Universidad de las Ciencias Informáticas

Facultad 6

SIMDECC

Título: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Módulo Publicador, diseño e implementación del submódulo "Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos".

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

Autores: **Kenia Rodríguez Clavijo**

Daili Segura Durán

Tutores: **Ing. Andrés Ballester Marsal**

Ing. Elvira López Santos

Junio, 2008

"Año 50 de la Revolución."

"La sabiduría suprema, es tener sueños lo suficientemente grandes,

como para no perderlos de vista mientras los persigues."

William Fulker

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Kenia Rodríguez Clavijo	Ing. Andrés Ballester Marsal
----------	----------
Daili Segura Durán	Ing. Elvira López Santos
----------	----------

Datos de Contacto

Ing. Andrés Ballester Marsal:

El compañero Andrés Ballester, actual tutor del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Módulo Publicador, diseño e implementación del submódulo “Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos, fue graduado en la Universidad de las Ciencias Informáticas con el título de Ingeniero en Ciencias Informáticas en el año 2007, actualmente se desempeña como profesor de Gráficos por Computadoras en la universidad donde se graduó y como arquitecto en el proyecto Ensayos Clínicos al cual pertenece.

Ing. Elvira López Santos:

La compañera Elvira López, actual tutora del trabajo de diploma: Sistema de Manejo de Datos de Ensayos Clínicos Cubano: Módulo Publicador, diseño e implementación del submódulo “Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos, fue graduada en la Universidad de las Ciencias Informáticas con el título de Ingeniera en Ciencias Informáticas en el año 2007, actualmente se desempeña como profesora de Práctica Profesional en la universidad donde se graduó y como analista y jefa del módulo Publicador del proyecto Ensayos Clínicos al cual pertenece.

Agradecimientos

Ballester, queremos darte las gracias por ser el mejor de los tutores, por orientarnos, guiarnos y pasar tantas malas noches junto a nosotras. Por tus consejos constructivos y tu infinita paciencia.

Queremos agradecerle además a los compañeros del proyecto Ensayos Clínicos que contribuyeron día a día a aumentar nuestros conocimientos y convertirnos en mejores estudiantes. A Richard y Osvaldo por ser tan atentos y dispuestos a ayudar, a Yordel y Erislán, por hacer a un lado importantes tareas y sentarse junto a nosotras para ayudarnos a desarrollar algún algoritmo. A todos en general por querernos y hacernos más fácil la vida, Muchas Gracias.

Kenia Rodríguez Clavijo:
Oda y Ema: Gracias por estar siempre dispuestas a ayudar, por asumir esta tesis como suya y trabajar codo con codo junto a mí. Por pasar días enteros debatiendo sobre patrones y por amenizarme la vida. Quiero agradecerles más que por ayudarme en la tesis, por formar parte de mi vida, por estar presente en cada uno de los momentos que viví en la UCI. Gracias por aceptarme como soy y saber levantarme el ánimo. Gracias por ser especiales.

Yuneimy y Lorna: Gracias por enseñarme que amigo no es cualquiera, por estar presente en mi vida y darme fuerzas día a día.

Negrito: Gracias por iniciarme como programadora y por confiar en mí, por estar siempre atento y tener a mano un gracioso chiste para acompañar nuestros almuerzos y nuestras noches de películas.

A las niñas del apartamento: Por hacerme la vida más llevadera, por aguantar mis resabios y ver el televisor sin apenas escucharlo, por cuidarme el sueño. Estos han sido los mejores años de mi vida, espero que no me olviden.

Para concluir le doy las gracias a Maciel y Juanita por no decir nunca no, y estar siempre dispuestas a ayudarme, corregirme y enseñarme.

Daili Segura Durán:

Le agradezco a mi novio el haberme guiado para que estudiara, para que aprovechara mejor el tiempo. Su cariño y su amor me ha dado aliento a seguir adelante; su perseverancia, por brindarme su casa y su familia, para que no me sintiera sola, para que no sufriera por estar lejos de los míos.

A mis amistades que me aguantaron los 5 años mis malcriadeces, mis llantos cuando me he sentido triste, vulnerable, me cuidaron y apoyaron en todo, tanto en la vida cotidiana como en los estudios. Les agradezco todo lo que hicieron por mí, en estos años, por haberme hecho reír y llorar con sus crudas verdades. Marbelis, Magdelis, Lorna y Annia les agradezco cada minuto que me dejaron compartir con ellas sus vidas, en la alegría y en tristeza, en la pobreza y en la riqueza, en la salud y en la enfermedad.

También quiero agradecerle a toda mi familia a mi papá, a mi tía Mari, a mi padrastro, a mi cuñado Ernesto, que fue también un espejo, por darme apoyo y por preocuparse por mi, pero en especial a dos personas que ya no están con nosotros, que son mi abuela Elena y Adolis Dumois, que con su experiencia y sabiduría supieron guiarme y enseñarme el camino correcto, pasarse horas dando consejos ayudándome a entender lo importante que es sacrificarse en la vida y tener una profesión, alentándome a seguir adelante y a romper todos los obstáculos que se me presentaran por delante.

Dedicatoria

A mis padres por ser el más perfecto reflejo de la bondad infinita, por tantos años de sacrificio y esfuerzo desmedido. Por tantas noches de insomnio, por estar ahí siempre para cuidarme, eliminar mis miedos y hacerme una mujer de bien. Por ser la guía y el aliento de mi vida.

A mis hermanas por apoyarme siempre, comprenderme y compartir tanta dulzura. Por dejar atrás sus intereses personales y sacrificarse tanto para lograr que este sueño, hoy se materialice.

A mi novio, por decir siempre las palabras correctas en el momento preciso, por no dejar que me derrumbara, por ser mí soporte, mi espíritu, por enseñarme que podemos ser mejores cada día y que todo conocimiento adquirido es poco.

A mis sobrinos por ser la alegría de mi vida y por dibujarme una sonrisa en los peores momentos.

A mi familia en general y a aquellos que aun sin serlo me tienen tanta estima, por dedicarme tantos momentos y darme consejos sabios.

Kenia Rodríguez Clavijo

Le dedico con todo el amor del mundo, la tesis a mi mamá no solo por haberme dado la vida, sino también por ser la mejor madre del mundo, porque todo lo que hace, lo hace pensando en sus hijos, por mimarme, por consentirme, por cuidarme todos estos años, por quererme y darme mucho amor todos los días, por sus noches de insomnios cuidándome cuando he estado enferma, porque no se separa de mi lado hasta que este bien, por creer en mí y darme aliento cuando estoy triste y susceptible. Porque lo que soy actualmente, es gracias a ella, todo lo que tengo en la vida se lo debo a ella, cuando pienso que no voy a poder o que esta muy lejos para mí, me viene ella a la mente y me da valor, fuerzas para continuar, para seguir adelante porque es un ejemplo como persona para mí.

A mi hermana por ser más que una hermana, mi madre, mi amiga, toda mi vida gira a su alrededor, ella es el ombligo de mi mundo. La que cuando tengo problemas acudo a ella por consejos, para que me guíe, me enseñe el camino que debo seguir hacia adelante para no tropezar. Ella es mi patrón, mi guía en la vida, mi apoyo. Con su dulzura, amor, besos, abrazos, consejos y regaños, ha sabido cuidarme, enseñarme casi todo sobre la vida.

Daili Segura Durán

Resumen

La presente investigación surge a partir del trabajo realizado en el proyecto Ensayos Clínicos (EC). El proyecto fue propuesto por el Centro de Inmunología Molecular (CIM) y desarrollado en la Universidad de las Ciencias Informáticas (UCI) con el propósito de desarrollar un Sistema de Manejo de Datos de Ensayos Clínicos para Cuba (SIMDECC). Dicho sistema consta de 5 módulos, en este trabajo de diploma se le da continuidad a una investigación precedente que estuvo centrada en el módulo Publicador. Los resultados anteriormente obtenidos fueron: El modelamiento del negocio, el levantamiento de requisitos y una propuesta del diseño, recomendando la implementación. El módulo Publicador, por su complejidad, fue dividido en dos submódulos. El presente trabajo estará centrado en el submódulo Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos (CRD). Al mismo será necesario realizarle un nuevo diseño fundamentado en cambios realizados en la arquitectura del proyecto. Además de realizarle la correspondiente implementación para obtener un producto funcional que reduzca el costo en tiempo y esfuerzo de los procesos de realización de EC a nivel nacional, aumentando la calidad del proceso, incrementando la integridad de los datos obtenidos e implicando una menor movilización de recursos por parte del centro promotor.

Índice

Agradecimientos I
Dedicatoria III
Resumen IV
Introducción 1
Capítulo 1: Fundamentación Teórica 6
1.1 ¿Qué es un Ensayo Clínico? 6
1.2 ¿Qué es un CRD? 6
1.3 Módulo Publicador. 6
1.4 Sistemas automatizados vinculados al campo de acción. 8
1.4.1 Pivotal. 8
1.4.2 Hipócrates. 9
1.4.3 OpenClínica. 10
1.5 Metodología de Desarrollo de Software y lenguaje de modelado. 12
1.5.1 Metodología de desarrollo. 12
1.5.2 Lenguaje de modelado. 14
1.6 Tecnologías. 15
1.6.1 Servidor Web. 15
1.6.2 Gestor de Base de Datos. 16
1.6.3 Lenguaje de programación. 18
1.7 Herramientas a utilizar. 18
1.7.1 Herramienta CASE para modelación del sistema. 18
1.7.2 Framework. 19
1.7.3 Entorno de Desarrollo Integrado. 21
1.7.4 Editores de interfaz Web. 22
1.7.5 Controlador de Versiones. 22
1.8 Conclusiones. 23
Capítulo 2: Solución propuesta. 24
2.1 ¿Qué es el diseño? 24

2.1.1 Propósitos y objetivos del diseño. 24
2.1.2 Estereotipos del diseño Web. 25
2.2 Patrones. 26
2.2.1 Patrones aplicados en el diseño: 26
2.2.2 Patrones arquitectónicos: 30
2.3 Paquetes del diseño. 31
2.4 Clases del diseño. 33
2.4.1 Diagramas clases del diseño. 34
2.4.2 Descripción de las clases del diseño 39
2.5 Diagramas de interacción del diseño. 46
2.6 Mapa de Navegación. 61
2.7 Prototipos de interfaz de usuarios. 62
2.8 Modelo de datos. 62
2.9 Diagrama de Despliegue. 65
2.10 Implementación. 66
2.11 Conclusiones 68
Conclusiones 69
Recomendaciones 70
Referencias Bibliográficas. 71
Bibliografía. 73
Anexos. 74
Glosario 98

Índice de imágenes.

Fig. 1 Origen de Symfony 20
Fig. 2 Estereotipo de la página servidora 26
Fig. 3 Estereotipo de la página cliente 26
Fig. 4 Estereotipo del formulario 26
Fig. 5 Paquetes del diseño 32

Fig. 6 Diagrama de clases del diseño del caso de uso Insertar Paciente Ensayo 34
Fig. 7 Diagrama de clases del diseño de los casos de uso Insertar Datos Modelos Paciente, Visualizar Modelo y Modificar Datos Modelos 35
Fig. 8 Diagrama de clases del diseño del caso de uso Visualizar Cronograma 36
Fig. 9 Diagrama de clases del diseño del caso de uso Listar Investigadores 37
Fig. 10 Paquete Modelo 38
Fig. 11 Diagrama de secuencia del caso de uso Insertar Paciente Ensayo escenario 1 47
Fig. 12 Diagrama de secuencia del caso de uso Insertar Paciente Ensayo escenario 2 48
Fig. 13 Diagrama de secuencia del caso de uso Insertar Paciente Ensayo escenario 3 49
Fig. 14 Diagrama de secuencia del caso de uso Insertar Datos Modelos Paciente escenario 1 50
Fig. 15 Diagrama de secuencia del caso de uso Insertar Datos Modelos Paciente escenario 2 51
Fig. 16 Diagrama de secuencia del caso de uso Insertar Datos Modelos Paciente escenario 3 52
Fig. 17 Diagrama de secuencia del caso de uso Visualizar Datos Modelo escenario 1 53
Fig. 18 Diagrama de secuencia del caso de uso Visualizar Datos Modelo escenario 2 54
Fig. 19 Diagrama de secuencia del caso de uso Modificar Datos Modelo escenario 1 55
Fig. 20 Diagrama de secuencia del caso de uso Modificar Datos Modelo escenario 2 56
Fig. 21 Diagrama de secuencia del caso de uso Modificar Datos Modelo escenario 3 57
Fig. 22 Diagrama de secuencia del caso de uso Visualizar Cronograma 58
Fig. 23 Diagrama de secuencia del caso de uso Visualizar Investigadores escenario 1 59
Fig. 24 Diagrama de secuencia del caso de uso Visualizar Investigadores escenario 2 60
Fig. 25 Mapa de Navegación del submódulo Gestión de la información de pacientes y de los CRD del SIMDECC 61
Fig. 26 Diagrama de clases persistentes 63
Fig. 27 Modelo Físico 64
Fig. 28 Modelo de despliegue de la aplicación 65
Fig. 29 Código de la clase ControladorPublicador 66
Fig. 30 Código del método getListadoAccionesModelo de la clase ControladorPublicadorPrincipal 67
Fig. 31 Código del Método insertarPaciente de la clase ControladorPublicador 68

Introducción

La calidad de vida es un concepto asociado a la salud cubana, y está presente en nuestra sociedad en un momento donde predomina la convicción de que el papel de la medicina no debe limitarse solo a alargar el tiempo de vida sino que, debe contribuir a mejorarlo, aportando mayor calidad al tiempo vivido. Para lograr esta premisa, en Cuba se han creado después del triunfo de la revolución varios centros biotecnológicos y científicos investigativos asociados a la producción, dentro de los cuales se encuentra el Centro de Inmunología Molecular (CIM). El mismo se dedica a realizar estudios en busca de nuevos anticuerpos monoclonales y vacunas moleculares terapéuticas capaces de mejorar la vida de pacientes con enfermedades asociadas al sistema inmune, a materializar estos productos y realizar todas las pruebas pertinentes hasta encontrar un alto nivel de seguridad para que puedan ser aplicados en humanos.

La línea base de la investigación que lleva a cabo el CIM en estos años, está basada en la inmunoterapia del cáncer, una de las enfermedades con mayor porcentaje de muerte anual a nivel mundial y de mayor dificultad para encontrar un posible tratamiento, puesto que engloba más de 100 padecimientos dispersos por el organismo humano. Los avances realizados no son pocos, ya se cuenta con una variada gama de productos aprobados a nivel mundial y más en fase de prueba, aunque una gran mayoría de ellos, son propiedad de las grandes multinacionales que piden un precio extremadamente alto para su adquisición. En el CIM después de realizar los análisis pertinentes para probar el funcionamiento de los fármacos, se aplican en pacientes que padecen diferentes enfermedades. Siempre contando con el previo consentimiento de cada uno de ellos, conociendo el hecho de que son productos en fase de prueba que pueden generar eventos adversos, aunque son mayores las posibilidades de encontrar un aliciente a su padecimiento.

Al proceso de aplicación de los productos en fase de prueba a los humanos, junto a la recuperación de todos los datos de la evolución del medicamento en ellos, es a lo que se llama Ensayo Clínico (EC). Este es un proceso en el que se deben recoger una enorme cantidad de datos, ya que como todos los organismos son diferentes genéticamente y se desarrollan bajo condiciones ambientales diferentes, reaccionan de formas distintas al efecto de los fármacos, por lo que es de vital importancia para el pleno desarrollo de este proceso, obtener la mayor cantidad de datos registrados por paciente.

Los datos recogidos en los Ensayos Clínicos deben mantenerse almacenados durante 15 años

posteriores a la realización del mismo; para poder ser consultados por las agencias reguladoras internacionales. Esto implica un enorme cúmulo de información y por ello se torna engorroso el proceso para el manejo de la misma por parte de los especialistas.

La información es recopilada en los Cuadernos de Recogida de Datos (CRD). Por cada paciente se debe llenar uno de éstos, los cuales se encuentran en formato duro. Para la realización de un EC, en dependencia de su fase, se pueden necesitar varios hospitales del país, por tanto, al concluir el ensayo, los CRD se encuentran dispersos y se necesita un gran esfuerzo por parte del CIM y una gran movilización de recursos para recopilarlos ralentizando así los procesos investigativos. Una vez recolectados todos los cuadernos se necesita la digitalización de los mismos, pues la realización de comparaciones y estadísticas de la información que se recoge sería imposible teniendo en cuenta que en cada ensayo, dependiendo de su fase, se involucran de 350 hasta 700 pacientes y se realizan más de 50 ensayos anuales.

Para digitalizar los CRD se realiza una doble entrada de datos, hacia una base de datos hecha en EPINFO o Access. Posteriormente estas dos bases de datos se comparan para encontrar la posible existencia de errores y poder corregirlos. La probabilidad de introducir errores en esta fase es bastante alta, puesto que los operadores de datos asignados para la digitalización son personas ajenas al proceso de llenado de los mismos. Puede darse el caso que ellos desconozcan mucho de los límites de los valores y las unidades de medidas registradas. Además contando con un enorme volumen de CRD a digitalizar, se llega a la conclusión de que el trabajo es bastante tedioso.

En el mundo existe una tendencia a realizar los ensayos clínicos de manera electrónica por transmisión remota de datos, ya que con esto se optimiza tiempo y se puede trabajar con una mayor cantidad de pacientes. Por las ventajas que ha reportado esta nueva tecnología, el CIM conjuntamente con la Universidad de Ciencias Informática, comenzó un proyecto llamado “Ensayos Clínicos”. Como resultado, se espera la obtención de un producto funcional que permita gestionar los procesos de ensayos clínicos de forma tal que se pueda realizar la transmisión remota de datos dentro de la Red Nacional de Hospitales hacia el Centro de Inmunología Molecular. Además, la actualización constante de las bases de datos y el procesamiento de la información clínica que de éstas se deriven.

Para ello el proyecto se dividió en los siguientes módulos:

1- Módulo de Administración
2- Módulo de Diseño
3- Módulo de Validación
4- Módulo de Monitoreo
5- Módulo Publicador

El módulo Publicador dada su complejidad fue dividido en dos submódulos:

Submódulo 1: Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos.
Submódulo 2: Traceo y reporte de datos.

La presente investigación está centrada en el submódulo Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos, la cuál está precedido por un trabajo de diploma desarrollado en la Universidad de Ciencias Informáticas que tuvo por título “Sistema de Manejo de Datos de Ensayos Clínicos: Módulo Publicador”. En dicha investigación se realizó el modelamiento del negocio, el levantamiento de requisitos, concluyendo con una propuesta del diseño y la recomendación de su implementación.

Teniendo en cuenta la situación antes expuesta emerge el siguiente **Problema Científico:**

¿Cómo obtener un producto funcional a partir de los requerimientos relacionados con la gestión de la información de pacientes y de los Cuadernos de Recogida de Datos, identificados para el módulo Publicador del Sistema de Manejo de Datos de Ensayos Clínicos?

Teniéndose como **Objeto de Estudio:** los Sistemas de Manejo de Datos de Ensayos Clínicos basados en aplicaciones Web.

Como **Campo de Acción:** los procesos de gestión de información, y estándares presentes, en Sistemas de Manejo de Datos de Ensayos Clínicos basados en aplicaciones Web.

Presentando como **Objetivo General**: desarrollar el diseño y la implementación del submódulo Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos del módulo Publicador del SIMDECC.

Del cual se plantean los siguientes **Objetivos Específicos** a desarrollar:

- Diseñar el submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador del SIMDECC.
- Implementar el submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador del SIMDECC

Para dar cumplimiento a los objetivos específicos, se llevará a cabo la resolución de las siguientes tareas.

Tareas:

- Revisión de la solución propuesta por la investigación precedente realizada al módulo Publicador del SIMDECC.
- Búsqueda de estándares presentes en sistemas automatizados vinculados al campo de acción.
- Análisis y selección de las herramientas de software a utilizar para el desarrollo del submódulo Gestión de la información de pacientes y de los CRD.
- Elaboración de los diagramas de clases del diseño.
- Descripción de las clases del diseño.
- Elaboración de los diagrama de interacción.
- Elaboración del diagrama de clases persistentes.
- Elaboración del diagrama de despliegue.
- Implementación del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador del SIMDECC.

Estructuración del contenido

Este documento está estructurado en dos capítulos. Una breve descripción de cada uno de ellos se muestra a continuación:

Capítulo 1: Fundamentación teórica, en este capítulo se abordan algunos conceptos generales y se brinda información del estado del arte de los Sistemas de Manejo de Datos de Ensayos Clínicos, basados en aplicaciones Web. Se explican las características fundamentales de las herramientas, metodologías y tecnologías escogidas para conformar la arquitectura del proyecto. Herramientas que serán asumidas para el diseño y la implementación del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador del SIMDECC.

Capítulo 2: Solución propuesta, en este capítulo se presentará una propuesta para dar solución al problema científico planteado. Para ello se mostrarán conceptos importantes, así como los patrones de diseño que serán aplicados en la realización de los correspondientes casos de uso. Se mostrarán los diagramas de interacción, despliegue y el de clases persistentes. Por último se logrará una mejor comprensión de la navegabilidad del sistema presentando el mapa de navegación. Se evidenciará la factibilidad del diseño desarrollado a través de la presentación de fragmentos de código.

Capítulo 1: Fundamentación Teórica

En este capítulo se abordan algunos conceptos generales y se brinda información del estado del arte de los Sistemas de Manejo de Datos de Ensayos Clínicos, basados en aplicaciones Web. Se explican las características fundamentales de las herramientas, metodologías y tecnologías escogidas para conformar la arquitectura del proyecto. Herramientas que serán asumidas para el diseño y la implementación del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador del SIMDECC.

1.1 ¿Qué es un Ensayo Clínico?

Un Ensayo Clínico es cualquier investigación en seres humanos que se realice con el objetivo de descubrir o verificar los efectos clínicos, farmacológicos y/o farmacodinámicos de un producto en investigación. Los EC son usados para investigar cualquier reacción adversa a un producto en investigación, y estudiar su absorción, distribución, metabolismo y expresión, con el objetivo de probar su seguridad y/o eficiencia (1)

1.2 ¿Qué es un CRD?

Un CRD es un documento impreso o electrónico diseñado para recoger toda la información que requiere un protocolo de investigación. Se llena uno por cada uno de los sujetos involucrados en un ensayo clínico, para que pueda ser reportado al promotor de la investigación. Es conocido también como libreta de trabajo o registro de resultados de datos de interés, documento donde están recogidos y archivados todos los datos que se toman y se generan, tanto en una investigación, prueba o ensayo clínico, necesarios para poder realizar un informe comprobable sobre lo realizado y que en un futuro puedan necesitarse. (2)

1.3 Módulo Publicador.

El proyecto Ensayos Clínicos está formado por cinco módulos que se relacionan entre sí, dentro de lo cuales está el Módulo Publicador. Las funcionalidades del módulo en cuestión, son agrupadas dadas sus características de similitud en dos submódulos: Traceo y reporte de datos y Gestión de la información de pacientes y de los Cuadernos de Recogida de Datos, en el cual se centra la presente

investigación. Una vez que se inicia un EC comienzan los procesos de inclusión de los pacientes que están involucrados en el estudio. Por cada paciente se llena un CRD donde se completan un conjunto de formularios para la recogida de información específica, relacionada con el estudio de la efectividad de los fármacos aplicados al paciente. Después de realizado este proceso, tanto el paciente como los modelos llenados transitarán por diferentes estados y permanecerán en una base de datos, concluido el ensayo, por más de 15 años. Estas acciones serán solo desarrolladas por el personal autorizado, regido por roles. A continuación se explica la función de cada uno de ellos:

Investigador Promotor: Persona o institución que promueve o auspicia el ensayo clínico. Es el propietario final de la data clínica generada. Este usuario no tiene acceso a modificar dato alguno, su trabajo es revisar y dar por correcto el diseño de un CRD en particular. Teniendo acceso a visualizar cualquier información sin importar el momento en que se encuentre el EC, pudiendo además generar copias totales o parciales de la información de su ensayo.

Investigador Principal: Es el responsable máximo de toda la información clínica recogida en su sitio. Tiene acceso a toda la información relativa a la investigación a la que pertenece y podrá incluso, hacer modificaciones en la misma, mientras permanezca abierto el ensayo y el modelo en el que se desea realizar los cambios, se encuentre en estado de completo.

Coordinador de la Investigación: Es la persona encargada de introducir en un sitio dado, en un ensayo específico, los pacientes que serán estudiados. Introduce además al CRD electrónico, la información clínica que se le recoja al paciente. Tiene acceso a introducir y modificar información de forma cronológica, según la ejecución de un protocolo mientras no haya dado el modelo por completo. Tendrá acceso a visualizar la información que él mismo ha introducido en el ensayo mientras esté abierto. Solo podrá generar copias impresas o electrónicas de la información de su sitio, bajo autorización del Investigador Principal del sitio y del Investigador Promotor.

