapa de navegación refleja la representación gráfica de la organización de la información de una estructura Web. Expresa de una manera gráfica todas las relaciones entre las diferentes páginas del sitio Web para que se pueda entender de una mejor forma.

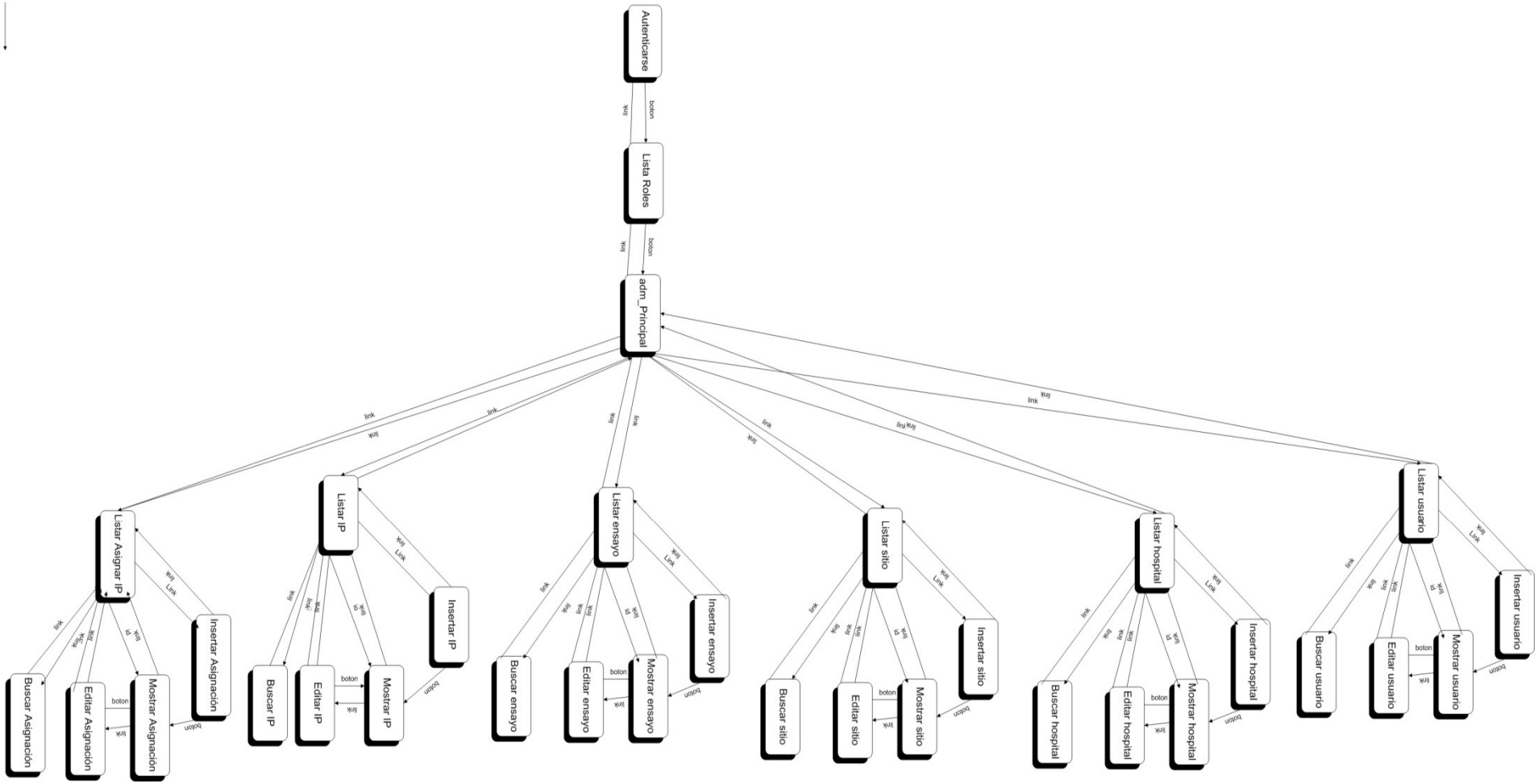

Figura 25 Mapa de navegación del módulo Administración.