El submódulo funcional que se obtendrá como resultado de la presente investigación, deberá ser capaz de realizar los procesos de gestión de la información de pacientes y de los CRD, especificados en la investigación precedente respondiendo a un total de 12 requisitos funcionales y 6 casos de uso del sistema. Para lograr la realización del submódulo en cuestión se realizará una búsqueda de sistemas automatizados existentes, dedicados a los procesos de gestión de la información clínica, de

ellos, se estudiarán los procesos de gestión de la información de pacientes y de los CRD y se buscará estándares existentes para el manejo de la data clínica que puedan ser aplicables al módulo que se desea desarrollar. Explorándose además la posibilidad de que exista un submódulo que realice las funcionalidades requeridas y que pueda ser integrado al SIMDECC

1.4 Sistemas automatizados vinculados al campo de acción.

La realización de la investigación antes mencionada arrojó como resultado, la existencia de varios sistemas Web para el manejo de datos de ensayos clínicos. Sistemas que en la presente investigación serán estudiados, para recopilar información sobre los estándares que utilizan y explorar la posibilidad de que puedan ser usados en el proyecto Ensayos Clínicos. Dentro los sistemas encontrados se tienen los siguientes: Pivotal, Hipócrates y OpenClínica.

1.4.1 Pivotal.

PIVOTAL es un sistema que brinda servicios de consulta médico-farmacéutica de investigación clínica online, especializado y organizado por diferentes terapias. Cuenta con una serie de servicios que se listan a continuación. (3)

Servicios que brinda:

- Diseño e implementación de bases de datos de ensayos clínicos.
- Diseño y programación de rutinas de validación de datos con PL/SQL y Perl-SQL.
- Diseño de CRD en papel o electrónicos.
- Doble entrada de datos con validación por tercera persona.
- 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. (3)

Sistemas que usan:

- Clinical Trial Data Management System: Oracle Clinical
- Conjunto de rutinas en PL/SQL para validación de datos clínicos

- Sistema de manejo de diccionarios y listas de codificación, usando diccionarios de cliente o estándar. (3)

Estándares para Proceso de Datos Clínicos:

- Plan de proceso de datos clínicos. Detallando y abarcando todos los aspectos de manejo y validación de datos. Incluyendo entrada y corrección de datos, importación de datos externos, codificación, resolución de discrepancias, etc.
- Bases de datos para análisis intermedios y 'revisión ciega' de alta calidad.
- Sistemas de seguimiento de CRD y de seguimiento de eventos y gestión de ensayos clínicos.
- Informes de seguimiento de calidad de datos, adaptados a las necesidades de los clientes.
- Librería de modelos de CRD con sus bases de datos correspondientes.
- Extenso catálogo de formatos de presentación de datos (3)

1.4.2 Hipócrates.

Hypocrates es un sistema informático que se usa para el control burocrático y archivo total de consultas médicas, recogiendo datos generales de enfermedades de los pacientes y la evolución de medicamentos aplicados a los mismos.

Servicios que brinda:

- Planificación de los ensayos (Vía link con el programa de realización de diagramas de planificación).
- Listado de todos los ensayos.
- Estado actual de los ensayos.
- Tiempo restante del ensayo.
- Gestión de centros del estudio.
- Monitorización.
- Envío de documentación.
- Investigadores participantes.
- Gestión de consultas (“queries”) de disconformidad entre datos entrados en la base de datos y la realidad.

- Inventario de CRD. (4)

Estándares para Proceso de Datos Clínicos:

- Bases de datos para análisis estadísticos intermedios.
- Sistemas de seguimiento de CRD y de seguimiento de eventos y gestión de ensayos clínicos.
- Informes de seguimiento de calidad de datos. (4)

1.4.3 OpenClínica.

Es una aplicación de código abierto para la recogida de datos electrónicos, usado en investigaciones clínicas y el diseño de cuadernos para la captura de datos. Los CRD pueden ser creados como se desee sin tener conocimientos de programación. OpenClínica proporciona además, una interfaz fácil de usar para la presentación y validación de los datos, facilitando su uso a los médicos e investigadores que participan en la inscripción de los pacientes. Permite la extracción de datos y filtrado de archivos para ser empleados por los investigadores, estadísticos, y los directores de estudio, así como la administración de cuentas de usuario, presentación de informes por los administradores, auditorias, y configuración. Posibilita el intercambio de recursos de una forma segura y transparente y el manejo de los datos de importación y exportación de herramientas, para la migración de datos clínicos a las hojas de cálculo y bases de datos locales. (5)

Servicios que brinda:

- Captura de datos clínicos.
- Extracción de datos
- Filtrado de archivos
- Composición modular.
- Presentación de informes
- Configuración de protocolos.
- Diseño de CRD.
- Validación de CRD.

Estándares para Proceso de Datos Clínicos:

El sistema OpenClínica se basa en un clásico patrón arquitectónico conocido como Modelo-Vista-Controlador o MVC. Haciendo a la aplicación flexible a varios niveles. (5)

Después de haber indagado en sistemas que se dedican a la gestión de datos de ensayos clínicos, en busca de un software que sea integrable al proyecto EC y que cumpla con los requerimientos a los que se necesita dar solución, se llegó a la conclusión: los sistemas encontrados no garantizan todas las funcionalidades requeridas .El sistema más completo de los encontrados en cuanto a funcionalidades y con mas posibilidades de ser usado es OpenClínica, pero presenta ciertas modificaciones a la hora de dar solución a los requerimientos planteados por los clientes y es imposible integrarlo como submódulo al SIMDECC. A continuación serán descritas razones tangibles por las cuales no es factible su uso para dar solución al problema propuesto.

Deficiencias de OpenClínica.

- El módulo publicador depende de un cronograma para el llenado de los datos, puesto que si se llena un dato fuera de fecha debe levantarse una querie, para ello se crea un cronograma enfocado al ensayo, para que los datos de un modelo sean llenados el día correspondiente a partir de la primera inmunización y en OpenClínica los cronogramas son enfocados a pacientes. Puede darse el caso que un paciente tenga más de un cronograma de ejecución, por lo que no se tendrá control del llenado de los datos y no se levantan queries.

- Para el diseño de los CRD, las variables son importadas de un Excel, donde se declaran, siendo éste un software propietario de la Microsoft, por tanto no se tiene licencia para su uso.

- Una vez cargadas las variables, OpenClínica las ubica de forma secuencial, no permitiendo la organización de las mismas, por tanto, el diseño no es flexible a las necesidades de los clientes.

- Según los requerimientos del negocio del cliente, se demanda la existencia de un Investigador Principal que supervise todo el proceso del llenado de los datos, los certifique y los firme, privilegio que no está concebido para el rol de Investigador Principal en OpenClínica.

- OpenClínica esta desarrollado en Java y el proyecto Ensayos Clínicos, tiene una arquitectura definida por la cual se rigen todos sus módulos para el desarrollo. Tomar funcionalidades de OpenClínica e integrárselas al proyecto significaría trabajar con dos arquitecturas.

Habiéndose expuesto las razones por las cuales no es factible el uso de OpenClínica, se ratifica la necesidad de construir un software que satisfaga las necesidades del CIM. Un software que se acople e implemente los estándares internacionales más relevantes de los encontrados en la investigación realizada, para así garantizar la eficiencia y calidad del producto que se obtendrá .Un producto que sea creado en Cuba, cuyo uso podrá ser extendido posteriormente a todas las instituciones que realizan ensayos clínicos en dicho país. Para ello, se da paso a la descripción de las herramientas, metodologías y tecnologías que serán usadas en el desarrollo del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador del SIMDECC. Las herramientas, metodologías y tecnologías que se presentarán a continuación fueron seleccionadas y aprobadas por el grupo de arquitectura del proyecto Ensayos Clínicos con la participación de todos los miembros del proyecto, por lo que solamente se explicarán las características más relevantes de las mismas. Las respectivas comparaciones para determinar su uso se encuentran fundamentadas en el documento de la arquitectura del proyecto y algunas en la investigación precedente.

1.5 Metodología de Desarrollo de Software y lenguaje de modelado.

1.5.1 Metodología de desarrollo.

Toda persona que se haya enfrentado ante la tarea de construir un software, ha encontrado la polémica de cómo realizarlo, que método utilizar para guiar el desarrollo y controlar los cambios. Para desarrollar un software se necesita el uso de una metodología de desarrollo, que permita guiar el proceso de forma correcta, para que los desarrolladores y clientes estén conformes con el resultado obtenido. Para ello es necesario el uso de una metodología de desarrollo flexible, que permita de forma sencilla poder manejar cambios pedidos por el cliente, después de haber concluido con la construcción del software. Existen varias metodologías de desarrollo de software, tales como: RUP, MSF y XP. Estas son utilizadas en dependencia de las características y dimensiones del proyecto solicitado. Después de hacer una revisión bibliográfica de la investigación precedente y analizar las metodologías propuestas con sus respectivas comparaciones, se llegó a la conclusión de que RUP constituye una de las metodologías más utilizada para el análisis, implementación y documentación de sistemas orientados a

objetos, es un metodologías adaptable al contexto y necesidades de cada organización, por lo que se coincide en el uso de RUP para el desarrollo del submódulo Gestión de la información de pacientes y de los CRD del módulo publicador. Contando además con que los artefactos generados en la investigación precedente fueron desarrollados usando RUP.

- **RUP:**

Proceso Unificado de Desarrollo de Software que consta de 4 fases. En cada fase se da cumplimiento a un objetivo en específico, teniendo al finalizar cada una de ellas, un resultado tangible y comprobable. Por ejemplo, en la fase de inicio se pretende determinar la visión del proyecto, en la fase de elaboración, determinar la arquitectura óptima, en la fase de construcción, obtener la capacidad operacional inicial y por último en la fase de transición, el objetivo es llegar a obtener el release del proyecto. RUP se desarrolla de forma iterativa incremental, por tanto, cada iteración está en función de la precedente. El ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas; la disciplina de desarrollo y la de soporte. En cada una de las faces de RUP se asumen roles y se generan artefactos. La presente investigación, esta centrada en los flujos de trabajo de Análisis y Diseño e Implementación. A continuación se listarán los artefactos construidos por los roles en cada flujo de trabajo. (6)

Flujo de trabajo Análisis y Diseño:

Trabajadores:

- Arquitecto
- Diseñador
- Diseñador de BD
- Diseñador de interfaz de usuario

Artefactos:

- Paquetes del diseño.
- Realización de CU-diseño: Diagramas de clases del diseño y diagramas de interacción.

- Diagrama de despliegue.
- Prototipos de interfaz de usuario.
- Mapa de navegación.
- Modelo de datos.

Flujo de trabajo Implementación:

Trabajadores:

- Programador

Artefactos:

- Prototipo ejecutable

1.5.2 Lenguaje de modelado.

Después de haber escogido una metodología de desarrollo, es necesario encontrar el lenguaje de modelado correcto. Atendiendo a la necesidad existente de definir un sistema de software, documentarlo, detallar los artefactos generados y construirlos, habiendo constatado el lenguaje propuesto en la investigación precedente, se llegó a la conclusión de que será usado UML. Este se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), que es la metodología de desarrollo que se utilizará en el proyecto EC.

- **UML 2.0:**

Lenguaje Unificado para la Construcción de Modelos, se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema notacional que entre otras cosas incluye el significado de sus notaciones, está destinado a los sistemas de modelado que usan conceptos orientados a objetos. (7)
Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), que será usado para el desarrollo del submódulo Gestión de la información de pacientes y de los CRD. UML para el modelado UML cuenta con varios diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de estados.
- Diagrama de secuencias.
- Diagrama de actividades.
- Diagrama de colaboraciones.
- Diagrama de componentes.
- Diagrama de distribución.

UML no es un método de desarrollo, por tanto, no dice cómo pasar del análisis al diseño y de este al código. No son una serie de pasos que te llevan a producir código a partir de unas especificaciones. UML es independiente del ciclo de desarrollo que se vaya a seguir, puede encajar en un tradicional ciclo en cascada, en un evolutivo ciclo en espiral o incluso, en los métodos ágiles de desarrollo. (7)

Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante resaltar, que UML es un "lenguaje" para especificar procesos. (8)

1.6 Tecnologías.

1.6.1 Servidor Web.

El producto que se obtendrá de la presente investigación será una aplicación Web con tecnología Cliente-Servidor por tanto se necesitará un servidor web que sea compatible con el sistema operativo GNU/Linux, que será el Sistema Operativo instalado en el servidor, un servidor que sea potente y soporte una gran cantidad de conexiones, de los encontrados, se llegó a la conclusión de que el mejor candidato es Apache.

- **Apache 2.2.3:**

Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita, es importante, pero no tanto como que se trate de código fuente abierto. Esto le da una alta transparencia a este software. Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar

sus capacidades. Apache trabaja con gran cantidad lenguajes de script. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache, para que ejecute un determinado script cuando ocurra un error en concreto. Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (9)

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Apache es un Servidor Web y de aplicaciones que se encuentra entre los más extendidos, por versátil y por potente, se debe tener en cuenta que se trata de un programa libre, y que su código fuente está disponible para la comunidad que lo va mejorando constantemente. Funciona sobre infinidad de sistemas y arquitecturas y suele hacerlo ofreciendo capacidades de rendimiento sorprendentes. Ventajas que ofrece:

- Modular
- OpenSource
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)
- Gratuito

1.6.2 Gestor de Base de Datos.

El sistema que se desarrollará deberá almacenar por más de 15 años los datos recogidos a los pacientes que se involucraron en el EC. Para esto se necesita contar con una base de datos, y para manejarla es necesario usar un robusto sistema gestor de bases de datos, que tenga la capacidad para almacenar un gran cúmulo de información y admita infinitas tablas.

Los gestores de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta; para la realización del submódulo Gestión de la información de pacientes y de los CRD será usado PostgreSQL. (10)

- **PostgreSQL 8.2 :**

Es uno de los servidores más utilizados en los sistemas de manejo de datos clínicos a nivel internacional por la alta seguridad y confiabilidad que posee, además de permitir hacer consultas en el lenguaje PLSQL que es uno de los estándares utilizados por los SIMDEC y por ser el gestor de base de datos libre más avanzado de los existentes. Soporta consultas complejas, incluyendo subselects, integridad referencial (Foreign Keys), triggers, vistas, integridad transaccional y Control de versionado concurrente. A continuación se listan otras de sus características:

- 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.

- 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 la reglas y directrices de integridad de la base de datos.

- 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.

- Durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

- Es altamente adaptable a las necesidades del cliente. Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc.

- Soporte de todas las características de una base de datos profesional (triggers, store procedures, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.)

- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.

- Utilidades para análisis y optimización de queries. (11)

1.6.3 Lenguaje de programación.

Para poder desarrollar el submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador y lograr la materialización del los Requisitos Funcionales intimados por el cliente, se necesita encontrar el lenguaje de programación correcto, que permita una comunicación entre el cliente, el servidor y la base de datos. Para ello se realizó una búsqueda de algunos lenguajes de programación existentes, viendo las ventajas de su uso y se concordó con el lenguaje propuesto por la investigación precedente. PHP (acrónimo de Hypertext Preprocessor) por ser un lenguaje 'Open Source' interpretado de alto nivel. Este fue creado para desarrollar aplicaciones Web, que es muy popular debido a que puede ser inmerso en páginas HTML y tiene una sintaxis fácil de aprender. (12)

- **PHP 5:**

Específicamente se usará la versión 5 complementando el uso del framework Symfony que será usado, posteriormente se procederá a su fundamentación. PHP5 permite una orientación a objetos bastante parecida al Java, presenta mejoras significativas y un ambiente de programación orientada a objetos mucho más completo. PHP5 proporciona un alto rendimiento a las aplicaciones Web. Aumentando el rendimiento y mejorando la utilización de la memoria. Presenta más seguridad a través de los filtros, que permiten a los programadores leer los campos de formularios de forma segura, tiene extensión ZIP para crear y editar ficheros zip y posibilidad de controlar el progreso de subida de ficheros (13). Además de tener en cuenta que es el lenguage estandarizado por los sistemas informáticos de la salud cubana.

1.7 Herramientas a utilizar.

1.7.1 Herramienta CASE para modelación del sistema.

Para desarrollar un software informático, se necesita una herramienta que permita su modelado contribuyendo a mejorar la calidad y la productividad, aplicando metodologías que permitan agilizar el trabajo y simplificar el mantenimiento de los programas, para ello surgieron las herramientas CASE. Para el desarrollo del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador se necesitará una herramienta que soporte el lenguaje de modelado UML y la metodología RUP. En la investigación precedente fue propuesta como herramienta CASE el Rational Rose por ser una de las más usadas en el mercado internacional y por poseer una plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a

entender el entorno de los sistemas, además de que no exige grandes requerimientos de hardware, pero es un software propietario por el cual hay que pagar una licencia por puesto de trabajo y no es multiplataforma, por lo que no podrá ser usado en la presente investigación. Tras hacerse un estudio de otras herramientas CASE multiplataforma existentes, se consideró que para el desarrollo del submódulo en cuestión es más conveniente el uso del Visual Paradigm.

- **Visual Paradigm 3.1:**

Es una herramienta CASE profesional para el modelado de UML que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales, demostraciones interactivas de UML y proyectos. (14)

1.7.2 Framework.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Los Frameworks son diseñados con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software, que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. (15)

Por todas las ventajas que presupone el uso de un Framework se decidió el uso de uno para el desarrollo del proyecto EC. La decisión de que framework utilizar se tomó a nivel de la arquitectura proyecto, con la participación de todos los miembros del mismo. Posteriormente se hizo extensible a todos los módulos.

El framework con el que se acordó desarrollar el proyecto EC fue Symfony 1.0.10, entre otras cosas porque propone el patrón arquitectónico Modelo Vista Controlador que es uno de los estándares utilizados en el desarrollo de software para el manejo de datos clínicos. En el documento de la arquitectura del proyecto están reflejadas las comparaciones que fundamentan la decisión del uso de Symfony, como framework de desarrollo, a continuación se exponen sus características fundamentales a modo de ratificar la decisión tomada.

- **Symfony 1.0.10:**

Es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Separa la lógica de negocio de la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony está desarrollado completamente con PHP5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de los gestores de bases de datos existentes, entre los que se encuentra PostgreSQL que es el que será utilizado para el desarrollo de la presente investigación. Es un Framework libre muy utilizado y con grandes funcionalidades puesto que su creador, Fabien Potencier, toma prestadas las mejores ideas de otros muchos framework y las adapta para Symfony, lo que permite la compatibilidad entre aplicaciones. (Ver Fig.1).

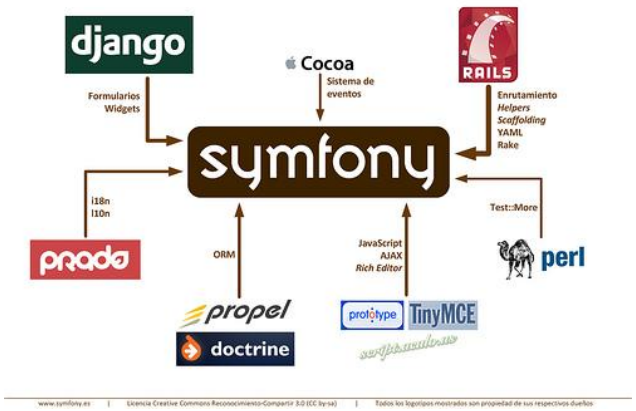


Fig. 1 Origen de Symfony

La lista de ideas que incorpora Symfony y que tienen su origen en otros frameworks y aplicaciones es la siguiente:

- Ruby on Rails: sistema de enrutamiento, helpers y archivos de configuración en formato yml
- Django: el nuevo mecanismo de formularios
- Propel y Doctrine: son los dos ORM principales de Symfony, sobre todo Propel, que se encuentra completamente integrado.
- Prado: Las funcionalidades relacionadas con la internacionalización y la localización

- Cocoa: el nuevo sistema de eventos de Symfony se basa completamente en este framework de Apple.
- Test More: la herramienta Lime, está basada en el framework Test More de Perl.
- Prototype, script.aculo.us, TinyMCE: utilidades relacionadas con Ajax y Java Script. (16)

1.7.3 Entorno de Desarrollo Integrado.

Un Entorno de Desarrollo Integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, que mediante plugins se le puede añadir soporte de lenguajes adicionales. En el desarrollo del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador se usará como IDE eclipse con el plugin PDT (PHP Development Tool) que soporta el lenguaje PHP5.para complementar el uso del framework de desarrollo escogido. Eclipse como IDE fue la herramienta propuesta para formar parte de la arquitectura del proyecto Ensayos Clínicos. En el documento de la arquitectura del proyecto están reflejadas las razones que justifican haberla escogido y a continuación se mencionan sus principales características. (17)

- **Eclipse 3.2:**

Es uno de los más poderosos editores, para lenguajes como JAVA, C, PHP, y otros, y lo mejor de todo es que es OpenSource. Consta de una comunidad de millones de usuarios y miles de desarrolladores, cientos de plugins, soporte multilenguaje en una única herramienta, cuenta con coloreado de sintaxis PHP, autocompletado de código e inspección de métodos y atributos. Además de soporte básico de depuración de scripts PHP Puede mostrar la herencia de clases, inserta bloques PHPDoc útiles para comentarios, autocompleta llaves, paréntesis, auto tabula el código de acuerdo a reglas predefinidas o propias. Muestra los errores de PHP incluyendo un tooltip mostrando cual es el error. Auto completa etiquetas, atributos HTML, incluso auto completa comentarios de PHPDoc. Puedes tener múltiples proyectos a la vez. (18)

1.7.4 Editores de interfaz Web.

Un editor de interfaz es un software que facilita las funcionalidades para crear un entorno amigable para el software que se desea desarrollar En la investigación precedente se había propuesto Dreamweaver como editor de interfaz Web, pero dado a que es un software propietario se realizó una investigación en busca de una herramienta de software libre similar al Dreamweaver. El más completo de los encontrados fue KompoZer.

- **KompoZer:**

Es un editor HTML OpenSource de licencia GNU/GPL 2.0. La interfaz no es la misma, pero los comandos resultan parecidos a Dreamweaver. KompoZer es un sistema que combina un fácil uso de la Web integrando funcionalidades encontradas en otro software. Kompozer está diseñado para ser extremadamente fácil de usar, lo que lo hace aprovechable tanto para personas con poca experiencia, como para diseñadores Web profesionales, incluye funcionalidades como: manejador de FTP, una nueva paleta de colores editores de CSS, entre otras (19).

1.7.5 Controlador de Versiones.

En el desarrollo de un proyecto en el que se cuenta con un equipo de trabajo y varios módulos que necesitan ser integrados es muy factible el uso de un controlador de versiones que permita a varios usuarios trabajar sobre los mismos archivos sin que se generen perdidas de información o se corra el riesgo de perder el código implementado. Para ello en el desarrollo del proyecto EC se propuso el uso de un controlador de versiones.

- **Subversion 3.2 :**

Es un sistema de control de versiones libre y de código fuente abierto. Maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. En este aspecto. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas

ubicaciones fomenta la colaboración. Como el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada por la pérdida de información si se ha hecho un cambio incorrecto a los datos, simplemente se deshace el cambio. (20)

1.8 Conclusiones.

En el presente capítulo se estudiaron las herramientas, tecnologías y metodologías propuestas en la investigación precedente sin dejar de considerar otras existentes que pudieran ser más factibles a utilizar para el desarrollo de la presente investigación. Fueron abordados satisfactoriamente conceptos necesarios para la comprensión del funcionamiento del proyecto. Se explicó el funcionamiento del submódulo Gestión de la información de pacientes y de los CRD del SIMDECC, así como las responsabilidades del personal por roles de autenticación al módulo. En el presente capítulo se estudiaron softwares que brindan funcionalidades similares a las necesitadas en el submódulo en cuestión, investigando los estándares que acatan, asumiéndose algunos de ellos para el desarrollo del submódulo Gestión de la información de pacientes y de los CRD.

Estándares que serán asumidos:

- Acceso por niveles de seguridad, con diferencias de permisos en la visualización y modificación de la información.
- Notificación de inconsistencias.
- Colección de datos en una bases de datos para la realización de análisis estadísticos.
- Seguimiento de eventos ocurridos a los pacientes involucrados en un EC y gestión de datos.
- Uso de patrones arquitectónicos como el Modelo Vista Controlador para el desarrollo de la aplicación.
- Uso de un potente lenguaje de consultas.