2.11 Ejemplos de código

Código del método Llenarips() de la clase adm_Asignaciones

El sistema verifica que se hayan escogido todos los criterios, usuario, ensayo, sitio y rol, verifica que la asignación no existe en la base de datos y que el usuario pueda tener el rol escogido. Si la variable redir es igual 2 significa que el usuario insertó una dirección IP, el sistema verifica que no haya sido introducida anteriormente llamando al método ValidarIPBd() de la clase ControlAsignacirIP, llama a la función setAttribute() la cual guarda en una sesión la dirección IP, luego redirecciona a la página insertSuccess con el formulario LlenarIpsSuccess; si redir es igual 3 hace lo mismo que lo anterior lo que ahora el sistema procede a guardar un rango IP; si redir es igual 4 significa que el usuario escoge una dirección IP para eliminar, el sistema llama al método eliminar_ip() de la clase ControlAsignarIP y procede a eliminarlo; si redir es igual 5 el sistema procede a eliminar un rango IP llamando al método eliminar_rango() de la clase ControlAsignarIP.

```
public function executeLlenarips()
      {
            $usuario = $this->getRequestParameter('usuarios');
            $rol = $this->getRequestParameter('roles');
            $sitio = $this->getRequestParameter('sitios');
            $ensayo = $this->getRequestParameter('ensayos');

            if($usuario=="0" || $rol == -1 || $sitio == -1 || $ensayo == -1)
            {
                  $this->aux =1;
            }
            else
            {
                  $this->aux =0;
                  $ip = $this->getUser();
                  $this->var =0;
                  $id_sitio = $this->getRequestParameter('sitios');
                  $id_ensayo = $this->getRequestParameter('ensayos');
                  $this->sitio=$id_sitio;
                  $this->ensayo=$id_ensayo;

                  $patron_ip      =      "/^([1-9]|25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-
```

```
9])(\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])){3}$/";

                $obj_ip                                                    =
ControlAsignarIP::DameIPPorEnsSitio($id_sitio,$id_ensayo);

                $id_ip = $obj_ip[0]->getIdIp();
                $lista_ip = $obj_ip[0]->getDireccionIp();
                $lista_rango = $obj_ip[0]->getRangoIp();

                $this->ips =  $ip->getAttribute('asignar_ip');
                $this->rangos =  $ip->getAttribute('asignar_rango');
            }
            if($this->getRequestParameter('redir') == 1)
            {

if(!(ControlAsignarIP::ValidarAsignacion($usuario,$ensayo,$rol,$sitio)))
             $this->aux =2;
            else
if(!(ControlAsignarIP::ValidarRolesPorUsuario($usuario,$ensayo,$rol)))
             $this->aux =3;
                else
                {
                      $this->aux =0;
         $this->listaip = $obj_ip;
         $ip->setAttribute('user',$usuario);
         $ip->setAttribute('rol',$rol);
         $ip->setAttribute('sitio',$sitio);
         $ip->setAttribute('ensayo',$ensayo);
         $ip->setAttribute('ip',$id_ip);
                }
            }


            if($this->getRequestParameter('redir') == 2)
            {

                $this->listaip = $obj_ip;
```

```
                $dir_ip = $this->getRequestParameter('direccion_ip');
                if (!(preg_match($patron_ip, $dir_ip)))
                $this->var =1;
            else
if(!(ControlAsignarIP::validarIpBd($dir_ip,$lista_ip,$lista_rango)))
                $this->var =2;
                else          if(!(ControlAsignarIP::validarIpVariable($dir_ip,$ip-
>getAttribute('asignar_ip'),$ip->getAttribute('asignar_rango'))))
                $this->var =3;
                else
                {
                      if($ip->getAttribute('asignar_ip')==NULL)
                      {
                            $ip->setAttribute('asignar_ip',$dir_ip);
                            $this->ips =  $ip->getAttribute('asignar_ip');
                      }
                      else
                      {
                            $dir = $ip->getAttribute('asignar_ip');
                            $dir = ($dir.';'.$dir_ip);
                            $ip->setAttribute('asignar_ip',$dir);
                            $this->ips =  $ip->getAttribute('asignar_ip');
                      }
                }
            }
            if($this->getRequestParameter('redir') == 3)
            {
                  $this->listaip = $obj_ip;
                  $primer_rango = $this->getRequestParameter('primer_ip');
                  $segundo_rango = $this->getRequestParameter('segundo_ip');
                  $long_primer_rango = ip2long($primer_rango);
                  $long_segundo_rango = ip2long($segundo_rango);
            if (!(preg_match($patron_ip, $primer_rango)) || !(preg_match($patron_ip,
$segundo_rango)) )
                  $this->var =1;
             else if($long_primer_rango >= $long_segundo_rango)
                  $this->var =4;
```

```
            else
if(!(ControlAsignarIP::validarRangoBd($long_primer_rango,$long_segundo_rango,$lista_
rango)))
                  $this->var =5;
            else
if(!(ControlAsignarIP::validarRangoVariable($long_primer_rango,$long_segundo_rango,$
ip->getAttribute('asignar_ip'),$ip->getAttribute('asignar_rango'))))
                  $this->var =3;
                  else
                  {
                        if($ip->getAttribute('asignar_rango')==NULL)
                        {
                              $rang = ($primer_rango.'-'.$segundo_rango);
                              $ip->setAttribute('asignar_rango',$rang);
                              $this->rangos =  $ip->getAttribute('asignar_rango');
                        }
                        else
                        {
                              $rango = $ip->getAttribute('asignar_rango');
                              $rang = ($primer_rango.'-'.$segundo_rango);
                              $rango = ($rango.';'.$rang);
                              $ip->setAttribute('asignar_rango',$rango);
                              $this->rangos =  $ip->getAttribute('asignar_rango');
                        }
                  }
            }

      if($this->getRequestParameter('redir') == 4)
            {
                  $this->listaip = $obj_ip;
                  $ips            =            ControlAsignarIP::eliminar_ip($this-
>getRequestParameter('lista_ips'),$ip->getAttribute('asignar_ip'));
                  $ip->setAttribute('asignar_ip',$ips);
                  $this->ips =  $ip->getAttribute('asignar_ip');
            }

      if($this->getRequestParameter('redir') == 5)
```

```
            {
            $this->listaip = $obj_ip;
            $rango              =              ControlAsignarIP::eliminar_rango($this-
>getRequestParameter('lista_rango'),$ip->getAttribute('asignar_rango'));
                  $ip->setAttribute('asignar_rango',$rango);
                  $this->rangos =  $ip->getAttribute('asignar_rango');
            }


      }
```

Código del método Insertar() de la clase adm_Asignaciones

Este método guarda en la base de datos mediante la clase ARolUseIP los datos de la asignación escogidos por el administrador, estos son, usuario, ensayo, rol y sitio, se accede a la clase Sitio para obtener el hospital al que pertenece el mismo, y luego se obtienen las direcciones y/o rangos IPs guardados en la sesión por la función Llenarips(), todos estos datos forman una asignación la cual es almacenada en la base de datos al finalizar la función.

```
public function executeInsertar()

      {

            $ip = $this->getUser();

            if($ip->getAttribute('asignar_ip') == NULL && $ip-
>getAttribute('asignar_rango') == NULL)

            return $this->redirect('adm_Asignaciones/insert?var=2');

            else

            {

                  if($this->getRequestParameter('habilitar_asignacion') == true)

                  $habilitado = true;

                  else
```

```
$habilitado = false;



$a_rol_use_ip = new ARolUseIp();



$d = new Criteria();

$d->add(ASitioPeer::ID_SITIO, $ip->getAttribute('sitio'));

$sitio = ASitioPeer::doSelect($d);

$hospital = $sitio[0]->getIdHospital();



$a_rol_use_ip->setIdUsuario($ip->getAttribute('user'));

$a_rol_use_ip->setIdRol($ip->getAttribute('rol'));

$a_rol_use_ip->setIdIp($ip->getAttribute('ip'));

$a_rol_use_ip->setIdSitio($ip->getAttribute('sitio'));

$a_rol_use_ip->setIdHospital($hospital);

$a_rol_use_ip->setIdEnsayoClinico($ip->getAttribute('ensayo'));

$a_rol_use_ip->setDireccionIp($ip->getAttribute('asignar_ip'));

$a_rol_use_ip->setRangoIp($ip->getAttribute('asignar_rango'));

$a_rol_use_ip->setHabilitado($habilitado);

$a_rol_use_ip->save();

$ip->setAttribute('user',NULL);
```

```
                    $ip->setAttribute('rol',NULL);

                    $ip->setAttribute('ip',NULL);

                    $ip->setAttribute('sitio',NULL);

                    $ip->setAttribute('ensayo',NULL);

                    $ip->setAttribute('asignar_ip',NULL);

                    $ip->setAttribute('asignar_rango',NULL);

                    $aux=1;

                    return $this->redirect('adm_Asignaciones/show?id='.$a_rol_use_ip-
>getId());

                }

            }
```

Conclusiones

En el capítulo se realizó el diseño del sistema, obteniendo como resultados los diagramas de clases del diseño para la aplicación Web, los diagramas de secuencia, mostrando la interacción de los objetos en la aplicación a través del tiempo, se muestra el diagrama de clases persistentes. También se desarrolló el diagrama de despliegue donde quedaron modelados los nodos en los que se distribuye la aplicación, especificando las conexiones de red y protocolos que los unen, se hizo referencia y se realizó una pequeña descripción de la utilidad de cada uno de los patrones que se utilizarán para la construcción de la aplicación, divididos en dos grandes grupos, patrones de diseño y de arquitectura, al final del capítulo se representan dos de las funciones más complejas y útiles que se implementaron durante el desarrollo de la aplicación.

Conclusiones Generales

Como resultado del estudio de los procesos de gestión de información para los Sistemas de Manejo de Datos de Ensayos Clínicos y los resultados obtenidos en investigaciones precedentes se puede concluir que la presente investigación logró darle solución al objetivo propuesto; para alcanzar este resultado se desarrolló:

- Un estudio de software como Pivotal, CS - CoreSystem y OpenClinica, los cuales se encargan de la administración y gestión de ensayos clínicos. Teniendo en cuenta las funcionalidades que estos brindan y las necesidades imperantes en Cuba para el manejo de Ensayos Clínicos y algunas situaciones adversas que la compra de estos software pudiesen traerle al país, se decidió implementar el Sistema de Manejo de Datos de Ensayos Clínicos Cubano.
- Una exhaustiva investigación de cuales serían las tecnologías y herramientas adecuadas para el desarrollo de la aplicación, estas fueron, PHP como lenguaje de programación, PostgreSQL como gestor de base de datos, Symfony como marco de trabajo (framework) y otras explicadas en el documento.
- Los artefactos inherentes al modelo de diseño, es decir, diagramas de clases del diseño con las respectivas descripciones de clases, diagramas de interacción (secuencia), proporcionando así una visión detallada de las funcionalidades a cumplir por la aplicación, requisitos funcionales y casos de uso que la componen.
- La implementación de la aplicación obteniendo un producto funcional para el módulo Administración del Sistema de Manejo de Datos de Ensayos Clínicos Cubano que cumple con los requisitos identificados y satisfaciendo las necesidades del cliente, mostrando en el documento ejemplos de código importantes con sus respectivas descripciones.

Gracias al estricto cumplimiento de las tareas a desarrollar al inicio del presente trabajo se logró con éxito todo el proceso de diseño e implementación de la aplicación Web, llegando sin mayores contratiempos al objetivo principal que no era más que desarrollar un producto funcional que se encargara de la administración del Sistema de Manejo de Datos de Ensayos Clínicos Cubano.

Recomendaciones:

A partir del producto obtenido se recomienda:

- Realización de las pruebas que propone RUP como flujo de trabajo.
- Lograr una nueva versión del producto incluyéndole más funcionalidades para el módulo Administración.
- Extender la aplicación para Ensayos Clínicos que no solo realicen estudios sobre el cáncer.
- Realización de una ayuda para el módulo Administración.

Referencia Bibliográfica

1. **OMS**. Organización Mundial de la Salud. [En línea] [Citado el: 15 de Noviembre de 2007.]
http://www.who.int/mediacentre/factsheets/fs297/es/index.html.

2. **CIM.** *Propuesta de proyecto de Colaboración con la UCI.* Ciudad de la Habana : s.n., 2007.

3. **Pivotal Website**. [En línea] [Citado el: 6 de Diciembre de 2007.]
http://www.pivotal.es/extranet/servicios/proceso-de-datos-clinicos.

4. **Gómez, Juan Pablo.** Scribd. *Fundamentos de la Metodología RUP Rational Unified Process.* [En línea] 16 de Septiembre de 2007. [Citado el: 6 de Diciembre de 2007.]
http://www.scribd.com/doc/297224/RUP.

5. **Aurora, Vizcaino.** Una Herramienta CASE para ADOO: Visual Paradigm. [En línea] [Citado el: 20 de Febrero de 2008.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

6. **Apache**. *Apache.* [En línea] [Citado el: 23 de Febrero de 2008.] HYPERLINK
"http://linux.ciberaula.com/articulo/linux_apache_intro"
http://linux.ciberaula.com/articulo/linux_apache_intro .

7. **Kompozer**. *Kompozer.* [En línea] [Citado el: 23 de Febrero de 2008.]
http://www.proyectonave.es/productos/kompozer.

8. **Ernesto, Quñonez.** Introducción a PostgreSQL. [En línea] [Citado el: 23 de Febrero de 2008.]
http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

9. **¿Que es un framework?** *¿Que es un framework?* [En línea] [Citado el: 3 de marzo de 2008.]
http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

10. **LibrosWeb.es**. *Symfony.* [En línea] [Citado el: 3 de Marzo de 2008.]
http://www.librosWeb.es/symfony/capitulo1/symfony_en_pocas_palabras.html.