Estos estándares son asumidos para mantener la confidencialidad e integridad de los datos obtenidos, propiedad fundamental que deben poseer los mismos para que el estudio sea factible. Permiten un rápido desarrollo del software y una alta flexibilidad en el mismo.

Capítulo 2: Solución propuesta

En este capítulo se presentará una propuesta para dar solución al problema científico planteado. Para ello se mostrarán conceptos importantes, así como los patrones de diseño que serán aplicados en la realización de los correspondientes casos de uso del sistema. Se mostrarán los diagramas de interacción, despliegue y el de clases persistentes. Por último se logrará una mejor comprensión de la navegabilidad del sistema presentando el mapa de navegación. Se evidenciará la factibilidad del diseño desarrollado a través de la presentación de fragmentos de código.

2.1 ¿Qué es el diseño?

Para crear una aplicación es necesario contar con una descripción detallada y de alto nivel de la solución lógica y saber como satisfacer los requerimientos y las restricciones. El Diseño pone de relieve una solución lógica: como el sistema cumple con los requerimientos.

Durante el diseño orientado a objetos, se procura definir los objetos lógicos del software que finalmente serán implementados en un lenguaje de programación orientado a objetos. El diseño se ocupa de desarrollar las directrices propuestas durante el análisis en función de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados, tanto desde el punto de vista funcional como del no funcional. (21)

2.1.1 Propósitos y objetivos del diseño.

Cuando la disciplina de diseño se asume como es debido, los objetivos quedan automáticamente determinados y son una poderosa herramienta para lograr cumplir los requerimientos del negocio. Puesto que el diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería, donde el objetivo del diseñador es producir un modelo o representación de una entidad que se construirá posteriormente. Con el diseño del submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador, se pretende identificar y describir los objetos dentro del dominio del problema planteado, o sea se pretende definir los objetos que posteriormente serán implementados. El diseño del submódulo será obtenido a partir de los artefactos generados en la investigación precedente, realizada al módulo en cuestión, centrándose solamente en los requisitos funcionales implicados en los procesos de gestión de la información de pacientes y de los CRD, dichos RF son listados a continuación:

R1- Insertar nuevo paciente.
R2- Listar hospitales involucrados en un ensayo.
R3- Listar sitios pertenecientes a un hospital.
R4- Listar especialistas pertenecientes a un sitio.
R5- Listar pacientes.
R6- Listar visitas de un paciente.
R7- Visualizar modelos.
R8- Insertar datos de un modelo para un paciente.
R9- Modificar modelos.
R10- Cambiar el estado de un modelo.
R11- Cambiar estado de un paciente.
R12 - Mostrar cronograma de ejecución del ensayo para un paciente específico.

Estos RF responden a un total de 6 casos de uso del sistema de los cuales se mostrará en la presente investigación los correspondientes diagramas de clases Web del diseño, por lo que, después de mencionarlos, se proseguirá a explicar los estereotipos que serán utilizados para modelarlos. Los casos de uso son los siguientes:

Insertar_Paciente_Ensayo.
Insertar_Datos_Modelos.
Visualizar_Cronograma.
Visualizar_Modelo.
Visualizar_Investigadores.
Modificar_Datos_Modelos.

2.1.2 Estereotipos del diseño Web.

En la presente investigación se utilizará UML como lenguaje de modelado para diseñar el submódulo Gestión de la información de pacientes y de los CRD del módulo Publicador, para la futura obtención de una aplicación Web, UML presenta una extensión de estereotipos para realizar el modelado de aplicaciones Web, dado que dicha extensión será usada en los diagramas de clases del diseño correspondientes al presente capítulo, se cree pertinente explicarlas.

<<Server Page>>: Estereotipo que representa las páginas servidoras, que son aquellas que poseen código que interactúa con recursos en el servidor, o que es ejecutado en el servidor.

Fig. 2 Estereotipo de la página servidora

<<Client page>>: Estereotipo que representa las páginas clientes que son interpretadas por el navegador, cuando las construye una página servidora, estas páginas están conformadas por código HTML y es una mezcla entre la capa lógica, la de presentación y los datos.

Fig. 3 Estereotipo de la página cliente

<<Form>>: Estereotipo que representa los formularios con los cuales la página cliente tiene una relación de inclusión, debajo de él deben especificarse todos los atributos de entrada del formulario, como los Text Area por ejemplo, desde este estereotipo se representa el envío de información hacia la página servidora a través del <<Submit>>.

Fig. 4 Estereotipo del formulario

2.2 Patrones.

Un patrón es un modelo que encapsula un problema que se repite una y otra vez en un ambiente determinado, proponiendo una solución para el mismo, solución que debe dar respuesta al problema aunque este tenga variaciones. Los patrones proporcionan catálogos de elementos reusables en el diseño de sistemas de software. Evitan la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Formalizan un vocabulario común entre diseñadores. Estandarizan el modo en que se realiza el diseño. Facilitan el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente. Generalmente son soluciones concretas y técnicas. Para el desarrollo de todo software deben valorarse el uso de estándares, en el caso del diseño de un software existen varios patrones a seguir. (22)

2.2.1 Patrones aplicados en el diseño:

Los Patrones de Diseño, son la base para la búsqueda de soluciones a problemas comunes en el

desarrollo de software y otros ámbitos referentes al diseño de interacciones o interfases. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. Son aquellos que muestran esquemas para definir estructuras de diseño con las que se puede construir sistemas de software. Dentro de los patrones aplicados al diseño se encuentran: (22)

GRASP:

Patrones de Asignación de Responsabilidades. Una de las cosas más complicadas en Orientación a Objeto, consiste en elegir las clases adecuadas y decidir como estas clases deben interactuar, es importante elegir cuidadosamente las responsabilidades de cada clase en la primera codificación y, fundamentalmente, en la refactorización del programa. A continuación se da una breve explicación de los patrones GRASP (22)

- **Bajo Acoplamiento:** Debe haber pocas dependencias entre las clases para poder extraerlas de un modo independiente y reutilizarlas.
- **Alta Cohesión:** Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable manteniendo la complejidad dentro de los límites manejables.
- **Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.
- **Creador:** Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando.

  B contiene a A.

  B es una agregación (o composición) de A.

  B almacena a A.

  B tiene los datos de inicialización de A (datos que requiere su constructor)
  B usa a A.
- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a

clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

- **Polimorfismo:** Identificar variaciones en un comportamiento, asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos.
- **Fabricación Pura:** Cuando los problemas se complican, construir clases que se encarguen de construir los objetos adecuados en cada momento (factorías).
- **Indirección:** Crear clases intermedias para desacoplar clientes y servicios.
- **No hables con extraños:** Un objeto, solamente invocará a :
    - Métodos de sí mismo (this)
    - Métodos de su área de parámetros
    - Un objeto creado en su propio ámbito (22)

- **Patrón Singleton:**

Es un patrón diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. El patrón Singleton se implementa creando una clase mediante 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 el alcance del constructor (con atributos como protegido o privado). La instrumentación del patrón puede ser delicada en programas con múltiples hilos de ejecución. Si dos hilos de ejecución intentan crear la instancia al mismo tiempo y esta no existe todavía, sólo uno de ellos debe lograr crear el objeto. La solución clásica para este problema es utilizar exclusión mutua en el método de creación de la clase que implementa el patrón. Las situaciones más habituales de aplicación de este patrón son aquellas en las que dicha clase, controla el acceso a un recurso físico único (como puede ser el ratón o un archivo abierto en modo exclusivo), o cuando ciertos tipos de datos deben estar disponibles para todos los demás objetos de la aplicación. (23)

- **Patrón Decorator:**

Responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto permite no tener que crear sucesivas clases, que hereden de la primera, incorporando la nueva funcionalidad, sino otras que la implementan y se asocien, añadiendo objetos individuales de forma dinámica y transparente. Este patrón es aplicable cuando:

- Las responsabilidades de un objeto pueden ser retiradas.
- La extensión mediante la herencia no es viable.
- Existe la necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderlo a través de la herencia, o hay la necesidad de extender dinámicamente la funcionalidad de un objeto y quizás quitar la funcionalidad extendida (24).

- **Patrón Fachada:**

El patrón fachada trata de simplificar la interfaz entre dos sistemas o componentes de software, ocultando un sistema complejo detrás de una clase que hace de pantalla o fachada. La idea principal es ocultar lo más posible la complejidad de un sistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca un punto de entrada al sistema tapado por la fachada. Una ventaja más de usar una clase fachada para comunicar las dos partes o componentes, es la de aislar los posibles cambios que se puedan producir en alguna de las partes. Si se cambia, por poner un ejemplo, el medio de comunicación o de almacenamiento de una de las partes, la otra, que por ejemplo hace la presentación, no tiene porque enterarse, y viceversa (25)

Los patrones antes mencionados son los encargados de regir el comportamiento de los objetos que serán creados, basados en el comportamiento de las clases definidas. Podrán ser evidenciados en muchos casos, por ejemplo: en el sistema se cuenta con tres roles de acceso a datos. Cada uno de ellos con permisos de realizar acciones diferentes, por lo que, se creará una clase controladora general, que establezca el comportamiento de los objetos que se crearán al autenticarse un usuario dado un listado de acciones a realizar, en dependencia del usuario que haya sido autenticado. Clase de la que heredarán otros tres controladores, pero en este caso serán específicos para cada rol que contiene el listado de las acciones a realizar por ellos sobre los modelos.

Hasta el momento se ven evidenciados los patrones: Experto, Controlador y Polimorfismo. Para la construcción de un objeto de tipo controlador específico se creará una clase fachada de la cual se puede crear una sola instancia y será la encargada de devolver un objeto de tipo controlador

específico. La clase fachada mencionada será el único punto de entrada de los demás módulos del proyecto EC, al submódulo Gestión de la información de pacientes y de los CRD, obteniéndose a través de ella el objeto necesario para invocar los métodos que se deseen ejecutar , puesto que los métodos solo pueden ser invocados en el ámbito al que pertenecen. Esto evidencia el uso de los patrones Fachada, Singleton, Creador, y No hables con extraños.

2.2.2 Patrones arquitectónicos:

Son aquellos que expresan un esquema organizativo estructural, fundamental para sistemas software, describen un problema particular y recurrente, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (26)

A continuación se explicará el patrón Modelo Vista Controlador que aún siendo un patrón arquitectónico es necesario explicarlo en esta sección, puesto que Symfony se acoge a este patrón y su uso implica reorganizar cada uno de los módulos del proyecto SIMDECC. Debido al uso de Symfony el submódulo Gestión de la información de pacientes y de los CRD debe ser nuevamente diseñado, para lograr una adaptación a la arquitectura propuesta por el framework de desarrollo.

MVC:

El modelo MVC es uno de los patrones que más se utiliza en la construcción de aplicaciones Web. Este permite construir en tres capas de aplicación, permitiendo la separación del código entre cada una de las capas, y ayudando tanto a desarrolladores como a diseñadores a cooperar y mantener el código fuente más fácilmente, además permite que el software sea fácil de modificar.

El Modelo (Model):

Esta capa se encarga de acceder a los datos, manejándolos y haciendo las respectivas transformaciones sobres estos. Esta capa no tiene conocimiento de la vista, no tiene conocimiento tampoco del controlador, ni referencia de estos.

La Vista (View):

Esta capa se encarga de manejar la presentación visual de los datos al usuario representados por el modelo.

El Controlador (Controller):

En esta capa se tiene lo que es llamado la lógica del negocio, se encarga de responder a los eventos o acciones que el usuario ejecuta en la Vista, la cual envía las variables al controlador para que ejecute la opción adecuada. (27)

El patrón arquitectónico MVC y el de diseño Decorator, serán utilizados puesto que están presentes en la arquitectura propuesta por el framework de desarrollo Symfony y se evidencian en los diagramas de clases del diseño, existiendo una separación en capas, la vista, el modelo y el controlador, el patrón Decorator se evidencia en la vista, en la interacción existente entre el layout y las plantillas.

2.3 Paquetes del diseño.

Paquetes de Diseño: Es una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén de alguna forma relacionados. Es usado para estructurar el modelo de diseño dividiéndolo en partes más pequeñas. Forma parte la vista lógica de la aplicación. Estos pueden ser agrupados atendiendo a la relación entre casos de uso, acciones realizadas por determinado actor y otros criterios. En la presente investigación se cuenta con tres paquetes del diseño, que fueron agrupados por funcionalidad a continuación se presenta su correspondiente diagrama. (28)

En el diagrama de paquetes del diseño que se presenta a continuación se agrupa la realización de 6 casos de uso, 4 de ellos dentro del paquete Gestionar Modelo Paciente, que fueron agrupados según su funcionalidad, puesto que, todos realizan acciones sobre los modelos. Incluyéndose el caso de uso Insertar_Paciente_Modelo pues la inserción de un paciente en el sistema está dada por el llenado de un modelo de inclusión y el cumplimiento o no, de los criterios que este describe. En el paquete Visualizar Cronograma se agrupa la realización de un caso de uso que lleva el mismo nombre del paquete y está relacionado con el anterior puesto que, cuando se inserta un paciente en el sistema ocurrirá una interacción con la fachada del módulo Cronograma del SIMDECC y se le generará un cronograma específico al paciente insertado. El paquete Visualizar Investigadores contiene la realización del caso de uso que lleva el mismo nombre y tiene una relación con el paquete Gestionar Modelo Paciente puesto que desde el paquete Visualizar Investigadores se puede ir al paquete Gestionar Modelo Paciente.

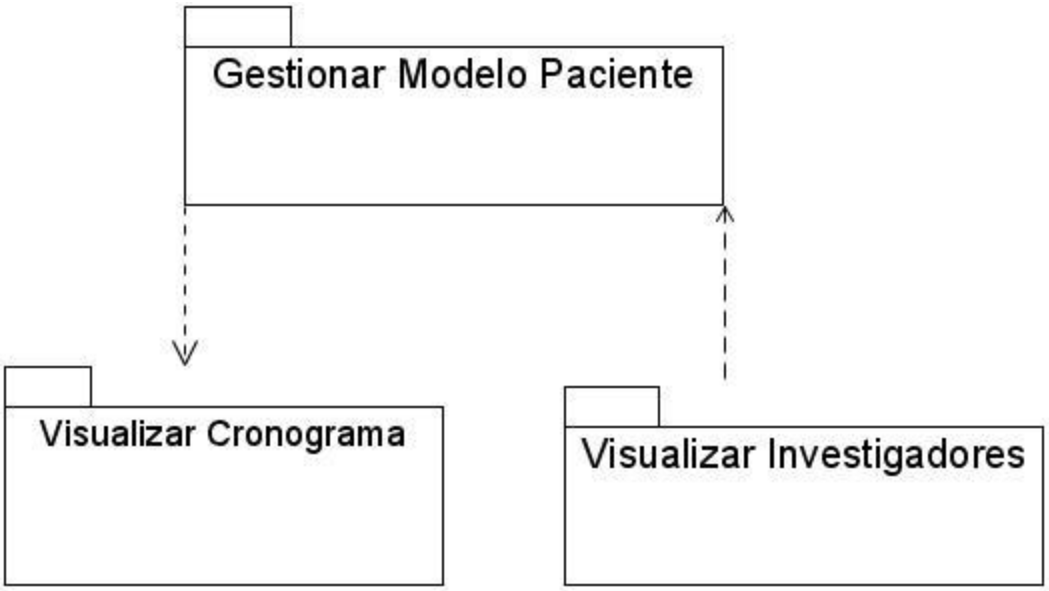


Fig. 5 Paquetes del diseño

2.4 Clases del diseño.

Clases del diseño: son una abstracción de una clase o construcción similar en la implementación del sistema. Para diseñar las clases del diseño se utilizan los estereotipos mencionados anteriormente, sumándole solamente las clases propias de la solución orientada a objetos, relacionada con el lenguaje de programación propuesto, las cuales se ejecutarán en el servidor, es decir, los únicos elementos de los descritos hasta el momento que pueden tener relación con las clases recién incorporadas, serían las páginas servidoras (Server Page). Lo más propio, por tanto, es encontrar que desde el código servidor, se creen objetos, instancias de las clases del negocio o de acceso a datos, en las cuales se delegan las responsabilidades. Quedando en las páginas servidoras el procesamiento de los pedidos de las páginas clientes, que es completado, delegando en las clases de negocio o de acceso a datos. (29)

A continuación se muestran los diagramas de clases del diseño correspondientes a los casos de uso en desarrollo. Es válido explicar una situación particular existente en los diagramas de clases del diseño de los casos de uso. Insertar_Datos_Modelo_Paciente, Visualizar_Modelo y Modificar_datos_Modelos. El diagrama de clases del diseño de estos tres casos de uso es el mismo para los tres, puesto que, interactúan con las mismas clases. El framework Symfony propone el uso de clases llamadas Action que son las encargadas de controlar lo que será mostrado en la vista a través de un método execute() puede hacerse un Action general para cada módulo del proyecto, o uno independiente para cada página que se le desea mostrar al usuario, para el desarrollo del submódulo Gestión de la información de pacientes y de los CRD se optó por la segunda opción, por lo que se tendrá una clase Action llamada mostrarModeloAction que será la encargada de mostrar los modelos ya sean vacíos o con las variables que hayan sido recogidas anteriormente, permitir la inserción de datos en ellos y la modificación de los mismos. Con la clase anteriormente mencionada interactuarán los tres roles que pueden autenticarse al módulo publicador, mostrándoseles solamente en la vista las acciones que estos pueden desempeñar sobre la página, por tanto sería la misma clase dando respuesta a tres casos de usos diferentes a través de un mismo método.

2.4.1 Diagramas clases del diseño.

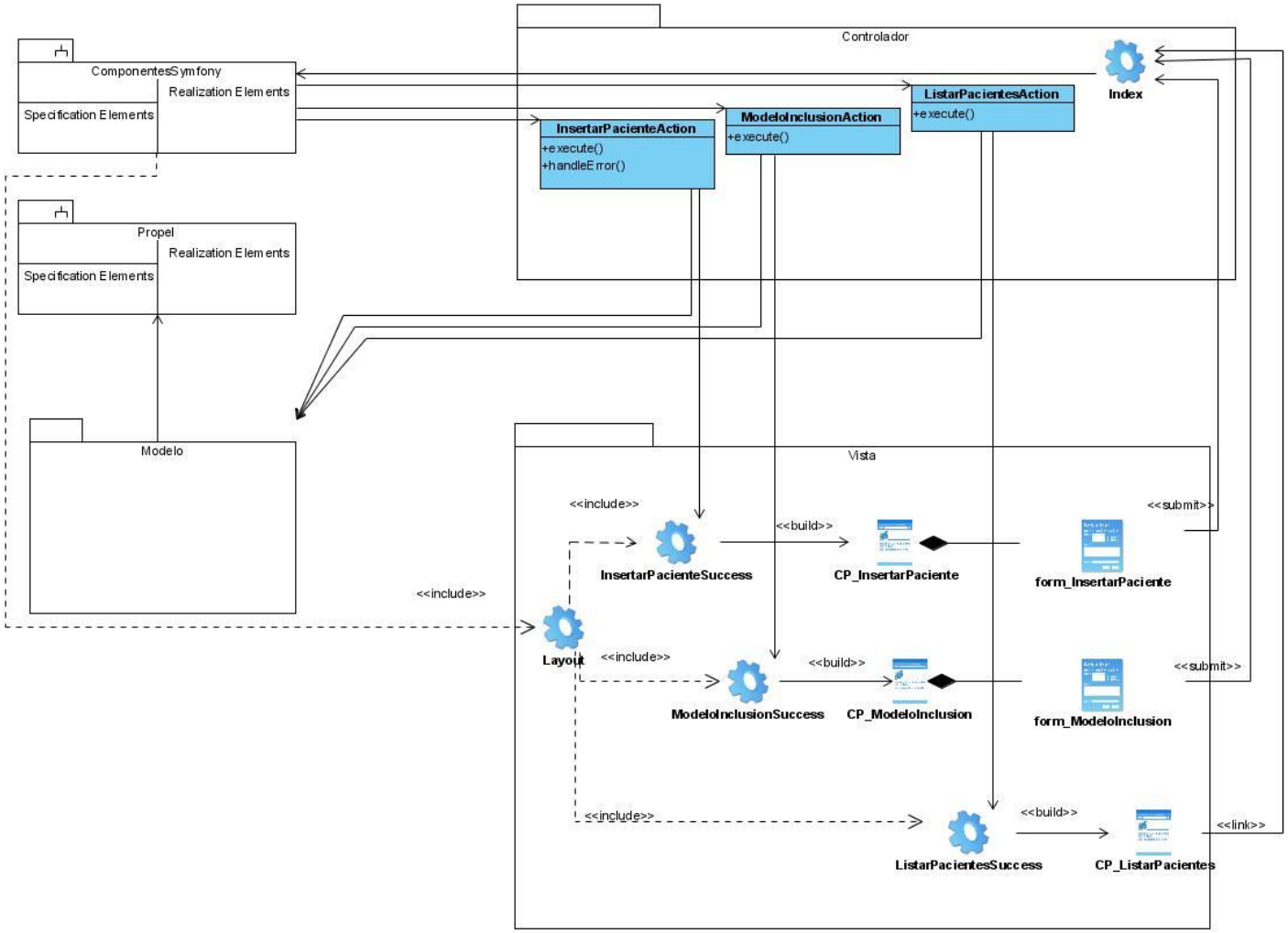


Fig. 6 Diagrama de clases del diseño del caso de uso Insertar Paciente Ensayo

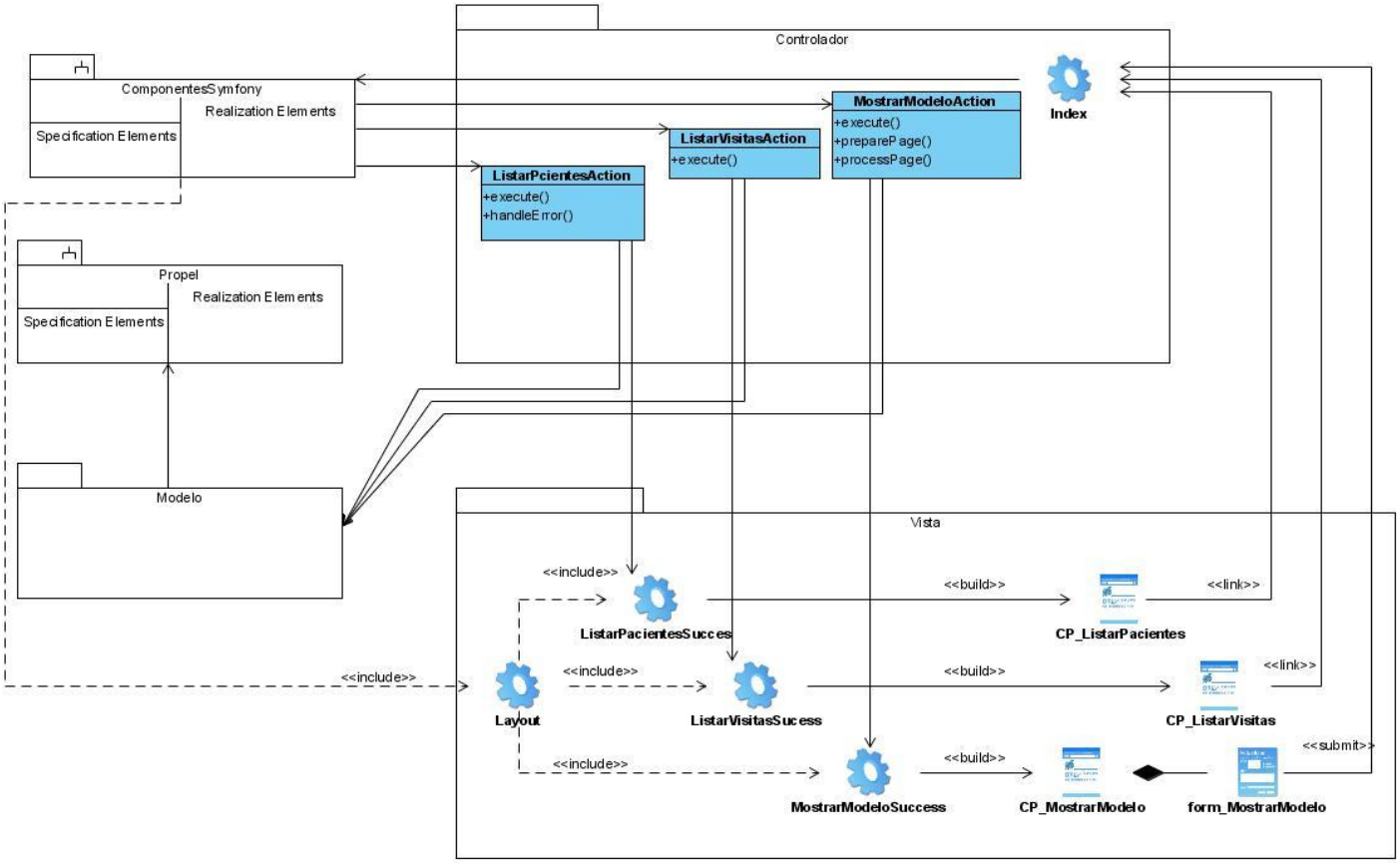


Fig. 7 Diagrama de clases del diseño de los casos de uso Insertar Datos Modelos Paciente, Visualizar Modelo y Modificar Datos Modelos

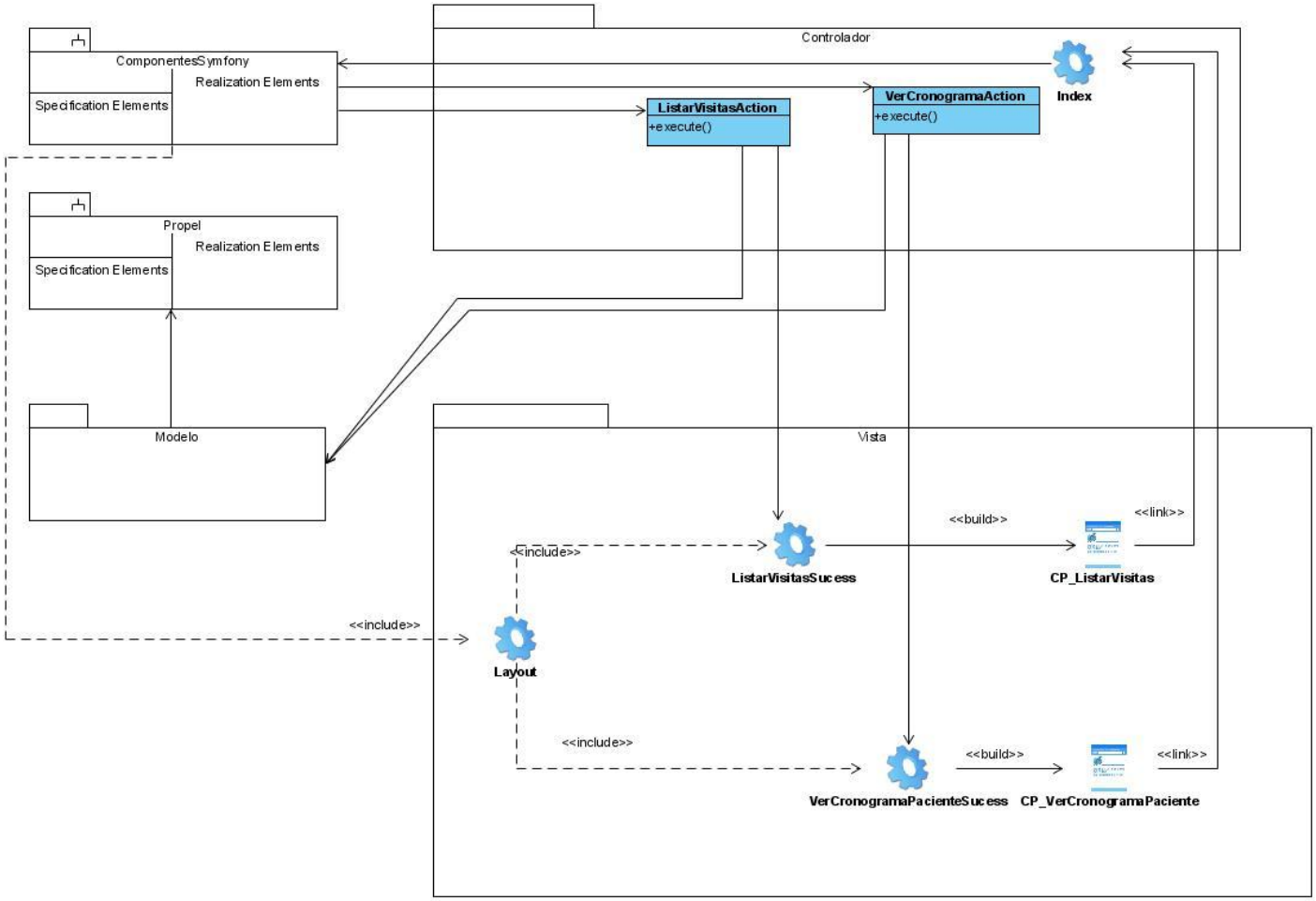


Fig. 8 Diagrama de clases del diseño del caso de uso Visualizar Cronograma

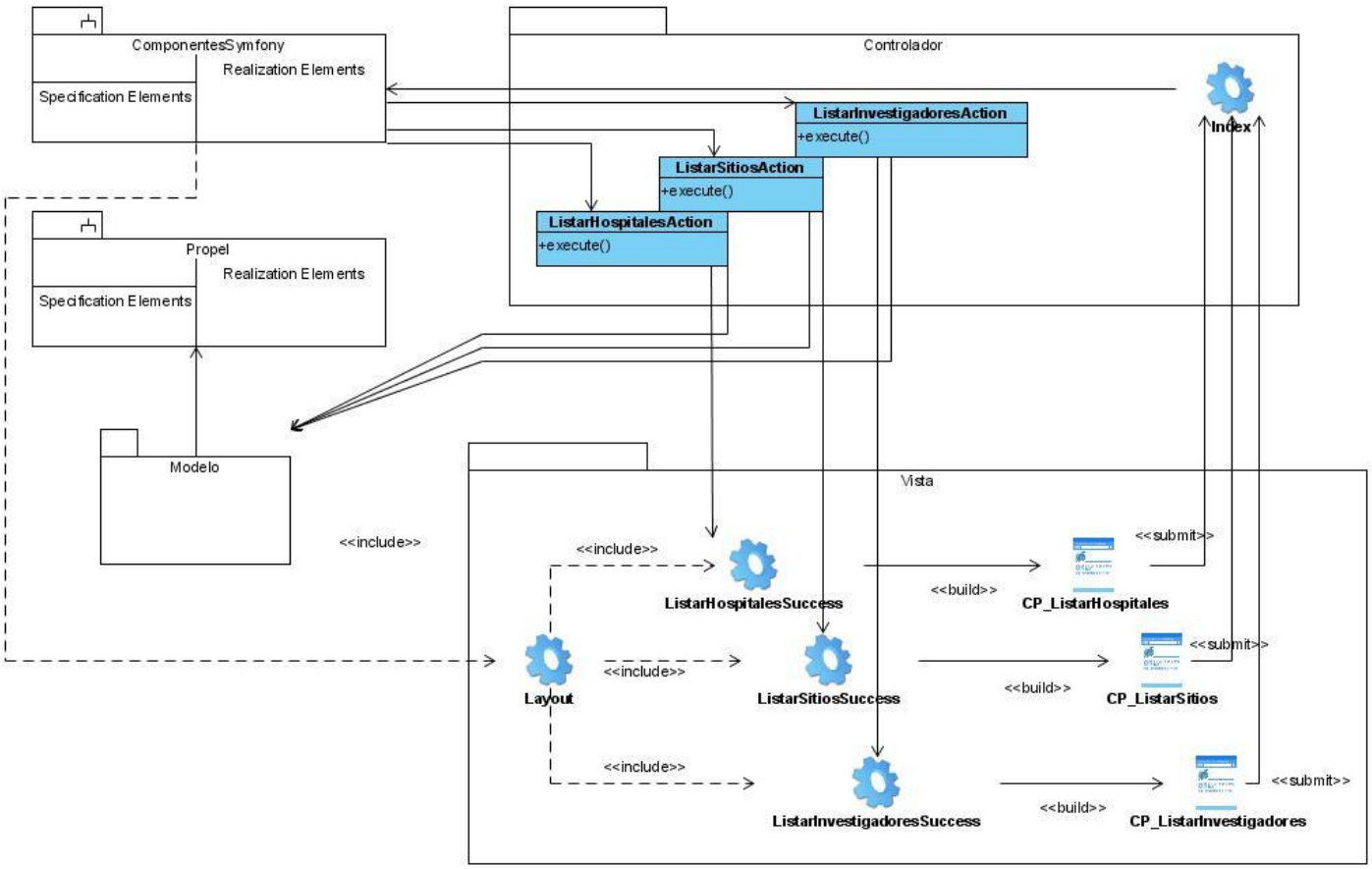


Fig. 9 Diagrama de clases del diseño del caso de uso Listar Investigadores

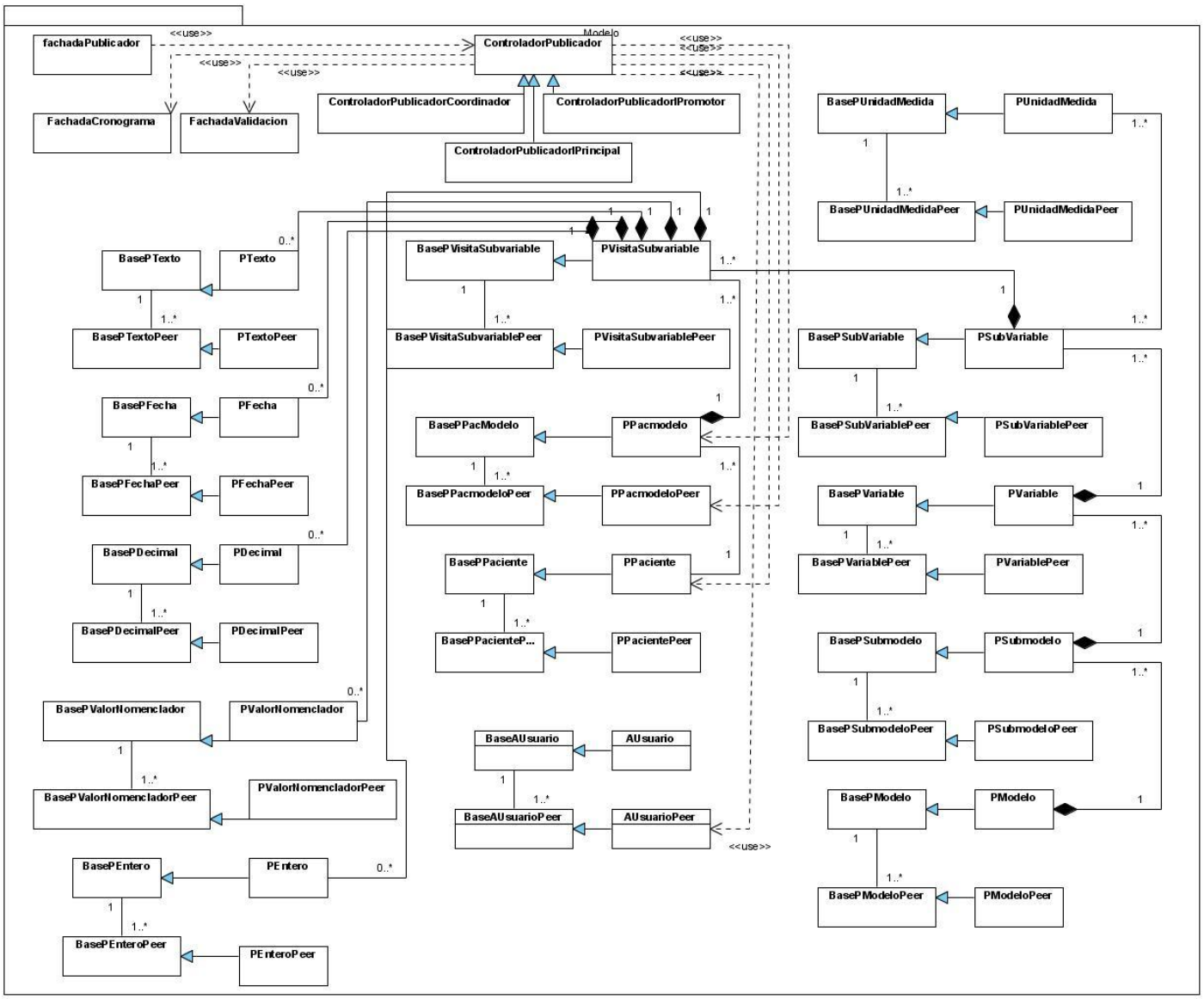


Fig. 10 Paquete Modelo

2.4.2 Descripción de las clases del diseño.

Tabla 2.4.2.1 Descripción de la Clase de Diseño: ControladorPublicador

Nombre: ControladorPublicador	
Tipo: Clase(Es la clase que contiene las funciones correspondientes al negocio a la hora de gestionar información referente a pacientes, como las inconsistencias que existen en los modelos de los mismos)	
Atributo	**Tipo**
usuarioActivo	
Responsabilidades	
Nombre:	**Descripción:**
getListadoAccionesModelo()	Devuelve el listado de acciones que el rol autenticado puede realizar sobre el modelo en cuestión. Es un método abstracto que será redefinido en las clases hijas de la presente.
getListadoInconsistenciasModelo()	Invoca una instancia de la clase fachada de validación y obtiene el listado de inconsistencias de un modelo para un paciente.
insertarPaciente()	Dado un paciente lo guarda haciendo uso del método save() implementado en la clase BasePPaciente e invoca a la fachada de cronograma para generar el cronograma específico del paciente insertado.
insertarDatosPacienteModelo()	Guarda el arreglo de subvariables pertenecientes a un modelo haciendo uso de la clase PPacModelo y a través de ella misma le cambia el estado al modelo que se está procesando. Invoca a la fachada de validación para validar el modelo pasado.
getListadoPacientes()	Devuelve un listado de pacientes producto de una consulta realizada a la clase PPacientePeer.

Tabla 2.4.2.2 Descripción de la Clase de Diseño: ControladorPublicadorCoordinador

Nombre: ControladorPublicadorCoordinador
Tipo: Clase(Esta clase hereda de ControladorPublicador por lo que tiene sus funcionalidades y

redefine el método getListadoAccioneModelo)	
Responsabilidades	
Nombre:	**Descripción:**
getListadoAccionesModelo()	Devuelve un arreglo con las acciones que puede realizar el coordinador sobre los modelos de forma tal que al cargarse dinámicamente los mismos solo se le habiliten las funcionalidades a las que tiene acceso
getListadoPacientes()	Invoca al método del padre con igual nombre pero pasándole el id del sitio al que pertenece el rol activo autenticado

Tabla 2.4.2.3 Descripción de la Clase de Diseño: ControladorPublicadorIPromotor

Nombre: ControladorPublicadorIPromotor	
Tipo: Clase(Esta clase hereda de ControladorPublicador por lo que tiene sus funcionalidades y redefine el método getListadoAccioneModelo , insertarPacientes y InsertarDatosModeloPacientes)	
Responsabilidades	
Nombre:	**Descripción:**
getListadoAccionesModelo()	Devuelve un arreglo con las acciones que puede realizar el coordinador sobre los modelos de forma tal que al cargarse dinámicamente los mismos solo se le habiliten las funcionalidades a las que tiene acceso
insertarPacientes()	Redefine este método para lanzar una excepción puesto que los investigadores promotores no tienen acceso a insertar un paciente
insertarDatosPacienteModelo()	Redefine este método para lanzar una excepción puesto que los investigadores promotores no tienen acceso a insertar datos de un paciente en un modelo. Solo pueden visualizar la información contenida

Tabla 2.4.2.4 Descripción de la Clase de Diseño: ControladorPublicadorIPrincipal

Nombre: ControladorPublicadorIPrincipalr	
Tipo: Clase(Esta clase hereda de ControladorPublicador por lo que tiene sus funcionalidades y redefine el método getListadoAccioneModelo e insertarPacientes)	
Responsabilidades	
Nombre:	**Descripción:**
getListadoAccionesModelo()	Devuelve un arreglo con las acciones que puede realizar el coordinador sobre los modelos de forma tal que al cargarse dinámicamente los mismos solo se le habiliten las funcionalidades a las que tiene acceso
insertarPacientes()	Redefine este método para lanzar una excepción puesto que los investigadores promotores no tienen acceso a insertar un paciente

Tabla 2.4.2.5 Descripción de la Clase de Diseño: FachadaPublicador

Nombre: FachadaPublicador	
Tipo: Clase(Esta clase devuelve un objeto de tipo controlador del usuario que se encuentre activo, sirviendo como único punto de entrada al submódulo Gestión de la información de pacientes y de los CRD)	
Nombre:	**Descripción:**
getInstancia()	Método que crea una instancia de la clase FachadaPublicador en caso de no estar instanciada en caso positivo devuelve la instancia que está creada.
factoriaControladorPublicador()	Devuelve un objeto de tipo controlador de tipo usuario activo para que los otros módulos puedan acceder a las responsabilidades del módulo publicador aislados de la lógica de negocio.

Tabla 2.4.2.6 Descripción de la Clase de Diseño: MostrarModeloAction.

Nombre: MostrarModeloAction	
Tipo: Clase(Esta clase es la encargada de procesar los modelos de los CRD diseñados para un ensayo clínico,)	
Nombre:	**Descripción:**
execute()	Método que invoca al método preparepage() o processpage() en dependencia de cual sea requerido
preparePage()	Prepara el modelo en dependencia del listado de acciones que tenga el usuario que está intentando visualizarlo. Si el modelo está vacío solamente podrá insertar datos el coordinador de la investigación en caso de que no este vacío pues se cargan las variables y se muestra el modelo , esta vista del modelo será en dependencia del tipo de usuario autenticado
processPage()	Método encargado de procesar la página después de haber sido abierta. En dependencia del botón que haya sido pulsado guarda la información, cambia el estado del modelo o cancela.

Tabla 2.4.2.7 Descripción de la Clase de Diseño: pub_PrincipalActions.

Nombre: pub_PrincipalActions	
Tipo: Clase(Esta clase es la clase action por defecto del módulo Publicador)	
Nombre:	**Descripción:**
execute()	Método que invoca al método preparepage() o processpage() en dependencia de cual sea requerido
executeIndexCoordinador ()	Prepara el modelo en dependencia del listado de acciones que tenga el usuario que esta intentando visualizarlo. Si el modelo está vacío solamente podrá insertar datos el coordinador de la investigación en caso de que no este vacío pues se cargan las variables y se muestra el modelo , esta vista del modelo será en dependencia del tipo de usuario autenticado

executeIndexIPrincipal()	Método encargado de procesar la página después de haber sido abierta. En dependencia del botón que haya sido pulsado guarda la información, cambia el estado del modelo o cancela.
executeIndexIPromotor()	Redirecciona para el listado de hospitales que es la página principal del módulo Publicador.

Tabla 2.4.2.8 Descripción de la Clase de Diseño: InsertarPacienteAction.

Nombre: InsertarPacienteAction	
Tipo: Clase(Esta clase es la clase action que posibilita insertar un paciente)	
Nombre:	**Descripción:**
execute()	Pasa al modelo de inclusión las iniciales del paciente y el id de la historia clínica.
handleError()	Es el encargado de manejar los errores ocurridos.

Tabla 2.4.2.9 Descripción de la Clase de Diseño: ModeloInclusionAction

Nombre: ModeloInclusionAction	
Tipo: Clase(Esta clase es la clase action posibilita incluir o no el paciente en un ensayo)	
Nombre:	**Descripción:**
execute()	Este método se encarga de crear un nuevo paciente insertarlo y notificarle a la fachada de cronograma la inserción de un paciente para que esta habilite la posibilidad de guardar el modelo de inclusión, una vez habilitada se procede a guardar las subvariables con la ayuda de un objeto de tipo ControladorPublicador que es la que porta el método que permite insertar un paciente, posteriormente notifica el llenado del modelo a la fachada de cronograma.

Tabla 2.4.2.10 Descripción de la Clase de Diseño: ListarHospitalesAction

Nombre: ListarHospitalesAction	
Tipo: Clase(Esta clase es la encargada de listar los hospitales involucrados en un ensayo)	
Nombre:	**Descripción:**
execute()	Este método se encarga de pedirle a AHospitalPeer un pager de hospitales.

Tabla 2.4.2.11 Descripción de la Clase de Diseño: ListarSitiosAction

Nombre: ListarSitiosAction	
Tipo: Clase(Esta clase es la encargada de listar los sitios pertenecientes a un hospital)	
Nombre:	**Descripción:**
execute()	Este método se encarga de pedirle a AHospitalPeer un pager de hospitales.

Tabla 2.4.2.12 Descripción de la Clase de Diseño: ListarInvestigadoresAction

Nombre: ListarInvestigadoresAction	
Tipo: Clase(Esta clase es la encargada de listar los investigadores pertenecientes a un sitio)	
Nombre:	**Descripción:**
execute()	Este método se encarga de pedirle a AUsuarioPeer a través del método getUsua() el listado de investigadores de un sitio.

Tabla 2.4.2.13 Descripción de la Clase de Diseño: ListarVisitasAction

Nombre: ListarVisitasAction	
Tipo: Clase(Esta clase es la encargada de listar las visitas de un paciente)	
Nombre:	**Descripción:**
execute()	Este método se encarga de listar las visitas de un paciente interactuando con la fachada de cronograma a través de los métodos obtenerModelosAsintomaticos() y obtenerModelosSintomaticos() además de redireccionar para un modelo en específico en caso de que el usuario decida visualizarlo.

Tabla 2.4.2.14 Descripción de la Clase de Diseño: VerCronogramaPacienteAction

Nombre: VerCronogramaPacienteAction	
Tipo: Clase(Esta clase es la encargada de visualizar el cronograma específico de un paciente)	
Nombre:	**Descripción:**
execute()	Este método se encarga de obtener el paciente al cual se le desea visualizar el cronograma específico, posteriormente interactúa con la fachada de cronograma y pide el listado de modelos asintomáticos junto con el de los sintomáticos y los lista.

Tabla 2.4.2.15 Descripción de la Clase de Diseño: PPacientePeer

Nombre: PPacientePeer	
Tipo: Clase(Esta clase es la encargada de la abstracción de datos de un paciente)	
Nombre:	**Descripción:**
getPaciente()	Devuelve un listado de pacientes dado un id
getPacientePager()	Devuelve un listado de pacientes a través de un sfPropelPager dado un Criteria.

Tabla 2.4.2.16 Descripción de la Clase de Diseño: PPacModelo

Nombre: PPacModelo	
Tipo: Clase(Esta clase es la encargada del acceso a los datos de un modelo)	
Nombre:	**Descripción:**
guardarSubVariables()	Este método se encarga de guardar todas las subvariables pertenecientes a un modelo.
cargarSubVariables()	Este método se encarga de cargar todas las subvariables de un modelo.
cambiarEstadoModelo()	Este método se encarga de cambiarle el estado a un modelo.

Tabla 2.4.2.17 Descripción de la Clase de Diseño: PPacModeloPeer

Nombre: PPacModeloPeer

Tipo: Clase(Esta clase es la encargada de la abstracción de datos de los modelos)	
Nombre:	**Descripción:**
getDatosRelevantes()	Este método se encarga de devolver los datos relevantes de la relación entre paciente y modelo

Tabla 2.4.2.18 Descripción de la Clase de Diseño: PSubVariables

Nombre: PSubVariables	
Tipo: Clase(Esta clase es la encargada del acceso a datos de las subvariables)	
Nombre:	**Descripción:**
InsertarSuvariable ()	Este método se encarga de guardar las subvariables de un modelo

Tabla 2.4.2.19 Descripción de la Clase de Diseño: PVisitaSubvariables

Nombre: PVisitaSubvariables	
Tipo: Clase(Esta clase es la encargada del acceso a datos de la relación entre visita y subvariables)	
Nombre:	**Descripción:**
guardarVisitaSubvariable ()	Este método se encarga de guardar las subvariables recogidas en una visita del paciente

2.5 Diagramas de interacción del diseño.

Los diagramas de secuencia del diseño muestran los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos, para llevar a cabo la funcionalidad descrita por el escenario. Es una representación visual de lo que ocurrirá entre las clases que serán implementadas, mostrando la colaboración entre las mismas a través de mensajes. Los mensajes, que implican responsabilidades son transformados en operaciones que finalmente se convertirán en métodos que definirán las responsabilidades de las clases. A continuación se mostrarán los diagramas de secuencia que guiarán la implementación del submódulo Gestión de la información de pacientes y de los CRD del SIMDECC, con sus respectivos escenarios.