11. **Patrones de Diseño**. *Patrones de Diseño.* [En línea] [Citado el: 20 de Marzo de 2008.]
http://mit.ocw.universia.net/6.170/6.170/f01/pdf/lecture-12.pdf.

12. **David Rodríguez Luque, Daniel Rodríguez Luque.** *Herramienta para la generación de código de aplicaciones Web.* Ciudad de la Habana : s.n., 2007.

13. **Teleformación**. Material_de_apollo_Conferencia_Diseño. [En línea] [Citado el: 11 de Junio de 2008.] http://teleformacion.uci.cu/mod/resource/view.php?id=21363.

Bibliografía

1. **Organización Mundial de la Salud**. [En línea] 15 de Noviembre de 2007. http://www.who.int/mediacentre/factsheets/fs297/es/index.html.

2. **CIM.** *Propuesta de proyecto de Colaboración con la UCI.* Ciudad de la Habana : s.n., 2007.

3. **Pivotal Website**. [En línea] 6 de Diciembre de 2007. http://www.pivotal.es/extranet/servicios/proceso-de-datos-clinicos..

4. **Gómez, Juan Pablo.** Scribd. *Fundamentos de la Metodología RUP Rational Unified Process.* [En línea] 16 de Septiembre de 2007. http://www.scribd.com/doc/297224/RUP.

5. **Aurora, Vizcaino.** Una Herramienta CASE para ADOO: Visual Paradigm. [En línea] 20 de Febrero de 2008. http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

6. **Apache**. *Apache.* [En línea] 23 de Febrero de 2008. http://linux.ciberaula.com/articulo/linux_apache_intro .

7. **Kompozer**. *Kompozer.* [En línea] 20 de Febrero de 2008. http://www.proyectonave.es/productos/kompozer.

8. **Ernesto, Quñonez.** Introducción a PostgreSQL. [En línea] 5 de Febrero de 2008. http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

9. **¿Que es un framework?** *¿Que es un framework?* [En línea] 3 de marzo de 2008. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

10. **LibrosWeb.es**. *Symfony.* [En línea] 3 de Marzo de 2008. http://www.librosWeb.es/symfony/capitulo1/symfony_en_pocas_palabras.html.

11. **Patrones de Diseño**. *Patrones de Diseño.* [En línea] 20 de Marzo de 2008. http://mit.ocw.universia.net/6.170/6.170/f01/pdf/lecture-12.pdf.

12. **David Rodríguez Luque, Daniel Rodríguez Luque.** *Herramienta para la generación de código de aplicaciones Web.* Ciudad de la Habana : s.n., 2007.

13. **Teleformación**. Material_de_apollo_Conferencia_Diseño. [En línea] 11 de Junio de 2008. http://teleformacion.uci.cu/mod/resource/view.php?id=21363.

14. **eMedical**. *CS-Core System.* [En línea] [Citado el: 3 de Febrero de 2008.] https://www.emedical.com.mx/flash/main2.swf.

15. **OpenClínica**. *OpenClínica.* [En línea] [Citado el: 4 de Febrero de 2008.] http://www.openclinica.org.

16. **Eclipse**. *Eclipse.* [En línea] [Citado el: 20 de Febrero de 2008.] http://www.eclipse.org.

17. **Aidacelys López Díaz, Lucia Rodriguez Garcia.** *Sistema de Manejo de Datos de Ensayos Clínicos: Módulo de Diseño. Capítulo 1. Herramientas para el control de versiones.* Ciudad de la Habana : s.n., 2007.

18. **Servidor Ensayos Clínicos**. Documentación. [En línea] http://10.34.17.251:3389/server/index.php?option=com_wrapper&Itemid=40 Ciudad de la Habana : s.n., 2008.

Glosario de Términos

Biomoléculas: Moléculas que conforman el cuerpo humano. Ejemplo el carbono, hidrógeno, oxígeno, nitrógeno.

Software: Aplicación informática.

Caso de uso: Es una técnica para la captura de requisitos potenciales de sistema.

Prototipo no funcional: Interfaces o páginas Web que se le muestran al usuario para que interactúe con ella pero que no tienen funcionalidad ninguno con algún gestor de base de datos u otro sistema.

CRD: (Cuaderno de Recogida de Datos) Documento impreso en el que se recoge toda la información referente a los pacientes en el Ensayo Clínico.

Ensayo Clínico: Tipo de estudio clínico que conforma una de las fases del proceso de desarrollo de los fármacos.

Agencia Regulatoria: Centro donde aprueban un Ensayo Clínico.

IP: Protocolo de internet. Está conformado por números que identifican de manera lógica y jerárquica a una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol).

PL/SQL: Lenguaje de programación embebido en gestores de base de datos como Oracle y PostgreSQL. El PL/SQL soporta todas las consultas y manipulación de datos que se usan en SQL.

HTML: HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

CSS: (Cascading Style Sheets, u Hojas de Estilo en Cascada) es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación. Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

C++: Lenguaje de programación para construir aplicaciones de escritorio.

XML: (del inglés Extensible Markup Language) Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un

papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

HTTP: El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol o Protocolo de transferencia de HiperTextos) es el protocolo usado en cada transacción de la Web. Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura Web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

HTTPS: Hypertext Transfer Protocol Secure (en español: Protocolo seguro de transferencia de hipertexto), mas conocido por su acrónimo HTTPS, es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

Licencia BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

Licencia GPL: Es una licencia creada por la Free Software Foundation (Fundación de software libre) a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Logs: Registro oficial de eventos durante un período de tiempo en particular. Para los profesionales en seguridad informática un log es usado para registrar datos o información sobre quien, que, cuando, donde y por que un evento ocurre para un dispositivo en particular o aplicación.

AJAX: Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

W3C: Consorcio World Wide Web. Es un consorcio internacional donde las organizaciones miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web.