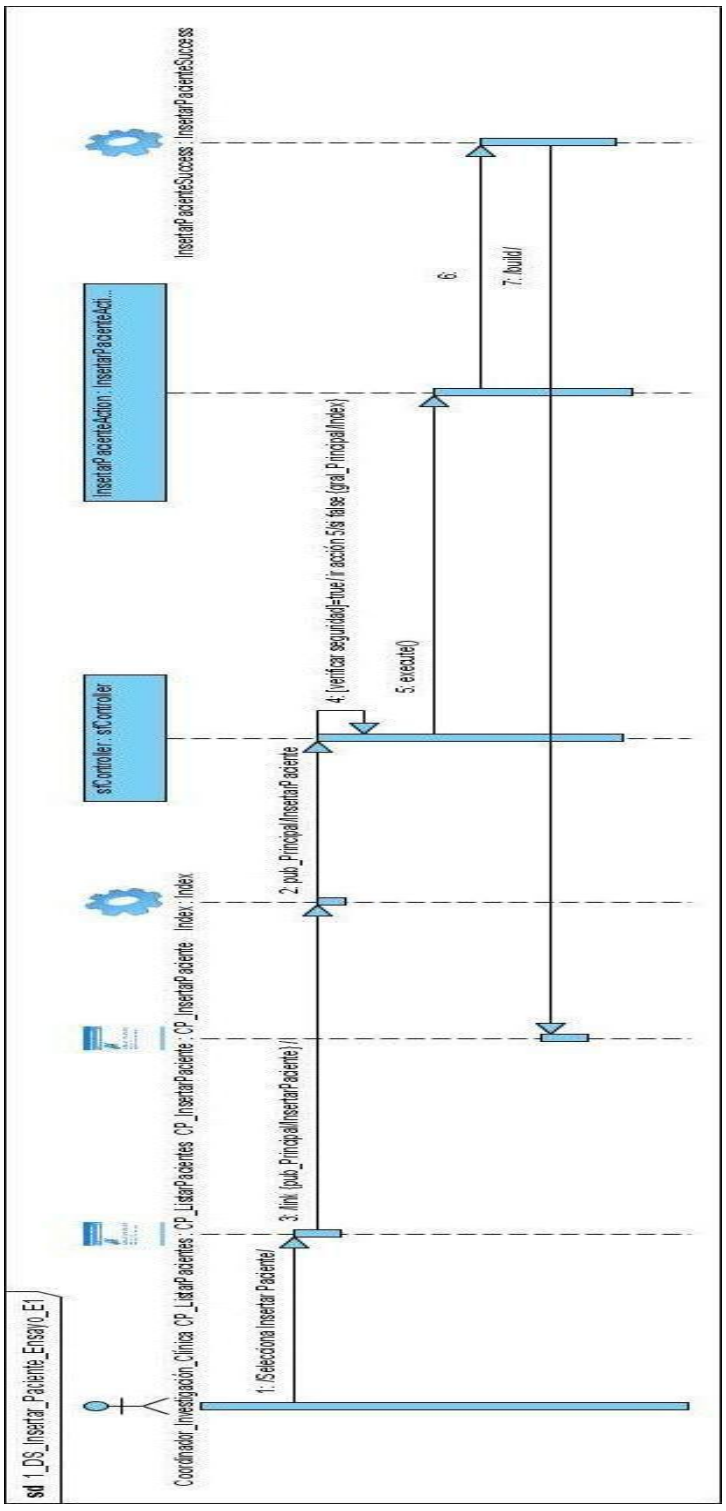


Fig. 11 Diagrama de secuencia del caso de uso Insertar Paciente Ensayo escenario 1

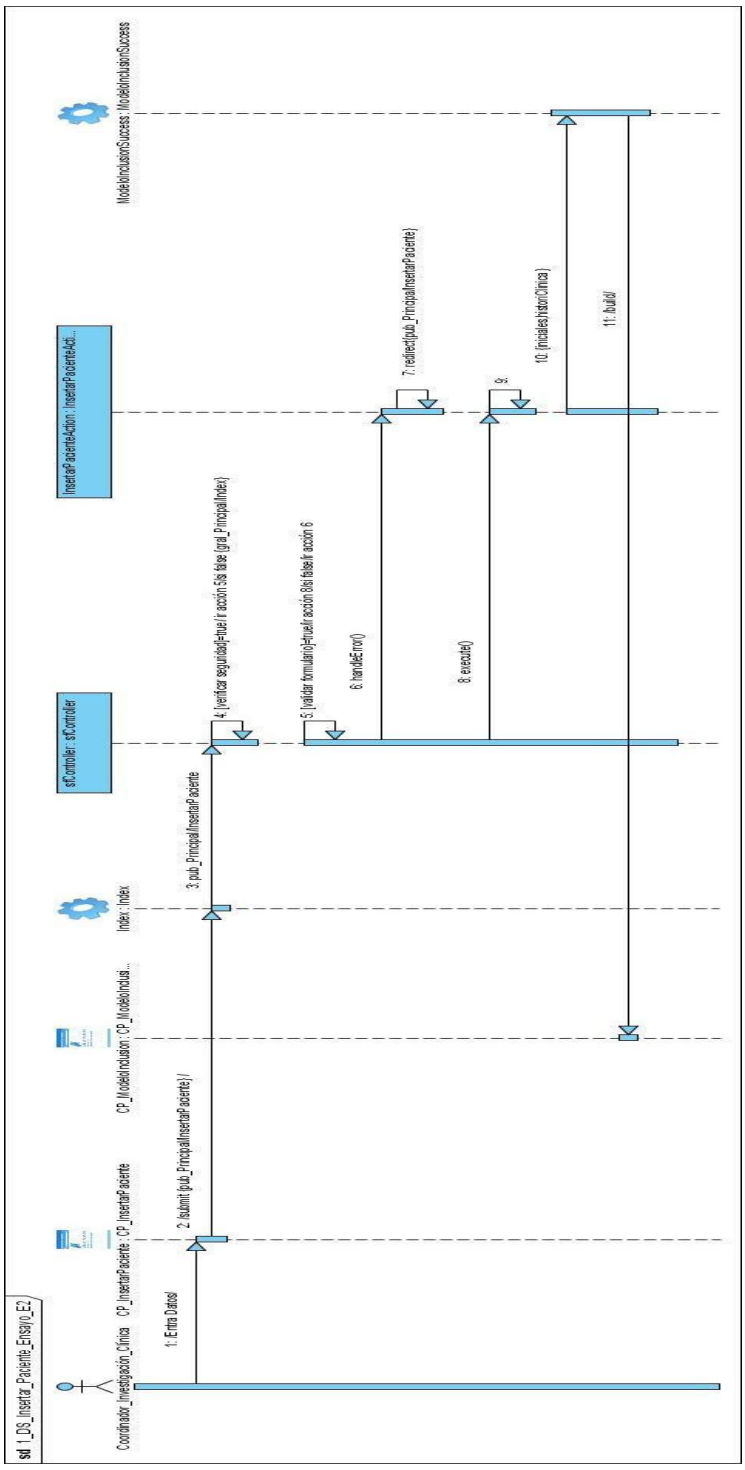


Fig. 12 Diagrama de secuencia del caso de uso Insertar Paciente Ensayo escenario 2

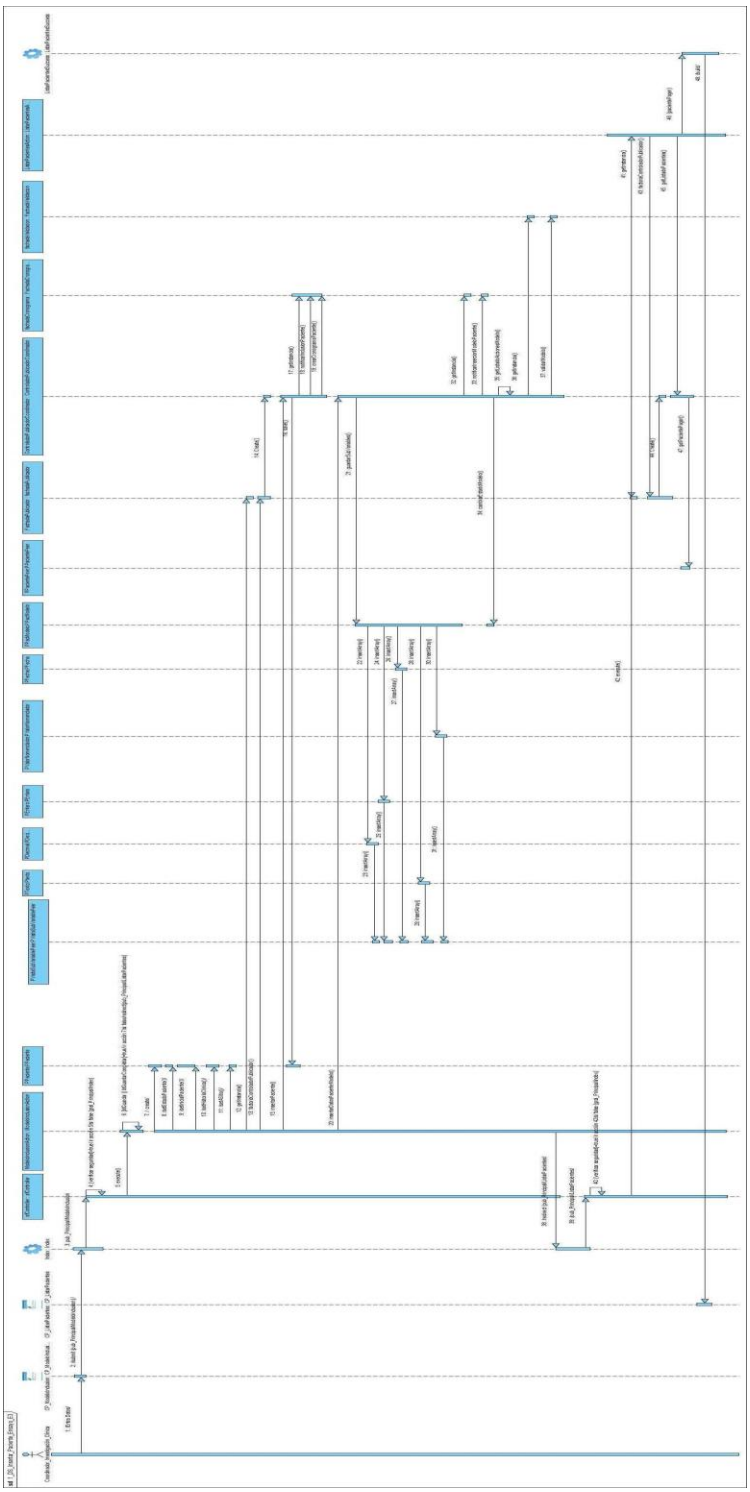


Fig. 13 Diagrama de secuencia del caso de uso Insertar Paciente Ensayo escenario 3

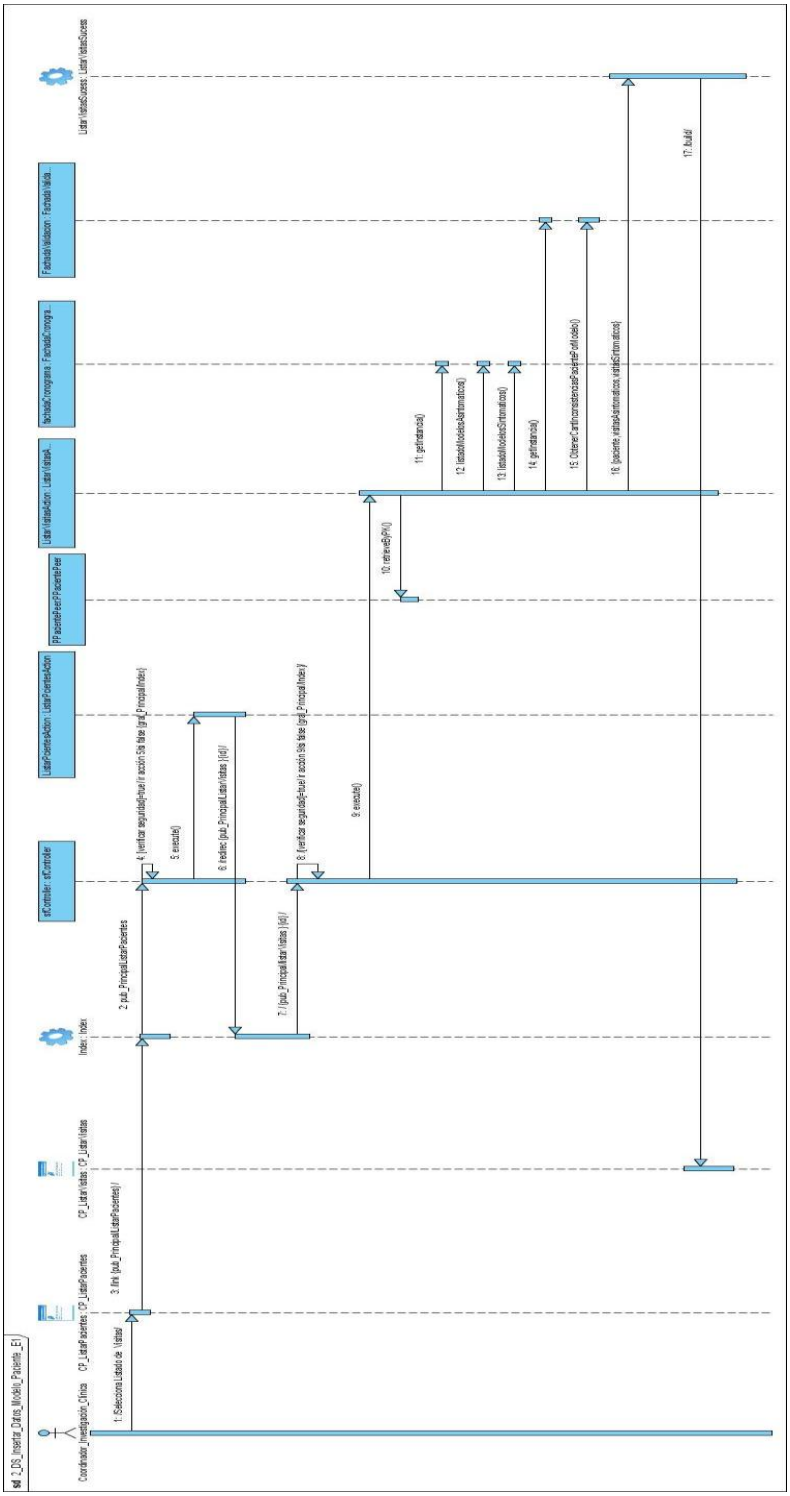


Fig. 14 Diagrama de secuencia del caso de uso Insertar Datos Modelos Paciente escenario 1

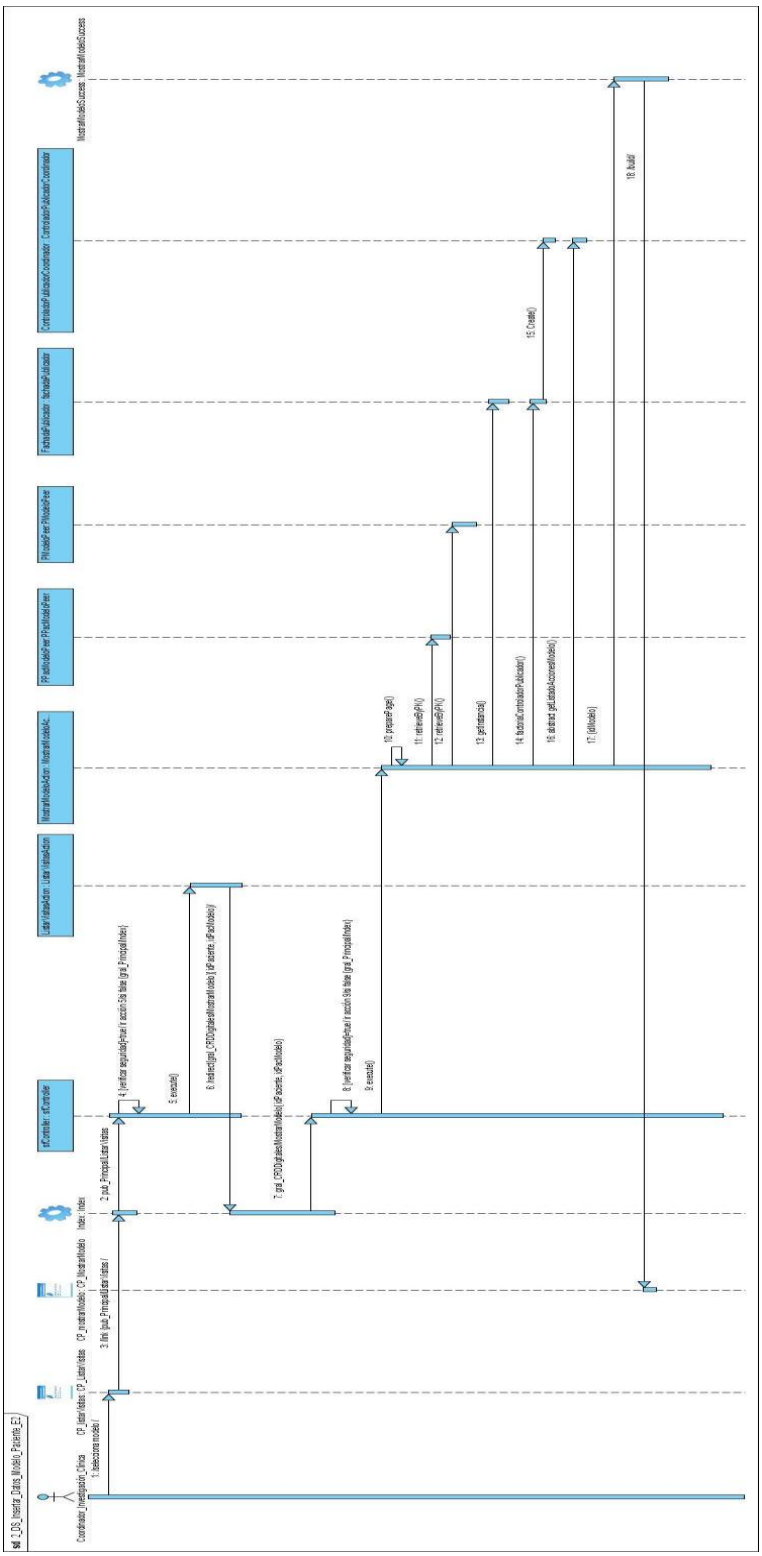


Fig. 15 Diagrama de secuencia del caso de uso Insertar Datos Modelos Paciente escenario 2

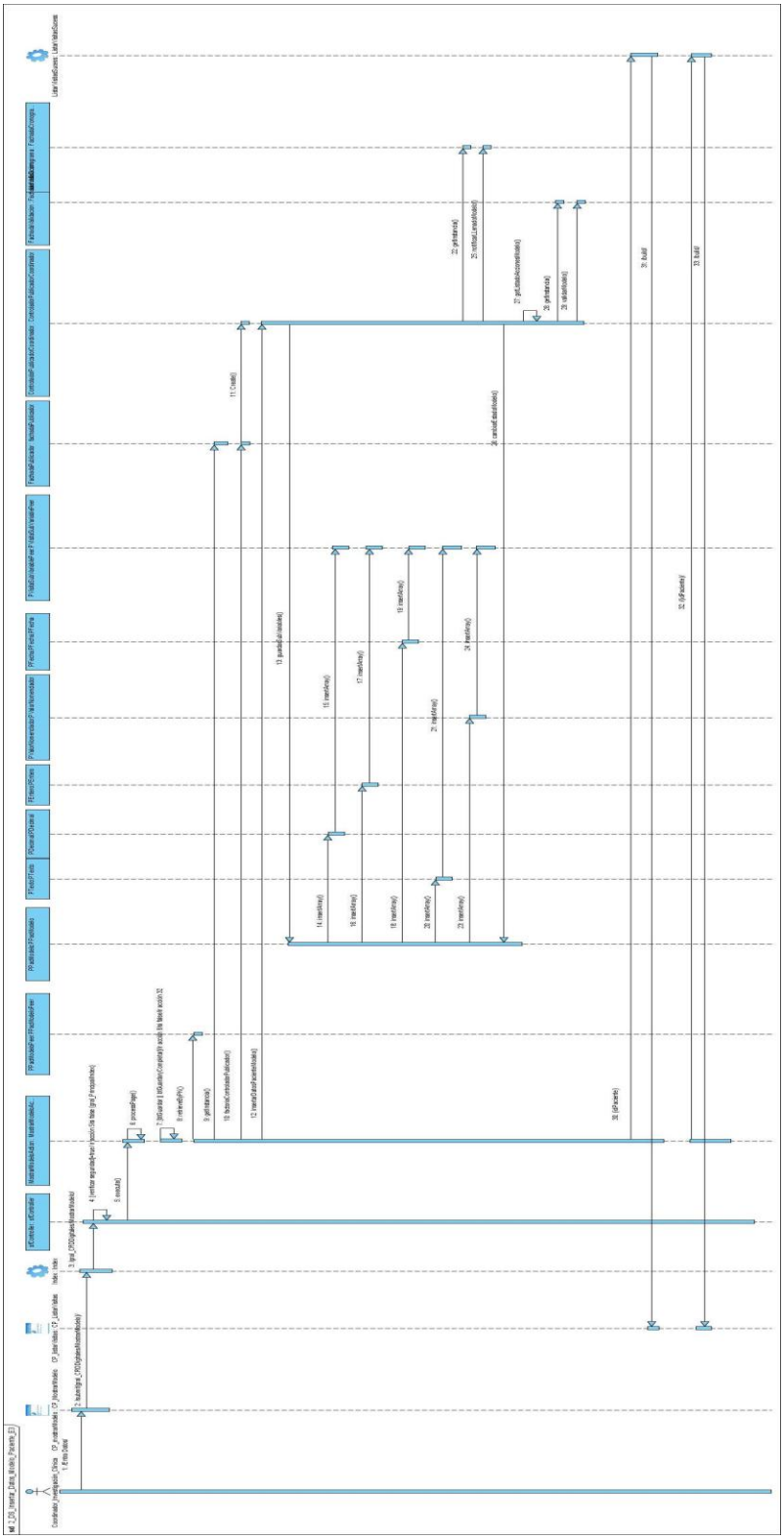


Fig. 16 Diagrama de secuencia del caso de uso Insertar Datos Modelos Paciente escenario 3

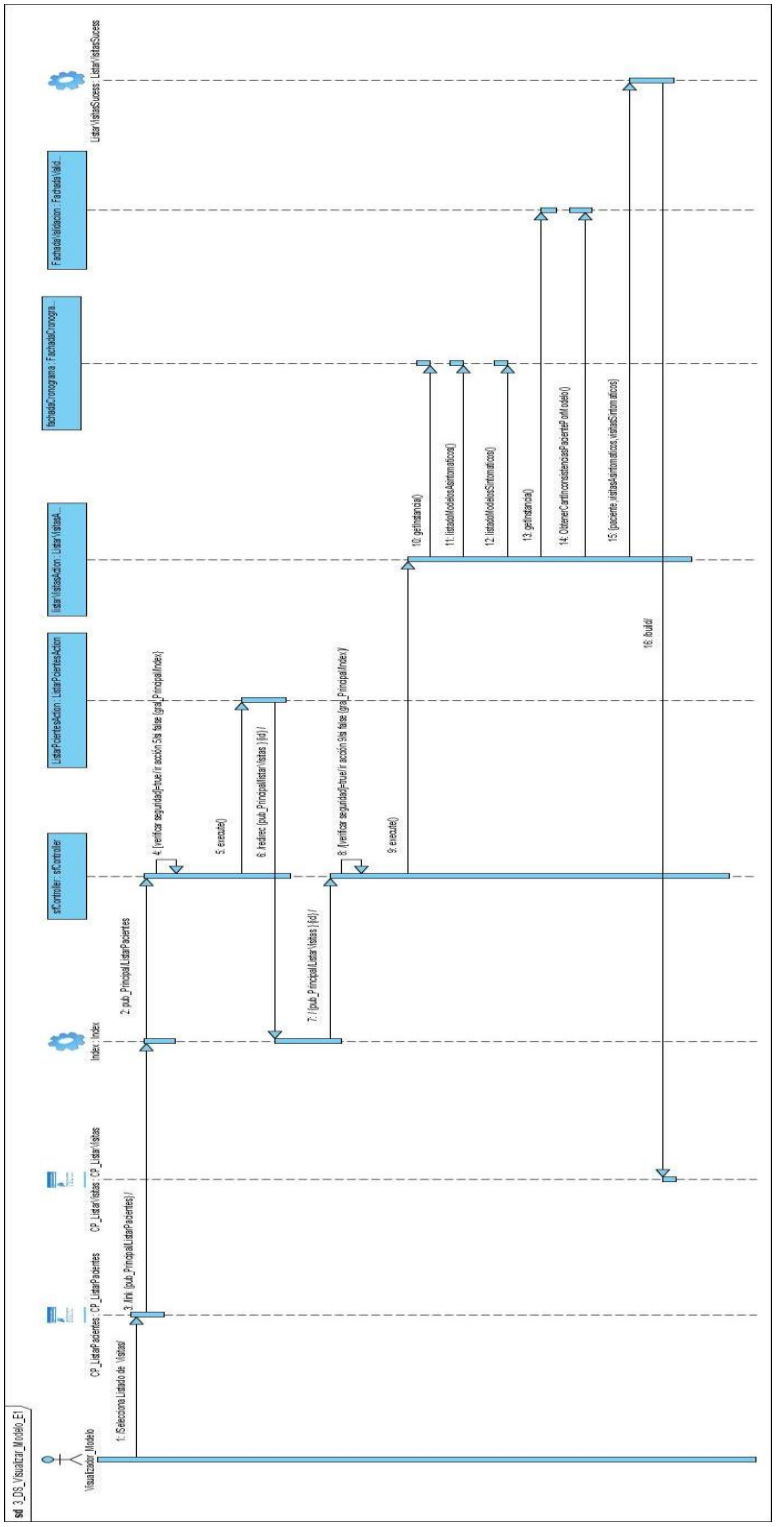


Fig. 17 Diagrama de secuencia del caso de uso Visualizar Datos Modelo escenario 1