Anexos

1. Diagramas de clases del diseño

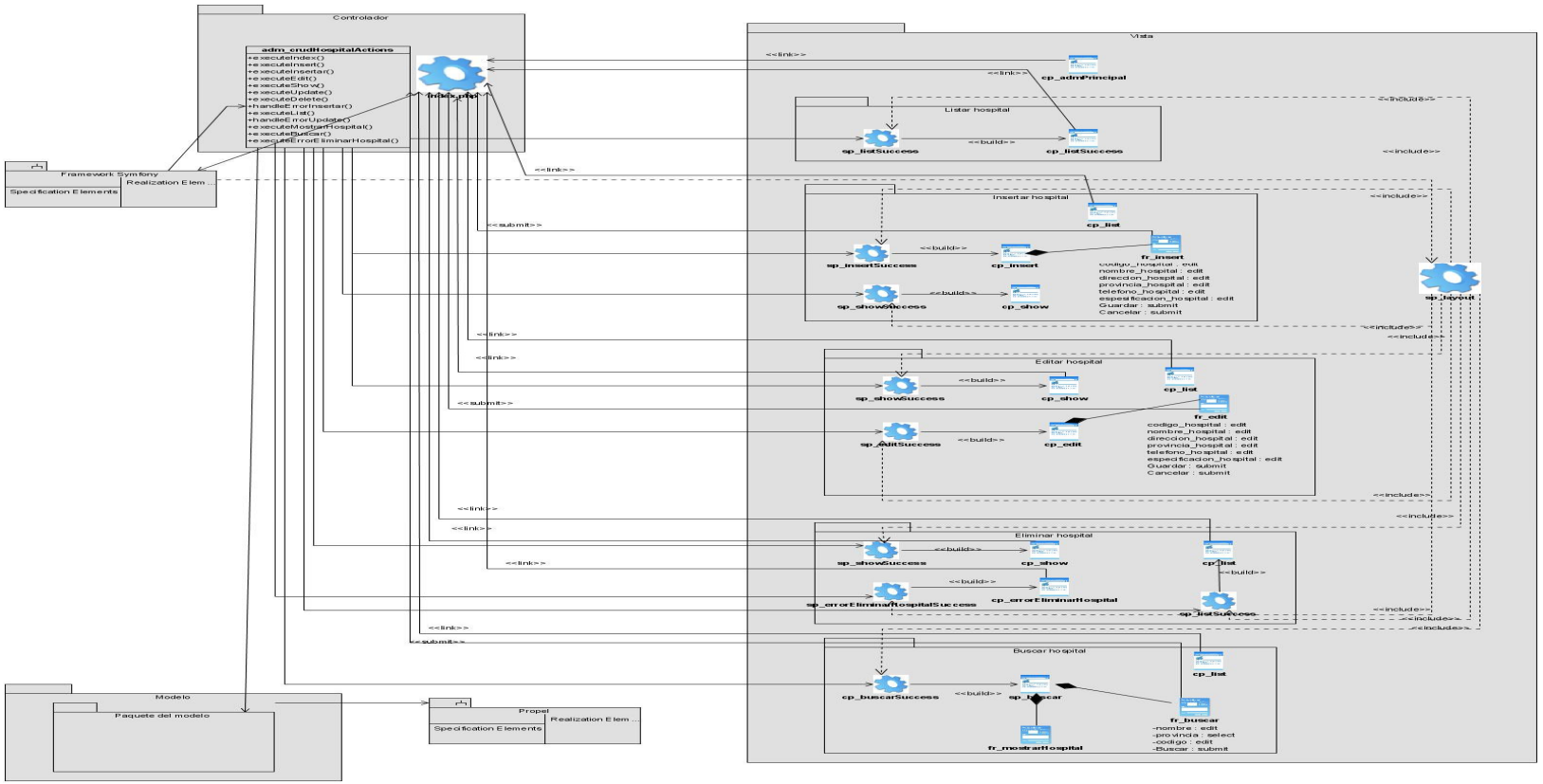

Figura 1 Diagrama de clases del diseño CU: Gestionar Hospital

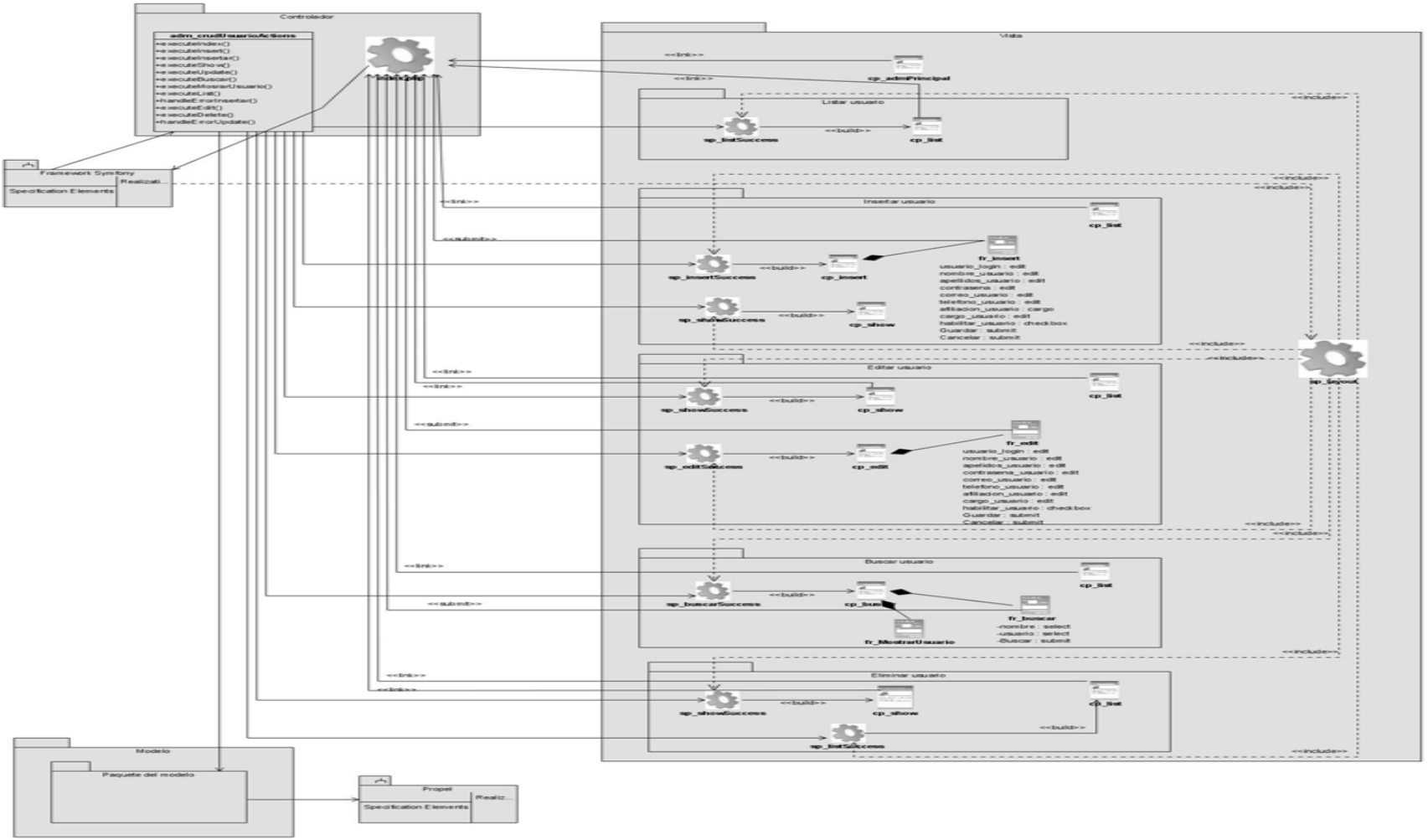

Figura 2 Diagrama de clases del diseño CU: Gestionar Usuario

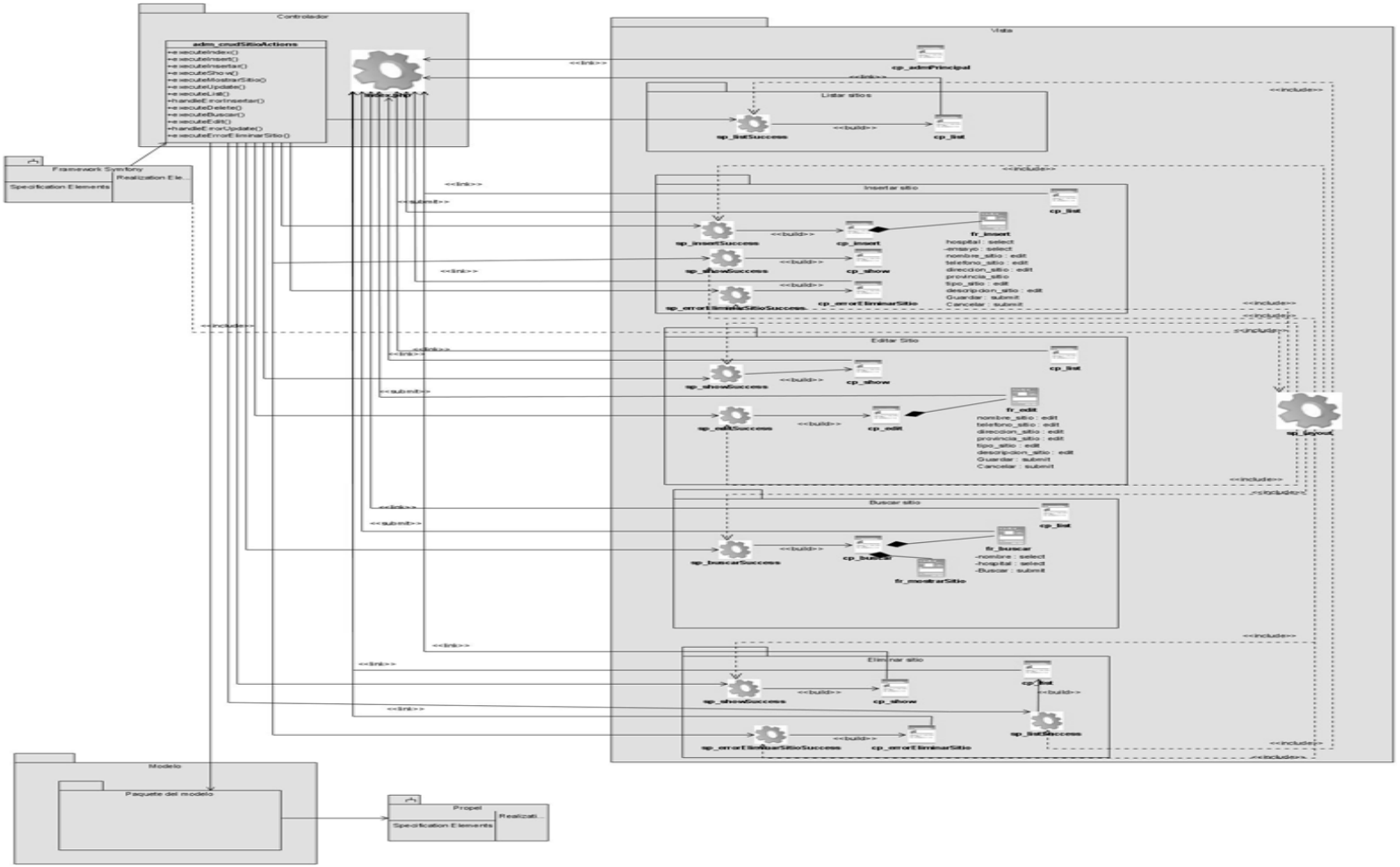

Figura 3 Diagrama de clases del diseño CU: Gestionar Sitio

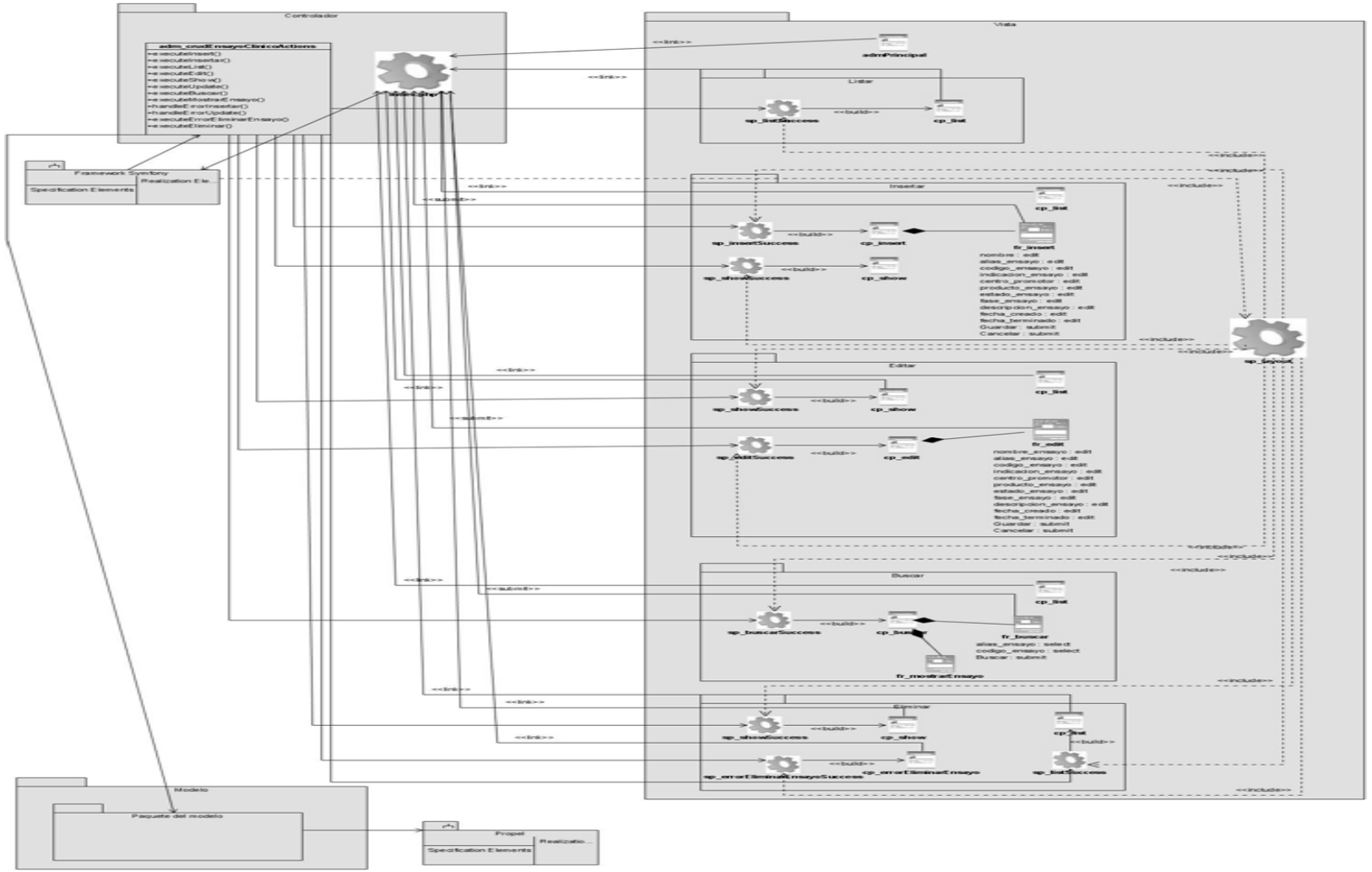

Figura 4 Diagrama de clases del diseño CU: Gestionar Ensayo Clínico

2 Descripción de las clases del diseño

Tabla 2.18 Descripción de la clase del diseño: adm_crudHospitalActions

Nombre: adm_crudHospitalActions	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Gestionar Hospital, llama a la Clase ControlHospital para utilizar sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de hospitales existentes en la base de datos.
executeList()	Llama a la función getHospitalPager() de la clase ControlHospital y muestra en la Client page correspondiente la lista de hospitales registrados en la base de datos.
executeShow()	Llama a la función DevolverPorPK() de la clase ControlHospital y muestra en la Client Page correspondiente los datos de un hospital seleccionado por el administrador; pero no puede modificarlos.
executeInsert()	Muestra en la Client Page correspondiente un formulario para insertar un hospital en la base de datos.
executeInsertar()	Llama a la función Insertar() de la clase ControlHospital y en caso de no haber errores inserta los datos y redirecciona a la página donde se muestran todos los hospitales que existen en la base de datos, sino lanza una excepción.
handleErrorInsertar()	Redirecciona a la página de insertar hospital en caso de dejar campos vacios a la hora de insertarlos.
executeEdit()	Llama a la función DevolverPorPK() de la clase ControlHospital y muestra en la Client Page correspondiente los datos de un hospital seleccionado por el administrador con posibilidad de modificarlos.
executeUpdate()	Llama a la función DevolverPorPK() de la clase ControlHospital, modifica los valores y redirecciona a la

	página donde muestra los datos del hospital modificado.
executeDelete()	Llama a la función DevolverPorPK() de la clase ControlHospital, elimina el hospital y redirecciona a la página donde se muestra la lista de hospitales que existe en la base de datos.
executeAutocompletar()	Función para completar una palabra o frase en un edit según las letras que se conozcan de la misma, en este caso para el nombre y el código del sitio.
executeMostrarHospital()	Recoge el código, el nombre y la provincia de un hospital, comprueba que no haya errores de entrada y Llama a la función MostrarHospital() de la clase CotrolHospital para hacer la búsqueda.