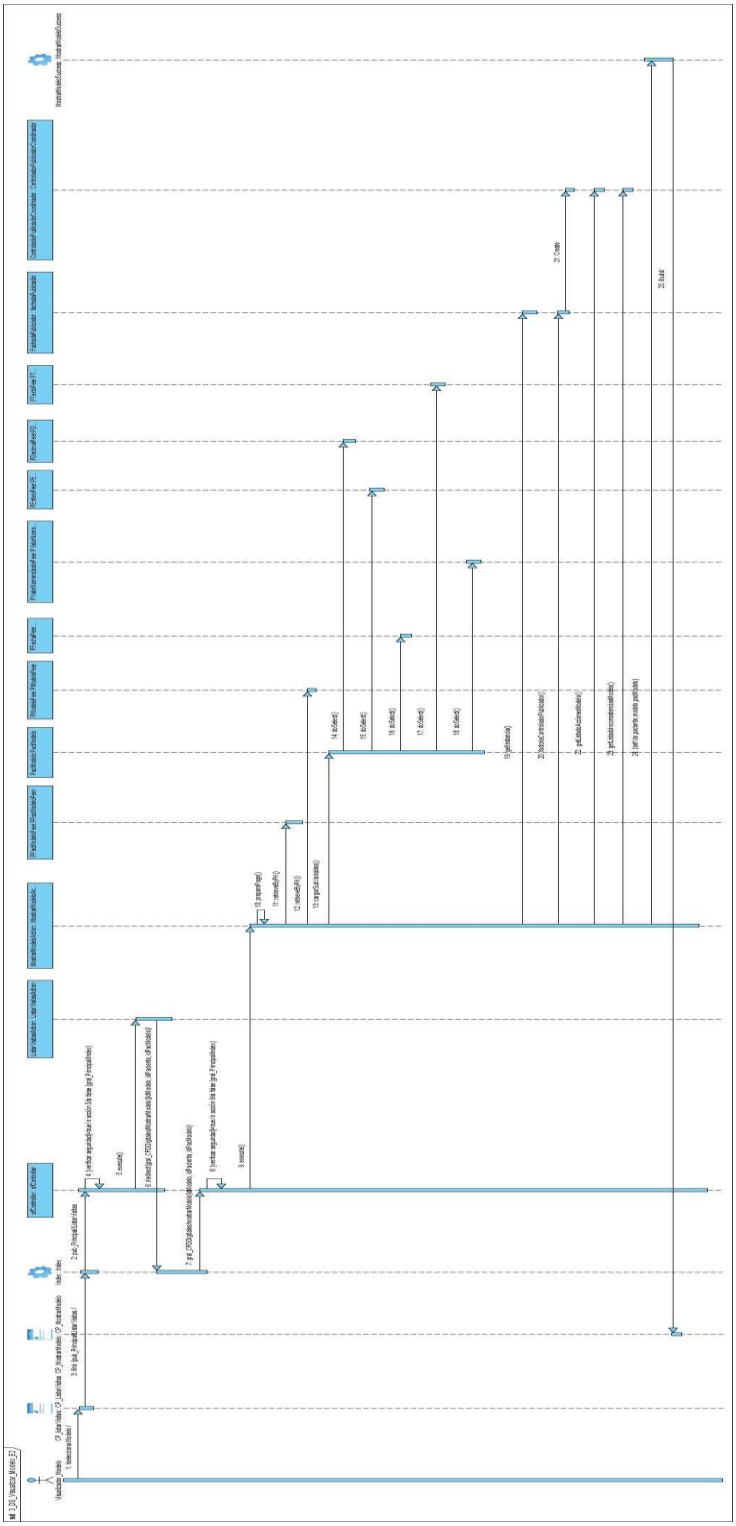


Fig. 18 Diagrama de secuencia del caso de uso Visualizar Datos Modelo escenario 2

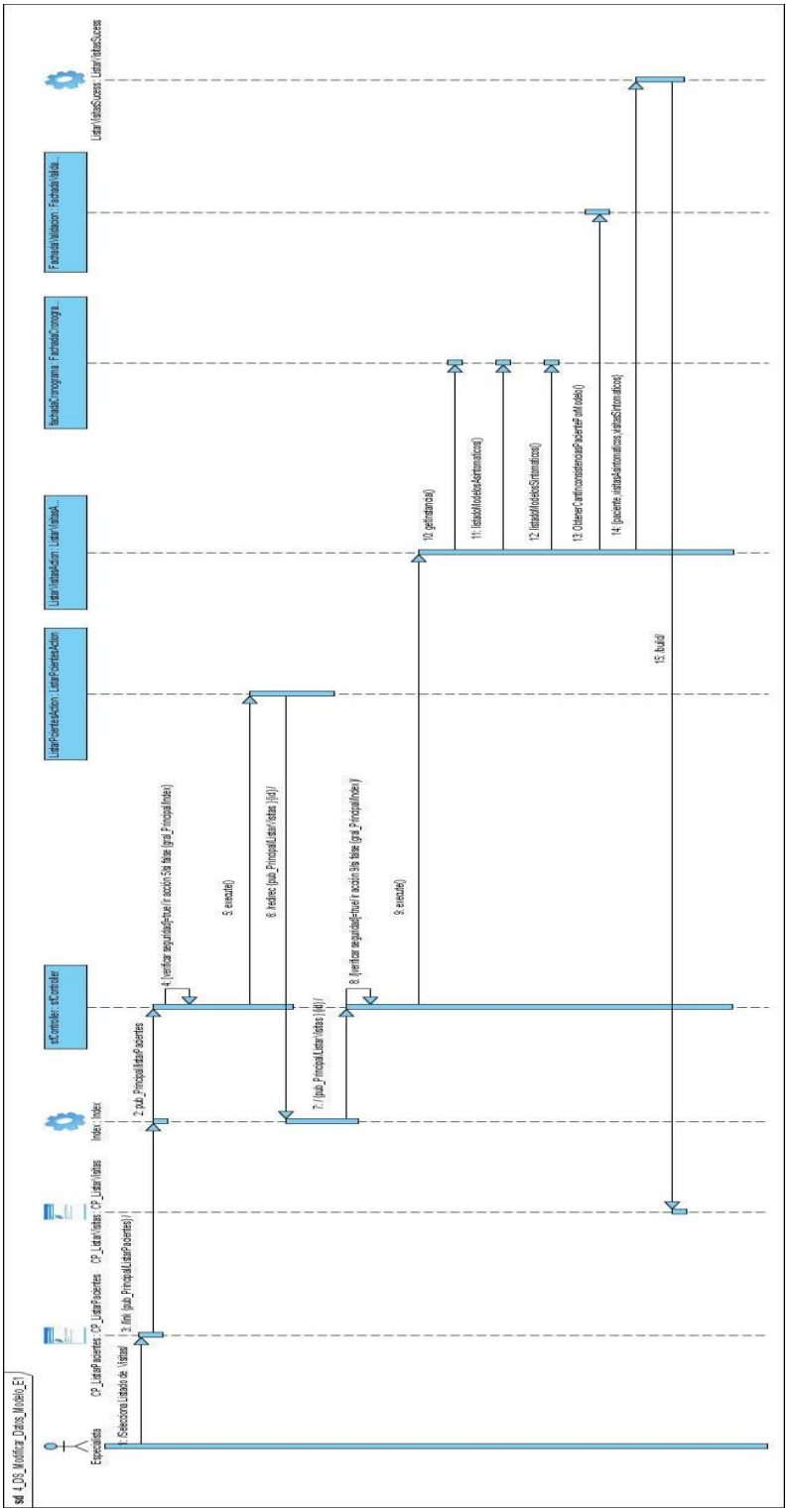


Fig. 19 Diagrama de secuencia del caso de uso Modificar Datos Modelo escenario 1

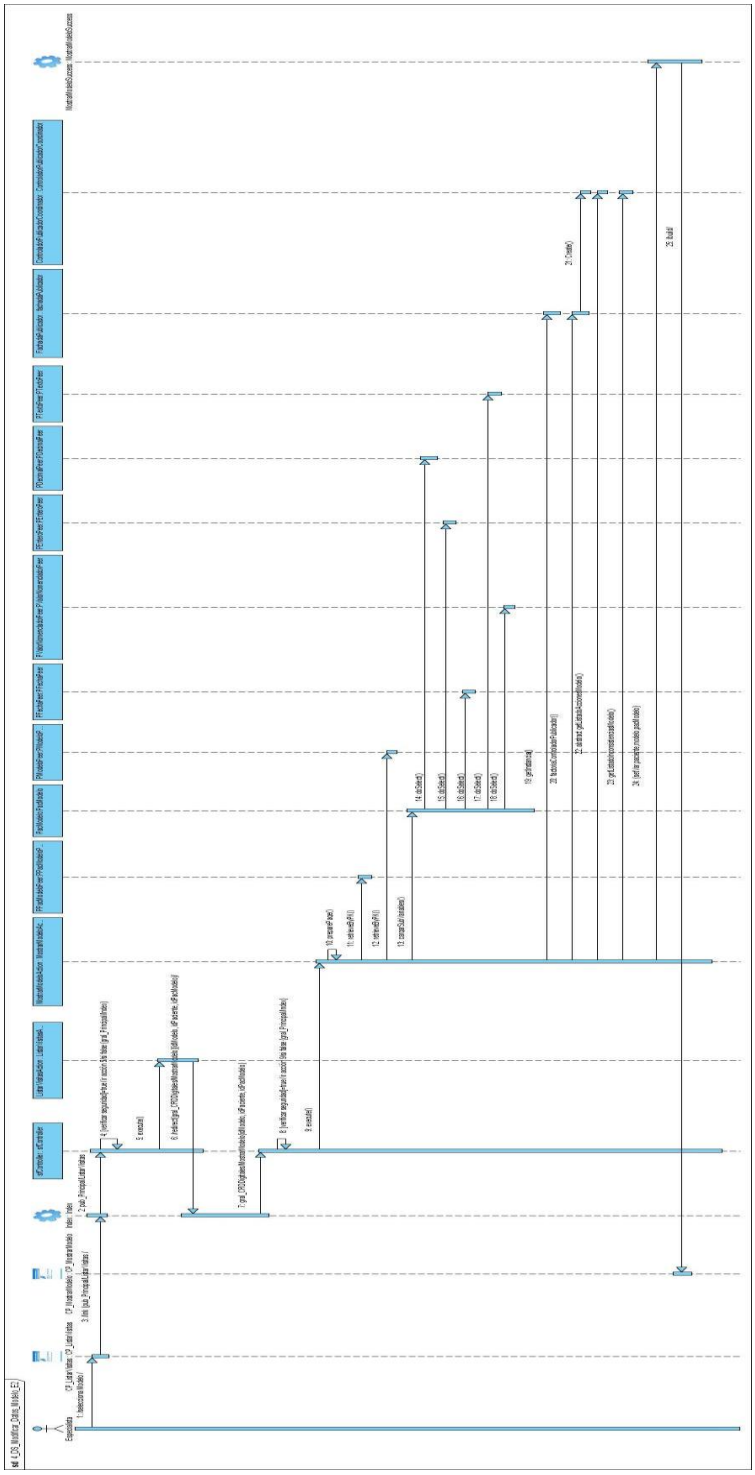


Fig. 20 Diagrama de secuencia del caso de uso Modificar Datos Modelo escenario 2

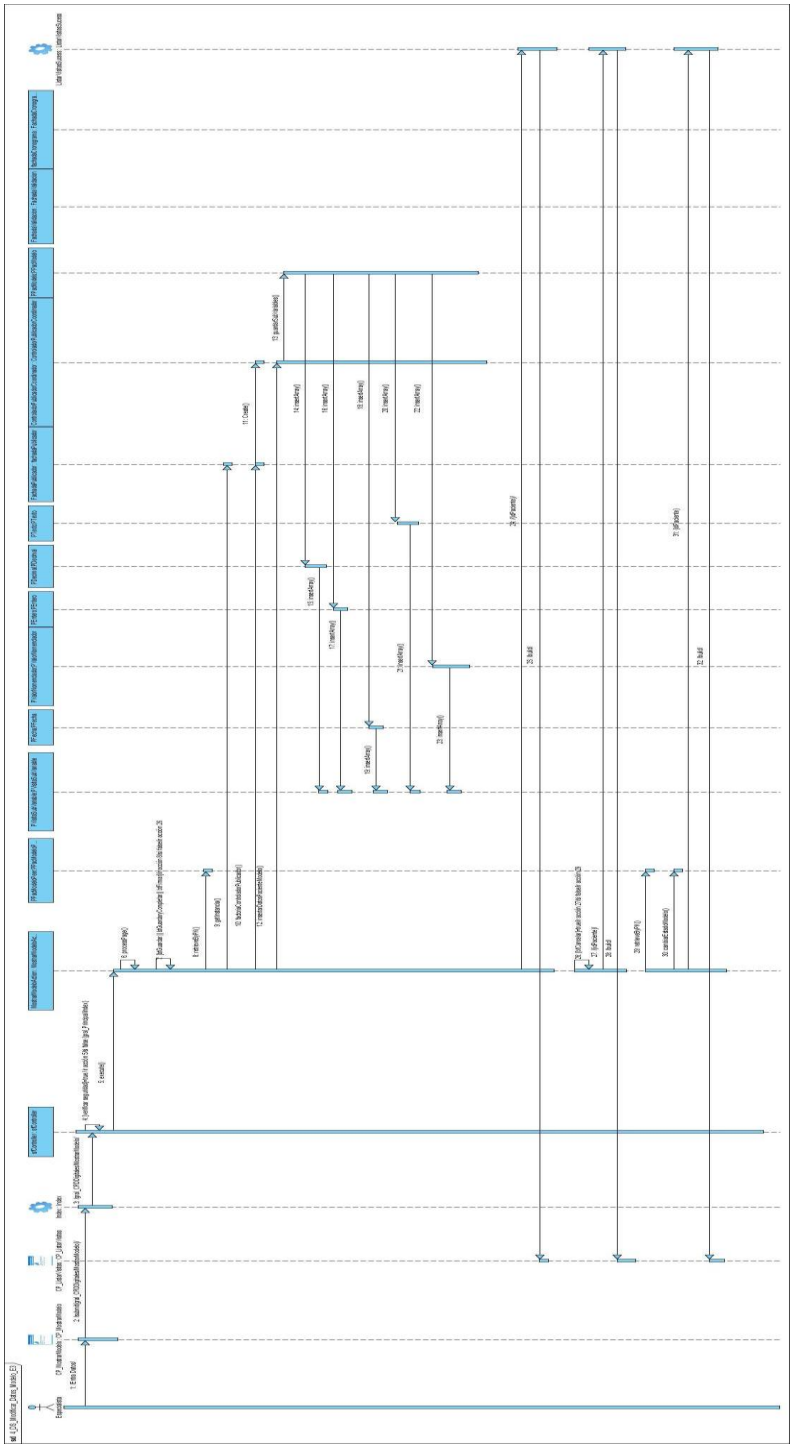


Fig. 21 Diagrama de secuencia del caso de uso Modificar Datos Modelo escenario 3

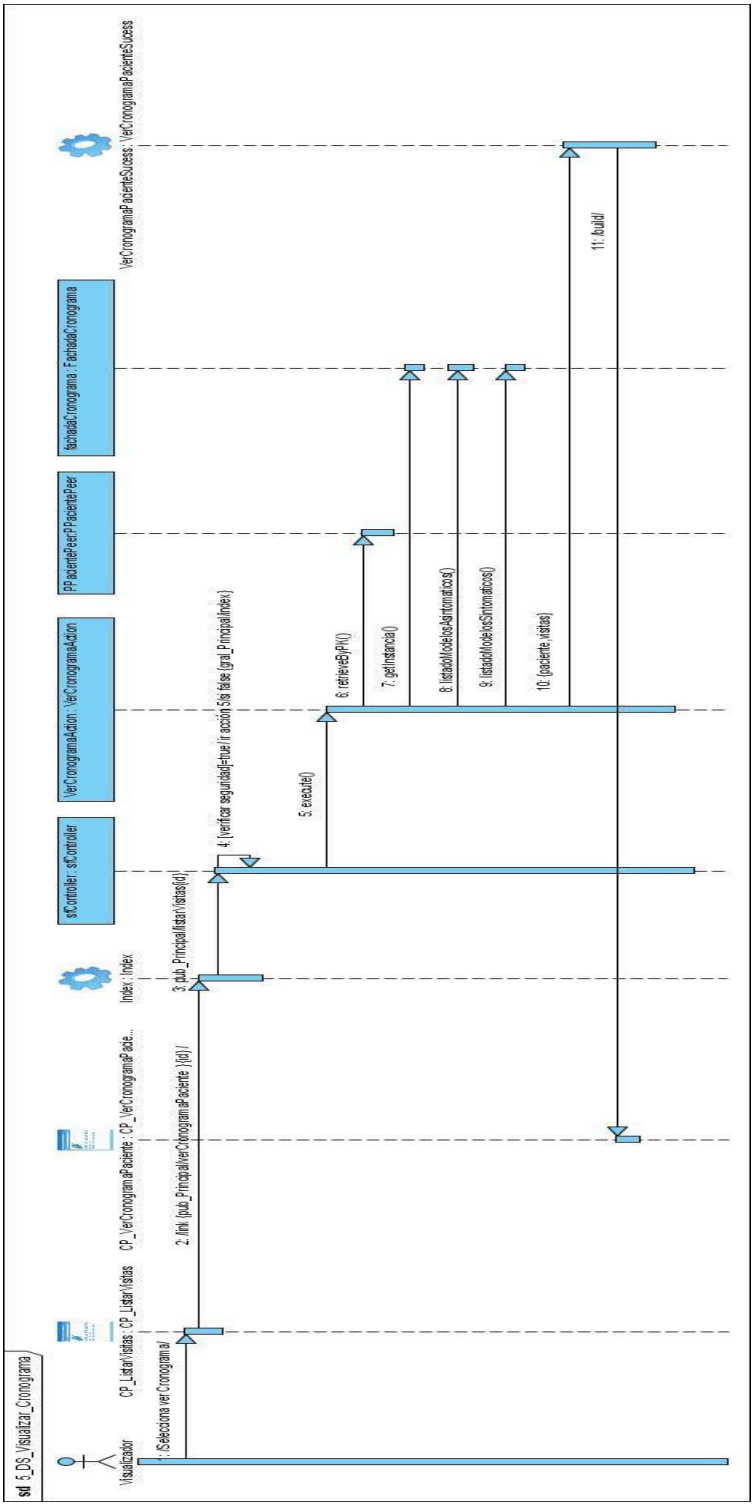


Fig. 22 Diagrama de secuencia del caso de uso Visualizar Cronograma

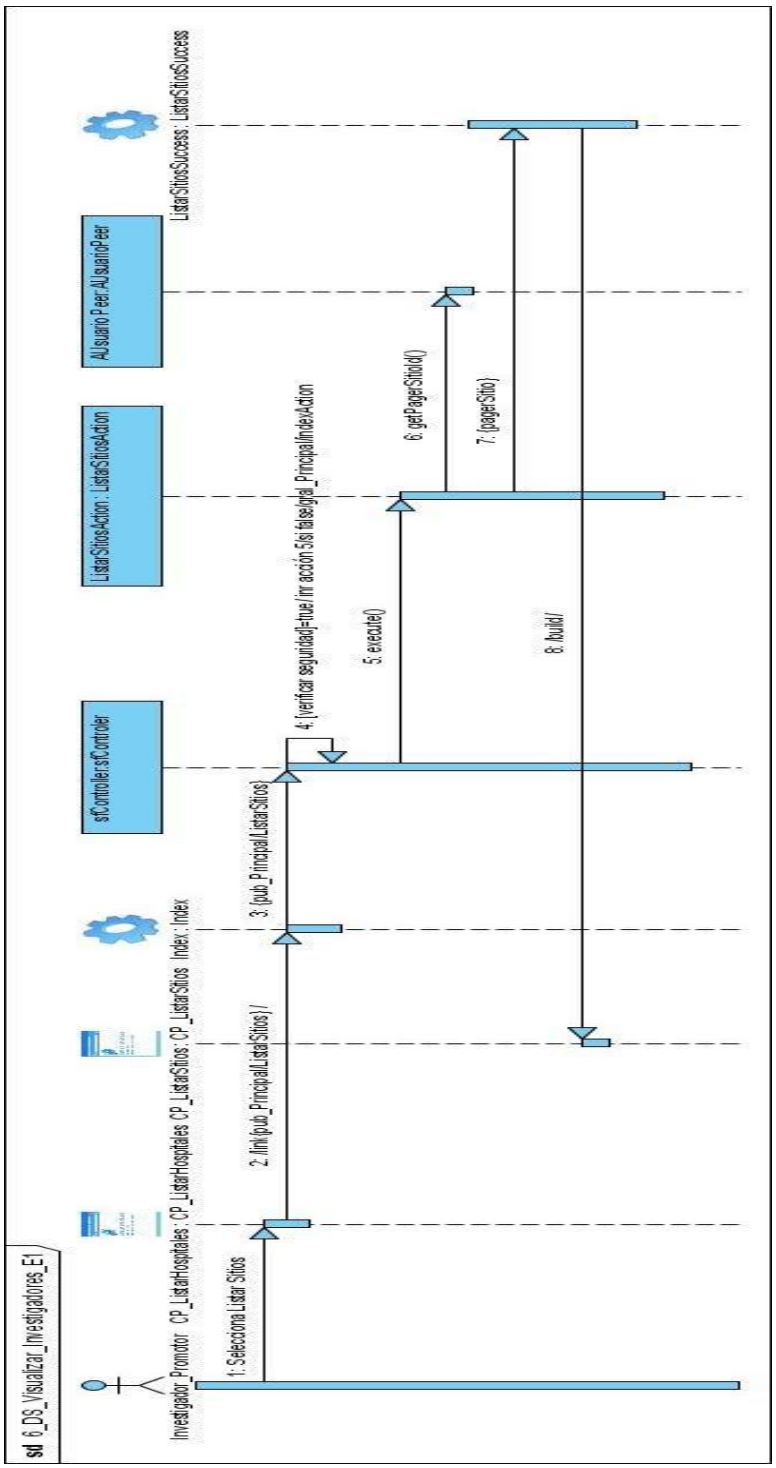


Fig. 23 Diagrama de secuencia del caso de uso Visualizar Investigadores escenario 1

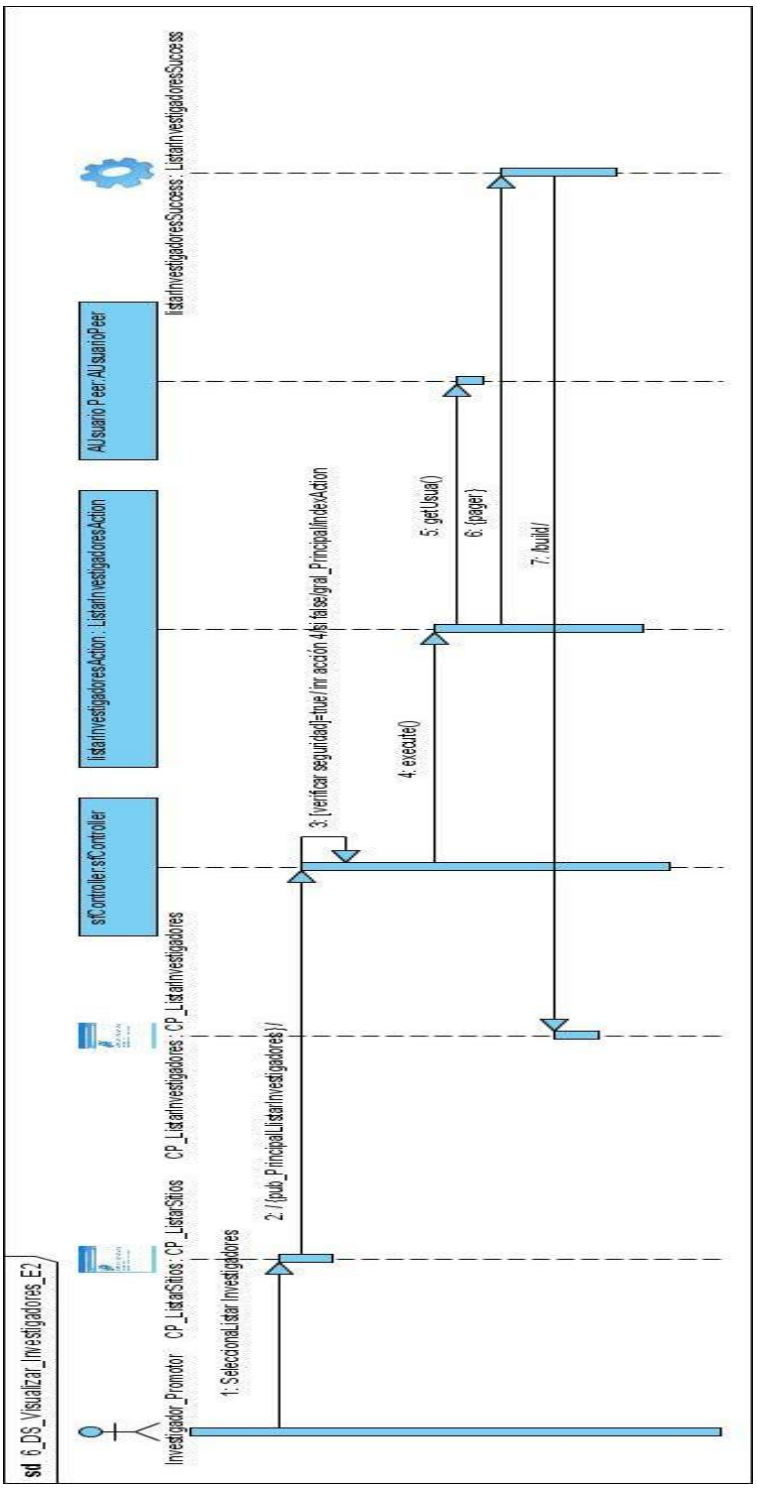


Fig. 24 Diagrama de secuencia del caso de uso Visualizar Investigadores escenario 2

2.6 Mapa de Navegación.

A través de la presentación del mapa de navegación se comprenderá mejor el funcionamiento del sistema, es válido aclara que las páginas en color lila son solo de acceso para el rol de Investigador Promotor, las blancas son accesibles por todos los roles y las amarillas solo accesibles desde el rol de Coordinador de la Investigación clínica. Las páginas de acceso común visualizarán su información de forma dinámica, de modo tal que se de el caso de que dos roles no vean la misma presentación de los datos accediendo a la misma página.

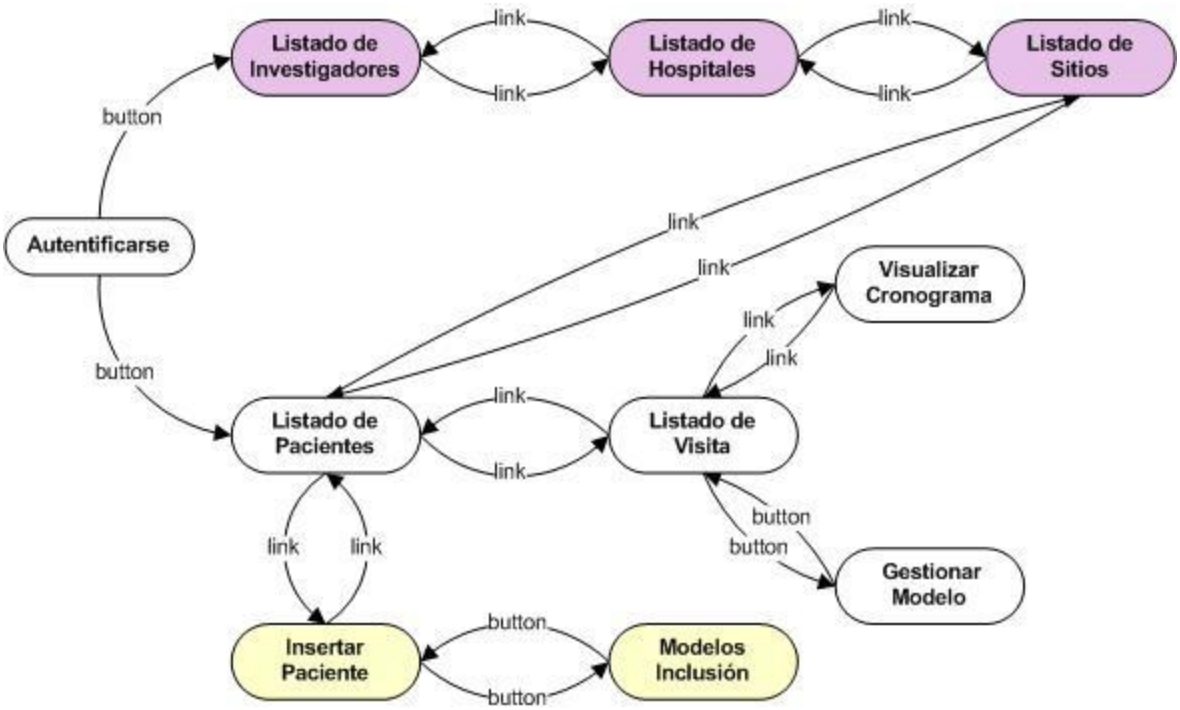


Fig. 25 Mapa de Navegación del submódulo Gestión de la información de pacientes y de los CRD del SIMDECC

2.7 Prototipos de interfaz de usuarios.

En el desarrollo del submódulo Gestión de la información de pacientes y de los CRD se elaboraron los prototipos de interfaz de usuario con la herramienta Kompozer. Sobre los prototipos obtenidos se realizó la implementación del sistema. Se diseñaron teniendo en cuenta el entorno científico donde serán aplicados, por lo que presentan pocos colores. Los prototipos diseñados cuentan con interfases para dar solución a todos los RF y cuentan con la aprobación del cliente, una representación de ellos se encuentra en el anexo 3 del presente trabajo de diploma.

2.8 Modelo de datos.

En este epígrafe se explica todo lo pertinente al modelo de datos de la aplicación desarrollada. Se describe como se representan los datos, ilustrándolo a través de la presentación del diagrama de clases persistentes y del modelo físico.

Antes de mostrar los diagramas, se cree conveniente explicar como se implementa el acceso a datos en el proyecto EC. La implementación de la capa de acceso a datos está ligada al uso del framework de desarrollo escogido, Symfony, el mismo contiene una potente herramienta llamada Propel. Propel es una capa de tipo ORM que permitirá el acceso y las modificaciones de los datos a través de objetos y la obtención de resultados de consulta, seleccionando los mismos según criterios definidos. En las clases generadas por Propel es donde se implementará la mayor parte de la lógica del negocio. Esta capa intermedia facilita la abstracción de los datos, de forma tal que los requerimientos necesarios para dar solución al problema propuesto, se puedan realizar de forma sencilla desde un entorno orientado a objetos. Cuando otro módulo necesite por ejemplo, el listado de pacientes, solamente invoca a un método del modelo de datos, a través de un objeto controlador devuelto por la fachada de publicador, sin conocer cómo este está implementado. Si el método dedicado a listar pacientes necesita ser cambiado solo se modificará en el modelo sin influir en la integración del proyecto.

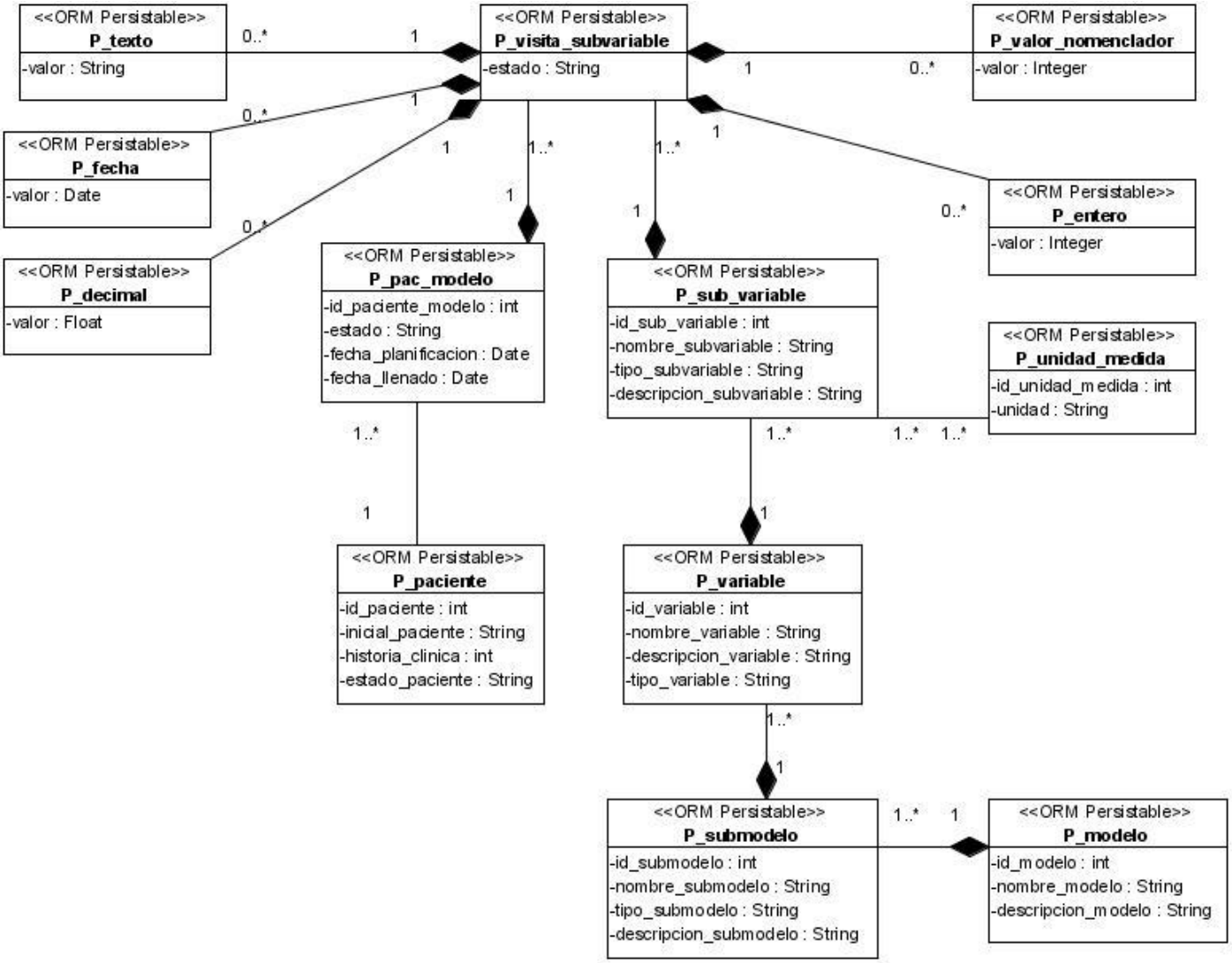


Fig. 26 Diagrama de clases persistentes

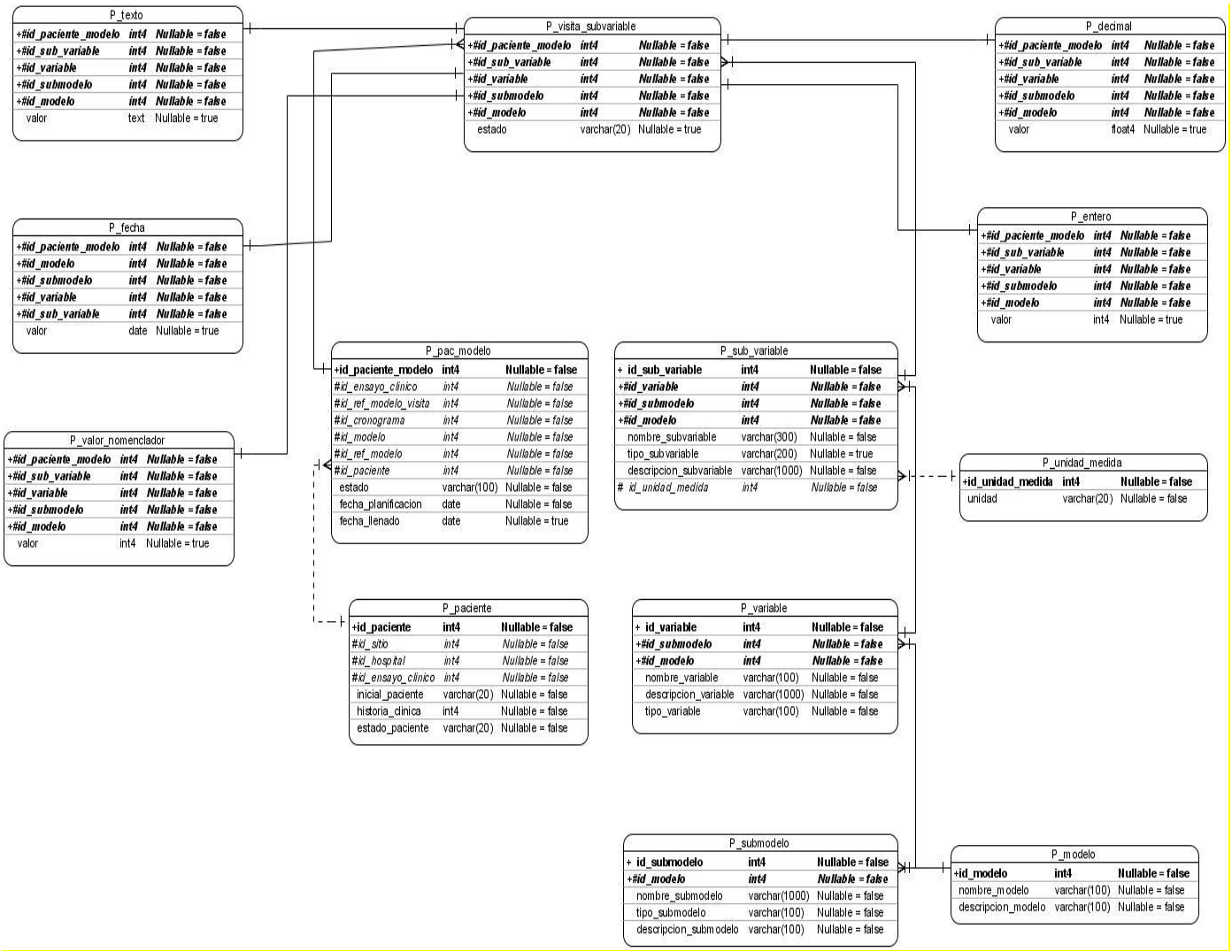


Fig. 27 Modelo Físico

2.9 Diagrama de Despliegue.

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

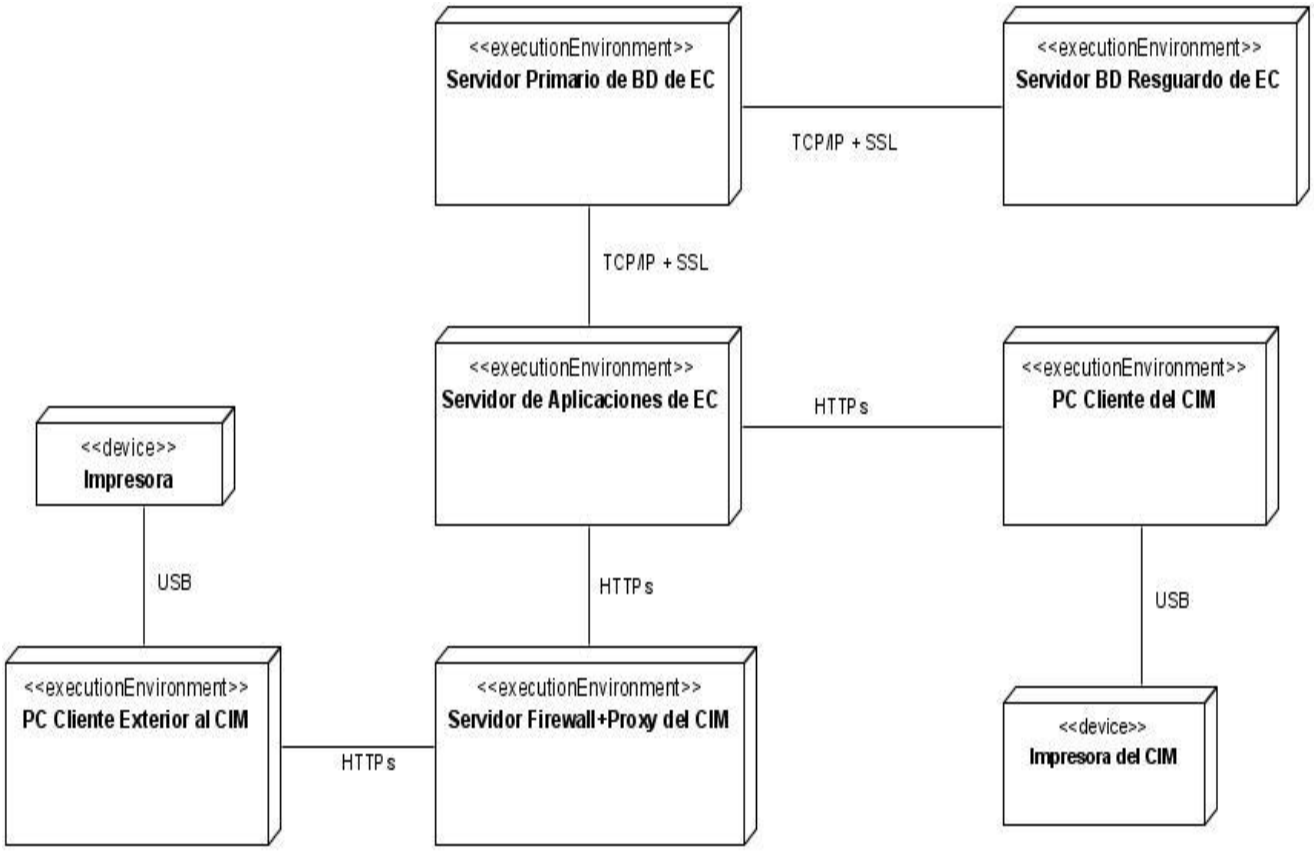


Fig. 28 Modelo de despliegue de la aplicación

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

- En el nodo Servidor de Aplicaciones del EC son ubicadas las capas lógicas Vista, Controlador,

Modelo y Servicios.

- En los nodos PC Cliente del CIM y PC Cliente Exterior al CIM se presentaría el resultado de la capa Vista en cada petición, producido mediante código HTML.

- Dentro de los nodos Servidor Primario de BD del EC y Servidor BD Resguardo de EC estaría la Base de datos y salvas, respectivamente (no forman parte de las capas lógicas, se presentan para una mayor claridad). (29)

2.10 Implementación.

Después de haberse obtenido los prototipos de interfaz de usuario en el flujo de trabajo de Análisis y Diseño de RUP se prosigue con la implementación de los mismos en aras de obtener los correspondientes prototipos ejecutables. Se usó el lenguaje de programación PHP5 y como framework de desarrollo Symfony. La implementación se realizó siguiendo las especificidades descritas en el diseño. A través del código que se muestra a continuación se podrá validar el correcto funcionamiento del diseño realizado.

listarInvestigadoresAction.class.php | controladorPublicador.class.php

```
1  <?php
2
4  abstract class ControladorPublicador
5  {
6      protected $usuarioActivo;
7
8      public function __construct($_usuarioActivo)
12
13     public abstract function getListadoAccionesModelo(PPacModelo $_pacienteModelo);
14
15     public function getListadoInconsistenciasModelo(PPacModelo $_pacienteModelo)
22
23     public function insertarPaciente(PPaciente $_paciente )
29
30     public function insertarDatosPacienteModelo(PPacModelo $_pacienteModelo, Array $_arrDatos, $_estado)
53
54     public function getListadoPacientes( $paginador = 1, ASitio $sitio = null)
69 }
70
71 ?>
```

Fig. 29 Código de la clase ControladorPublicador

La clase presentada anteriormente es una clase abstracta que representa el comportamiento de los objetos de tipo ControladorPublicador. De ella heredan:

- ControladorPublicadorCoordinador
- ControladorPublicadorPromotor
- ControladorPublicadorPrincipal

Estas clases redefinen el método getListadoAccionesModelo, dicho método devuelve un arreglo de las acciones a las que está autorizado el correspondiente controlador sobre los modelos, esto garantiza que una misma página pueda ser visualizada por diferentes roles al mismo tiempo y sin embargo tener diferencias dentro de la gestión de la información. A continuación se mostrará la implementación del método descrito para el rol de Investigador Principal.

```
public function getListadoAccionesModelo(PPacModelo $_pacienteModelo)
{
    $actionList = array();
    $actionList['ver'] = true;

    if ($_pacienteModelo->getEstado() == 'Completo')
    {
        $actionList['editable'] = true;
        $actionList['firmar_on'] = true;
        $actionList['validar'] = true;
    }
    else if ($_pacienteModelo->getEstado() == 'Firmado')
    {
        $actionList['firmar_off'] = true;
        $actionList['validar'] = true;
    }

    return $actionList;
}
```

Fig. 30 Código del método getListadoAccionesModelo de la clase ControladorPublicadorPrincipal

La integración entre los módulos del proyecto se realizó usando una clase fachada por módulo, de la cual solamente puede haber una instancia creada al mismo tiempo, presupone el uso del patrón Singleton. Esta clase fachada servirá como punto de acceso global a cada uno de los módulos, a ella se le pedirán las acciones que se deseen realizar, aislando de esa forma el funcionamiento interno de

los módulos entre ellos. A continuación se muestra un método bien sencillo que demuestra el uso de la clase fachada anteriormente mencionada en este caso la interacción se realiza con el módulo Cronograma.

```
public function insertarPaciente(PPaciente $_paciente )
{

    $_paciente->save();
    FachadaCronograma::getInstancia()->notificarInclusionPaciente( $_paciente );
    FachadaCronograma::getInstancia()->crearCronogramaPaciente($_paciente);
}
```

Fig. 31 Código del Método insertarPaciente de la clase ControladorPublicador

2.11 Conclusiones

En el presente capítulo se realizó una introducción al diseño en términos de conceptos, se realizaron los diagramas de clases del diseño y los correspondientes de interacción a cada caso de uso, donde se modeló la dinámica de la aplicación, teniendo en cuenta la influencia que ejerce el framework utilizado sobre el diseño de la aplicación. Se manifestó el uso de los patrones de diseño presentando fragmentos del código de submódulo funcional.

Conclusiones

En el presente trabajo de diploma se le dio continuidad de forma satisfactoria a la investigación precedente realizada al módulo Publicador, obteniéndose como resultados:

- La selección de estándares a utilizar para el desarrollo de Sistemas de Manejo de Datos de Ensayos Clínicos Cubanos partiendo del estudio de los estándares utilizados por los SIMDEC en el mundo.

- Se obtuvo el diseño del submódulo Gestión de la información de pacientes y de los CRD a partir de los requerimientos identificados en la investigación precedente. El diseño fue realizado teniendo en cuenta la arquitectura Modelo Vista Controlador, propuesta por el framework Symfony.

- Se obtuvo la implementación del submódulo Gestión de la información de pacientes y de los CRD, logrando la integración con el módulo de Administración y el submódulo del módulo de Diseño llamado Cronograma. Dándose respuesta a los requerimientos funcionales correspondientes a los procesos de gestión de la información de pacientes y de los CRD identificados en la investigación precedente.

Recomendaciones

Se recomienda elaborar un manual de usuario del producto para lograr un mejor entendimiento del funcionamiento del mismo. Realizar la implementación del módulo Monitoreo y Validación del proyecto que se encuentran en la actualidad en el flujo de trabajo de diseño, con el fin de lograr una integración completa y poder entregar en el plazo definido el SIMDECC.