Tabla 2.19 Descripción de la clase del diseño: cadm_crudUsuarioActions

Nombre: adm_crudUsuarioActions	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Gestionar Usuario, llama a la Clase ControlUsuario para utilizar sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de usuarios existentes en la base de datos.
executeList()	Llama a la función getUsuarioPager() de la clase ControlUsuario y muestra en la Client Page correspondiente la lista de usuarios registrados en la base de datos.
executeShow()	Llama a la función DevolverPorPK() de la clase ControlUsuario y muestra en la Client Page correspondiente los datos de un usuario seleccionado por el administrador; pero no puede modificarlos.
executeInsert()	Muestra en la Client Page correspondiente un formulario para insertar un usuario en la base de datos.

executeInsertar()	Llama a la función Insertar() de la clase ControlUsuario y en caso de no haber errores inserta los datos y redirecciona a la página donde se muestran todos los usuarios que existen en la base de datos, sino lanza una excepción.
handleErrorInsertar()	Redirecciona a la página de insertar usuario en caso de dejar campos vacios a la hora de insertarlos.
executeEdit()	Llama a la función DevolverPorPK() de la clase ControlUsuario y muestra en la Client Page correspondiente los datos de un usuario seleccionado por el administrador con posibilidad de modificarlos.
executeUpdate()	Llama a la función DevolverPorPK() de la clase ControlUsuario, modifica los datos y redirecciona a la página donde muestra los datos del usuario modificados.
executeDelete()	Llama a la función DevolverPorPK() de la clase ControlUsuario, elimina al usuario y redirecciona a la página donde se muestra la lista de usuarios que existe en la base de datos.
executeBuscar()	Llama a la función Buscar() de la clase ControlUsuario y muestra en la Client Page correspondiente un formulario para hacer una búsqueda avanzada de usuarios a criterio del administrador.
executeMostrarUsuario()	Llama a la función MostrarUsuarios() de la clase ControlUsuario y muestra en la Client Page correspondiente a executeBuscar(), utilizando AJAX, los usuarios de acuerdo al criterio seleccionado por el administrador.
executeAutocompletar	Función para completar una palabra o frase en un edit según las letras que se conozcan de la misma, en este caso para el usuario, el nombre, los apellidos y la afiliación de un usuario.

Tabla 2.20 Descripción de la clase del diseño: adm_crudSitioActions

Nombre: crudAdminSitioActions	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al paquete Gestionar Sitios, llama a la Clase ControlSitio, Controlhospital y ControlEC para utilizar sus funciones.)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
executeIndex()	Redirecciona a la página que muestra el listado de sitios existentes en la base de datos.
executeList()	Llama a la función getSitioPager() de la clase ControlSitio y muestra en la Client Page correspondiente la lista de sitios registrados en la base de datos.
executeShow()	Llama a la función DevolverPorPK() de la clase ControlSitio y muestra en la Client Page correspondiente los datos de un sitio seleccionado por el administrador; pero no puede modificarlos.
executeInsert()	Llama a la función Buscar() de la clase ControlHospital y Buscar_Habilitados() de la clase ControlEC, de donde obtienes los hospitales y los ensayos que existen y luego muestra en la Client Page correspondiente un formulario para insertar un usuario en la base de datos.
executeInsertar()	Llama a la función DevolverPorPK() de la clase ControlEC y crea un objeto Ensayo Clínico utilizándolo para generar un código entre el código del ensayo y el nombre del sitio, luego llama a Insertar() de la clase ControlSitio y en caso de no haber errores inserta los datos y redirecciona a la página donde se muestran todos los sitios que existen en la base de datos, sino lanza una excepción.
handleErrorInsertar()	Redirecciona a la página de insertar sitio en caso de dejar campos vacios a la hora de insertarlos.
executeEdit()	Llama a la función DevolverPorPK() de la clase ControlSitio y muestra en la Client Page correspondiente los datos de un

	sitio seleccionado por el administrador con la posibilidad de modificarlos.
executeUpdate()	Llama a la función DevolverPorPK() de la clase ControlSitio, modifica los datos y redirecciona a la página donde muestra los datos del sitio insertado.
executeDelete()	Llama a la función DevolverPorPK() de la clase ControlSitio, elimina los datos y redirecciona a la página donde se muestra la lista de sitios que existe en la base de datos.
executeBuscar()	Llama a la función Buscar() de la clase ControlSitio y muestra en la Client Page correspondiente un formulario para hacer una búsqueda avanzada de sitios a criterio del administrador.
executeMostrarSitio()	Crea objetos ensayo y hospital y llama a la función MostrarSitios() de la clase ControlSitio y muestra en la Client Page correspondiente a executeBuscar(), utilizando AJAX, los sitios de acuerdo al criterio seleccionado por el administrador.
executeAutocompletar()	Función para completar una palabra o frase en un edit según las letras que se conozcan de la misma, en este caso para el nombre de un sitio, de un hospital o de un ensayo clínico.

Tabla 2.21 Descripción de la clase del diseño: ControlHospital

Nombre: ControlHospital	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar un hospital, utiliza la clase AHospitalPeer)	
Atributo	**Tipo**
Responsabilidades	
Nombre:	**Descripción:**
getHospitalPager()	Devuelve los hospitales almacenados en la base de datos y en caso de que sean muchos hace una paginación de los mismos.

DevolverPorPK()	Llama a la función retriveByPK() de la clase AHospitalPeer y devuelve un objeto hospital según la llave primaria que se le pasó por parámetro.
Insertar()	Hace una consulta a la base de datos para comprobar si ya existe o no el hospital que se desea insertar devolviendo verdadero o falso según sea el caso.
ParcheAutoincrementar()	Hace una consulta a la base de datos y determina la cantidad de hospitales que existen y luego devuelve un entero = cantidad de hospitales + 1.
Buscar()	Devuelve todos los hospitales que hay en la Base de datos
MostrarHospital()	Devuelve un objeto hospital dado un criterio de búsqueda en la base de datos