Referencias Bibliográficas

1. Med line Plus. [Online] [Cited: 11 13, 2007.] http://www.nlm.nih.gov/medlineplus/spanish/clinicaltrials.html

2. Revista cubana de Informática Médica. [Online] [Cited: 11 13, 2007.] http://www.cecam.sld.cu/rcim/revista_4/articulos_html

3. PIVOTAL. [Online] [Cited: 11 14, 2007.] http://www.pivotal.es/extranet.

4. Hypocrates. [Online] [Cited: 11 16, 2007.] http://hipocrates.com

5. OpenClínica. [Online] 12 18, 2007. [Cited: 12 29, 2007.] http://www.openclinica.org

6. Rational. [Online] 6 7, 2004. [Cited: 1 5, 2008.] http://www.rational.com

7. Craig, Larman. UML y Patrones. La Habana : Editorial Félix Varela, 2004. pag 15

8. Unified Modeling Language. [Online] [Cited: 1 6, 2008.] http://www.uml.org

9. Ciberaula. [Online] [Cited: 1 15, 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro/

10 Postgres [Online] [Cited: 1 17, 2008.].http://www.freegestor.org/docs/postgres/x15.html/

11. PostgresSQLPE. [Online] [Cited: 1 20, 2008.] http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf

12. Ciberaula.[Online] 2006. [Cited: 2 4, 2008.] http://www.ciberaula.com/curso/php5/que_es/.

13. PHP.net. [Online] 7 9, 2007. [Cited: 2 5, 2008.] http://www.php.net/manual/es/migration5.php

14. Visual Paradigm para UML. [Online] 3 5, 2007. [Cited: 2 7, 2008.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/

15. Zend. [Online] [Cited: 2 8, 2008.] http://framework.zend.com.

16. Potencier, Fabien y Zaninotto, Francois. Definitive Guie to Symfony.

.17 NuspherephpED [Online] [Cited: 2 8, 2008.] http://www.nusphere.com/products/phped.htm

18. Eclipse. [Online] 10 27, 2007. [Cited: 2 9, 2008.] http://www.eclipse.org/ .

19. Kompozer. [Online] [Cited: 2 10, 2008.] http://kompozer.net/features.php.

20. Colling-Sussman, Ben, W.Fitzpatrick, Brian y Pilato, C.Michael. Control de versiones con

Subnersion . [Online] [Cited: 2 10, 2008.] http://svnbook.red-bean.com/nightly/es/index.html.

21. Craig, Larman. UML y Patrones. La Habana : Editorial Félix Varela, 2004, pag 6

22. AdictosalTrabajo. [Online] 12 22, 2003. [Cited: 2 23, 2008.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

23. AdictosalTrabajo. [Online] 12 22, 2003. [Cited: 2 23, 2008.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?singleton=singleton

24. Mundojava. [Online] 6 1, 2007. [Cited: 2 23, 2008.] http://mundojava.blogspot.com/2007/01/el-patrn-decorator-y-una-aplicacin-real.html

25.Programacion. [Online] [Cited: 2 24.2008.] ttp://www.programacion.net/java/articulo/joa_patrones4/#joa_patrones3_fachada

26. Introduccion a Patrones. [Online] [Cited: 2 25, 2008.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

27. Orfeo. [Online] 7 13, 2007. [Cited: 2 25, 2008.] http://www.orfeogpl.org/?q=node/242.

28. López Días, Aidaselys and Rodríguez García, Lucia. Sistema de Manejo de Datos de Ensayos Clinicos.

29. Ballester Marsal, Andres. Documenteo de Arquitectura de Software (Ensayos clínicos). Ciudad de la Habana : s.n., 2008.

Bibliografía

1. B. Bruegge, A. Dutoit. "Object-Oriented Software Engineering". Prentice-Hall, 2000.
2. C. Larman. "UML y Patrones". 2ª ed. Prentice-Hall, 2002.
3. Design Patterns. Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Group of Four)
4. E. Gamma. "Patrones de Diseño. Elementos de software orientado al objeto reutilizable". Addison-Wesley, 2002
5. G.Booch; J. Rumbaugh & I. Jacobson. "El lenguaje unificado de modelado". La Habana 2004
6. I. Jacobson, G.Booch & J. Rumbaugh. "El Proceso Unificado de Desarrollo del Software". La Habana .2004.
7. I. Sommerville. "Ingeniería del Software". 6ª ed. Addison-Wesley, 2002
8. Larman, C. UML y Patrones. La Habana 2004
9. M. Page-Jones. "Fundamentals of Object-Oriented Design in UML". Addison-Wesley, 2000
10. R.S. Pressman. "Ingeniería del Software. Un enfoque práctico".

Anexos

Anexo 1.

A continuación se presentan las descripciones textuales de los Caso de Uso del Sistema.

Caso de Uso:	Insertar_Paciente_Ensayo
Actores:	Coordinador_Investigación_Clínica (Inicia)
Propósito:	Permitir Insertar un paciente en el ensayo.
Resumen:	El caso de uso inicia cuando el Coordinador de la investigación clínica decide insertar un paciente, para ello introduce el nombre y el número de la historia clínica además de llenar el modelo de inclusión, el caso de uso termina cuando el paciente es insertado y se vuelve al listado de pacientes.
Referencia:	R1, R5, R11
Pre condiciones:	
Post condiciones:	El paciente queda insertado y se muestra en el listado de pacientes.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- El Coordinador selecciona insertar paciente	1.1.- El sistema muestra un formulario para la introducción de las iniciales y el número de historia clínica.
2.- El Coordinador introduce el número de historia clínica y las iniciales del paciente.	2.1-El sistema verifica que los datos sean correctos 2.2-El sistema muestra el modelo de inclusión.
3.- El Coordinador inserta los datos en el modelo de inclusión.	3.1.- El sistema verifica que botón fue pulsado 3.2.-Si el botón pulsado fue guardar o guardar y completar. 3.3.-El sistema a través de una instancia de la

	clase controladora del rol activo inserta el paciente y notifica su inclusión al módulo de cronograma para que se le genere el correspondiente cronograma de ejecución. 3.4 El sistema inserta los datos del modelo de inclusión y notifica al módulo de cronograma el llenado de un nuevo modelo además de verificar en el listado de acciones si se puede validar el modelo para notificarle al módulo de Validación que prosiga a validarModelo para que se guarden las inconsistencias generadas. 3.4.-El sistema muestra el listado de paciente con el nuevo paciente incluido finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	2.2.- Si los datos no fueron bien incluidos se Redirecciona a la misma página con un mensaje de error.
Acción 3.1	3.2 Si el botón pulsado fue cancelar se redirecciona al listado de pacientes. Finaliza el caso de uso.
Prioridad:	Crítico

Caso de Uso:	Insertar_Datos_Modelos_Paciente
Actores:	Coordinador_Investigación_Clínica (Inicia)
Propósito:	Permitir Insertar datos de un modelo para un paciente.

Resumen:	El caso de uso inicia cuando el Coordinador de la investigación clínica decide insertar datos en un modelo de un paciente del ensayo. Finalizando el caso de uso.
Referencia:	**R5, R6,R7, R8, R10**
Precondiciones:	
Poscondiciones:	Los datos del modelo quedan actualizados.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- El Coordinador selecciona un paciente de la "Lista de pacientes" del ensayo, para registrar los datos en uno de sus modelos.	1.1.- El sistema interactúa con la fachada de Cronograma y obtiene el listado de los modelos sintomáticos y asintomáticos pertenecientes al paciente 1.2.-El sistema interactúa con la fachada de de Validación para obtener el listado de inconsistencias de los modelos. 1.3.-El sistema muestra el Listado de Visitas.
2.- El Coordinador selecciona un modelo de la "Lista de Vistas", del paciente para registrar los datos.	2.1.-El sistema muestra el modelo seleccionado según las acciones que puede realizar sobre él, el rol autenticado.
3.- El Coordinador inserta los datos en el modelo.	3.1.- El sistema verifica que botón fue pulsado. 3.2.- Si el botón pulsado fue guardar o guardar y completar. El sistema guarda los datos y notifica al módulo de Cronograma el llenado del modelo y al módulo de validación para que obtemga las correspondientes inconsistencias. 3.3- El sistema muestra el listado de visitas y

	finaliza así es caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 3.1	3.2-Si el botón pulsado fue cancelar se retorna al Listado de Visitas.
Prioridad:	Crítico

Caso de Uso:	Visualizar_Modelo
Actores:	Visualizador (Inicia)
Propósito:	Visualizar un modelo seleccionado.
Resumen:	El caso de uso comienza cuando el Coordinador de la investigación clínica, Investigador promotor, Investigador principal decide ver un modelo de un paciente particular.
Referencia:	**R5, R6, R8**
Precondiciones:	
Poscondiciones:	

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1.- El Visualizador selecciona un paciente de la “Lista de pacientes” del ensayo.	1.1.- El sistema interactúa con la fachada de Cronograma y obtiene el listado de los modelos sintomáticos y asintomáticos pertenecientes al paciente 1.2.-El sistema interactúa con la fachada de de Validación para obtener el listado de inconsistencias de los modelos. 1.3.-El sistema muestra el Listado de Visitas.

2.- El visualizador selecciona de la “Lista de Vistas” el modelo que desea visualizar.	2.1.- El sistema obtiene el arreglo de subvariables a visualizar. 2.2.-Visualiza el modelo en dadas las acciones que el rol autenticado puede realizar sobre él ,mostrando las inconsistencias existentes en el modelo seleccionado, obtenidas mediante el módulo de Validación, finalizando así el caso de uso
Prioridad:	Crítico

Caso de Uso:	Modificar_Datos_Modelos
Actores:	Especialista (Inicia)
Propósito:	Permitir modificar datos de un modelo para un paciente.
Resumen:	El caso de uso inicia cuando el Investigador principal o el coordinador de la investigación clínica deciden modificar datos de un modelo de un determinado paciente del ensayo.
Referencia:	**R5, R6, R8, R9, R10**
Precondiciones:	
Poscondiciones:	Los datos del modelo quedan actualizados.

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1.- El Especialista selecciona un paciente de la “Lista de paciente” del ensayo, para modificar los datos en un modelo.	1.1.- El sistema interactúa con la fachada de Cronograma y obtiene el listado de los modelos sintomáticos y asintomáticos pertenecientes al paciente

	1.2.-El sistema interactúa con la fachada de de Validación para obtener el listado de inconsistencias de los modelos. 1.3.-El sistema muestra el Listado de Visitas.
2.- El Especialista selecciona un modelo de la “Lista de Vistas”, del paciente para modificar los datos.	2.1.- El sistema obtiene el arreglo de subvariables a visualizar. 2.2.-Visualiza el modelo en dadas las acciones que el rol autenticado puede realizar sobre él, mostrando las inconsistencias existentes obtenidas a través del módulo de Validación.
3.- El especialista modifica los datos en el modelo.	3.1.- El sistema verifica que botón fue pulsado. 3.2.- Si el botón pulsado fue guardar, guardar y completar o firmar. El sistema actualiza la base de datos con los nuevos datos introducidos 3.3- El sistema muestra el listado de visitas y finaliza así es caso de uso.
Flujo Alterno	
Acción del Actor	Respuesta del Sistema
Flujos Alternos de botón pulsado	
Acción del Actor	Respuesta del Sistema
Acción 3.1	3.2.- Si el botón pulsado fue cancelar el sistema redirecciona al Listado de Visitas finalizando el caso de uso.
Acción 3.1	3.2.- Si botón pulsado fue Quitar Firma se le restituye al modelo el estado de completo. Y se redirecciona al Listado de Visitas finalizando así el caso de uso.

Prioridad:	Crítico

Caso de Uso:	Visualizar_Cronograma
Actores:	Visualizador (Inicia)
Propósito:	Conocer la programación de la visitas del ensayo de forma general o para un paciente particular.
Resumen:	El caso de uso comienza cuando el visualizador decide ver el cronograma particular de un determinado paciente, para conocer la fecha exacta en las que tendrán lugar las visitas programadas para dicho paciente.
Referencia:	**R6, R12**
Precondiciones:	
Poscondiciones:	

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1.- El visualizador decide Visualizar el cronograma del paciente seleccionado.	1.1.- El sistema interactúa con la fachada de cronograma y obtiene el listado de los modelos sintomáticos y asintomáticos del paciente 1.2 El sistema muestra el cronograma del paciente finalizando el caso de uso.
Prioridad:	Crítico

Caso de Uso:	Visualizar_Investigadores
Actores:	Investigador_Promotor (Inicia)

Propósito:	Conocer los especialistas encargados de la realización de un ensayo clínico en un sitio dado.
Resumen:	El caso de uso comienza cuando el Investigador_Promotor decide ver el Listado de Investigadores.
Referencia:	**R2, R3, R4**
Precondiciones:	
Poscondiciones:	

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1.- El Investigador_Promotor escoge el sitio del que quiere visualizar el listado de investigadores	1.1.- El sistema muestra el listado de los sitios obtenido a través del id del usuario activo.
2.-El Investigador_Promotor escoge el sitio del que quiere visualizar los investigadores.	2.1 El sistema muestra el listado de los investigadores del sitio seleccionado partiendo del id del sitio.

Prioridad:	Crítico

Anexo 2

Requisitos no funcionales de la aplicación.

1- RNF de Apariencia o Interfaz externa

Las páginas no tendrán muchas imágenes y poseerán pocos colores. Cada rol tendrá una interfaz diferente con las funciones que le corresponden. Se hará uso de simbología mediante iconos para indicar el estado de los elementos utilizados en el diseño. Además, los iconos contendrán funcionalidades específicas.

2- RNF de Usabilidad

Las personas que interactuarán con el software serán médicos 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.

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 del sistema 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 rol, usuario y contraseña. Las contraseñas deberán tener más de 7 caracteres de longitud y tener una fortaleza media. Los usuarios estarán obligados a cambiar la contraseña cada 60 días como máximo. Cada usuario tendrá asignado uno o varios roles en el sistema. Cada rol definido tendrá niveles de acceso al Software. Solo podrán acceder a la aplicación, clientes a través de direcciones IP específicas bien controladas. Todo cambio o modificación en el sistema debe ser atribuible a un usuario particular según su autenticación. Paralelo a la base de datos primaria se debe mantener una base de datos que registre todas las modificaciones hechas a la base de datos original, ordenadas cronológicamente y con la especificación del usuario responsable de dicha modificación, de manera que siempre se realicen trazas a la información manejada. Se debe garantizar comunicaciones seguras entre los

clientes y el servidor, encriptando todo el tráfico de información usando llaves negociadas, algoritmos y protocolos.

5- RNF de Software

El servidor de la aplicación se encontrará en una computadora que tenga instalado el sistema operativo GNU Linux. En las computadoras de los clientes se podrá tener instalado cualquier Sistema Operativo siempre y cuando incluya un navegador de última generación para la visualización de las interfases 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 por al menos un procesador Dual Intel Xeon 3 GHz o similar y RAM suficiente (4 GB a 8 GB). Los servidores de almacenamiento de datos deben tener de 1 a 3 TB disponibles pues el volumen de información es bastante grande y perdura en el tiempo hasta 15 años.

7- RNF de Restricciones en el Diseño y la Implementación

El análisis y diseño de la aplicación será basado en la Metodología RUP haciendo uso del lenguaje de modelado UML. Se usará como herramienta CASE Visual Paradigm Suite 3.1 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 interfases se utilizará Kompozer. Se usará como lenguaje de programación PHP5. Se usará como Gestor de Base de Datos Postgre-SQL

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.

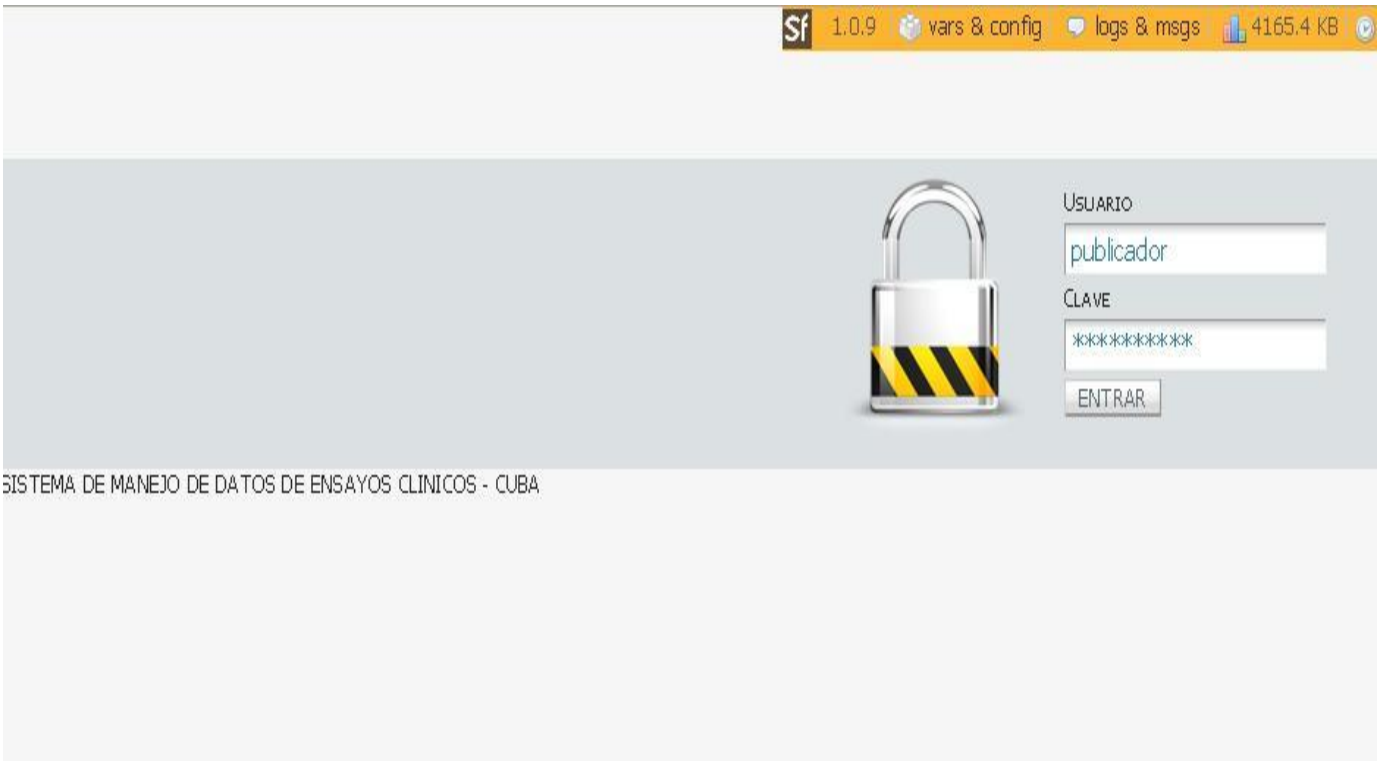
9- RNF de Disponibilidad

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

.

Anexo 3

Prototipos de interfaz de usuario de la aplicación.



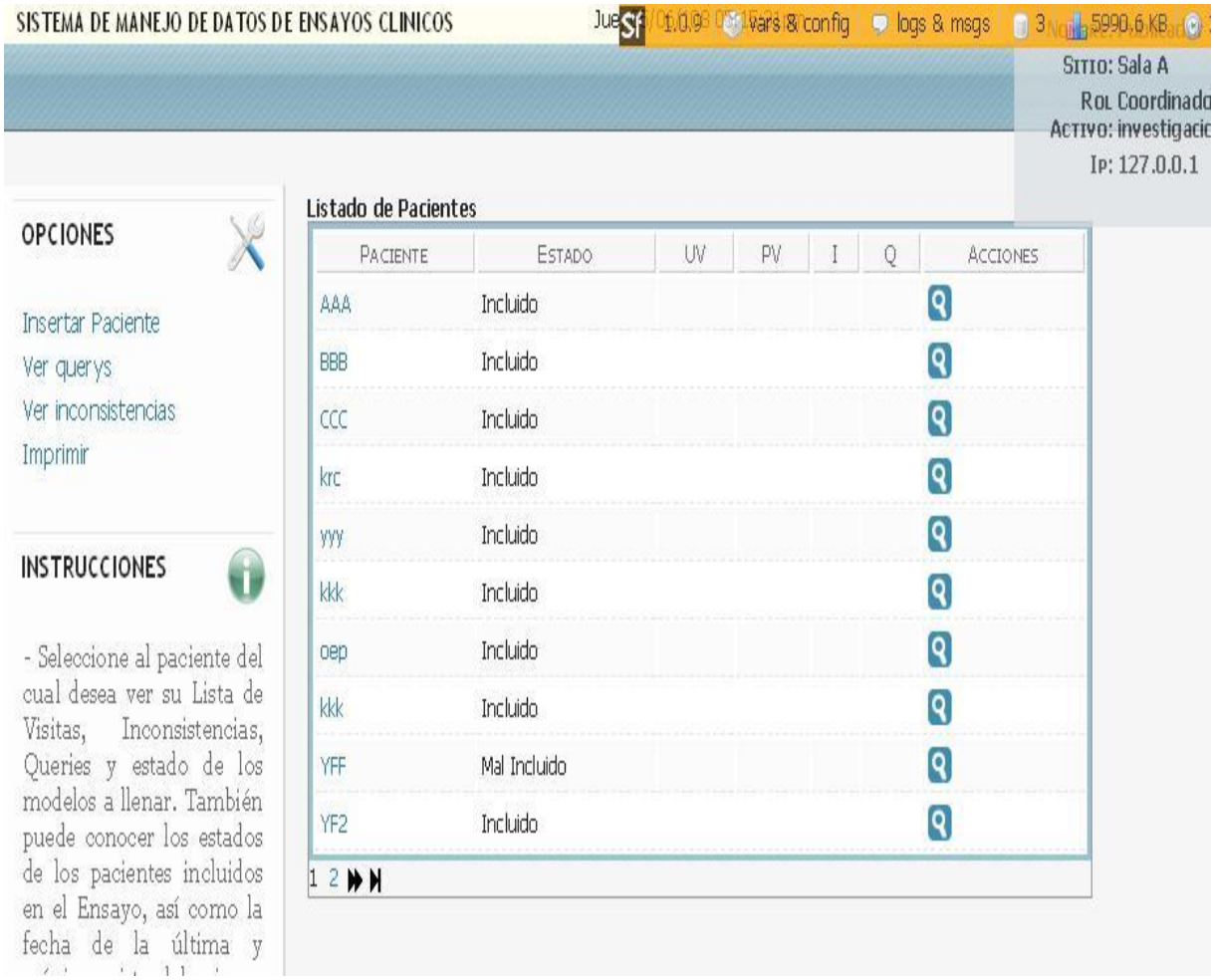
Prototipo de interfaz de usuario: Autenticación

Prototipo de interfaz de usuario: Escogiendo rol

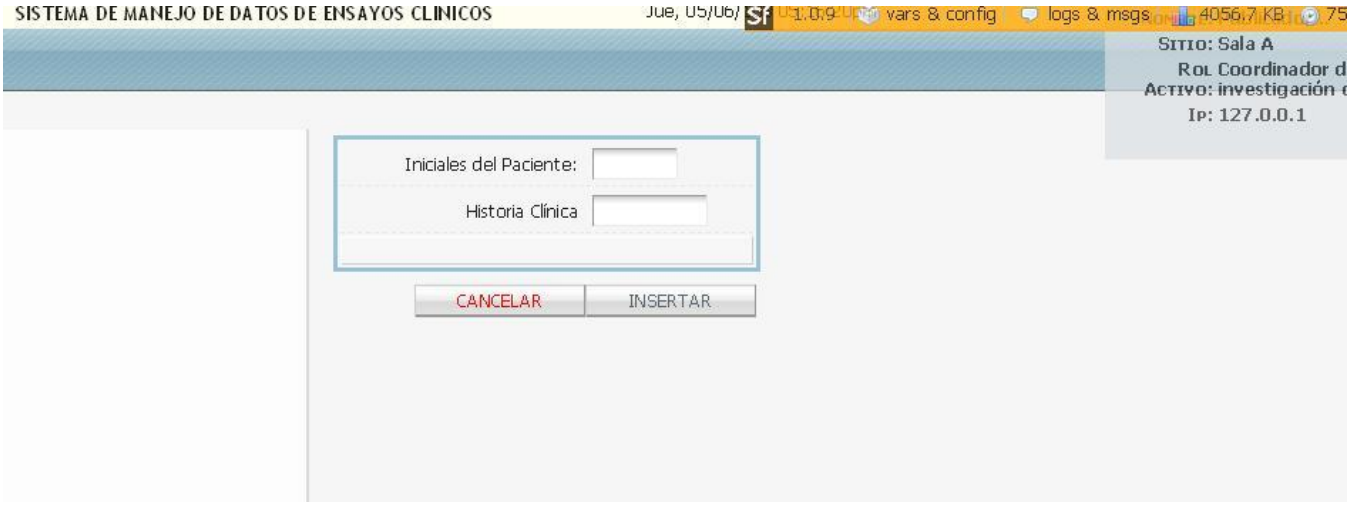
Prototipo de interfaz de usuario: Listado de hospitales

Prototipo de interfaz de usuario: Listado de sitios pertenecientes a un hospital

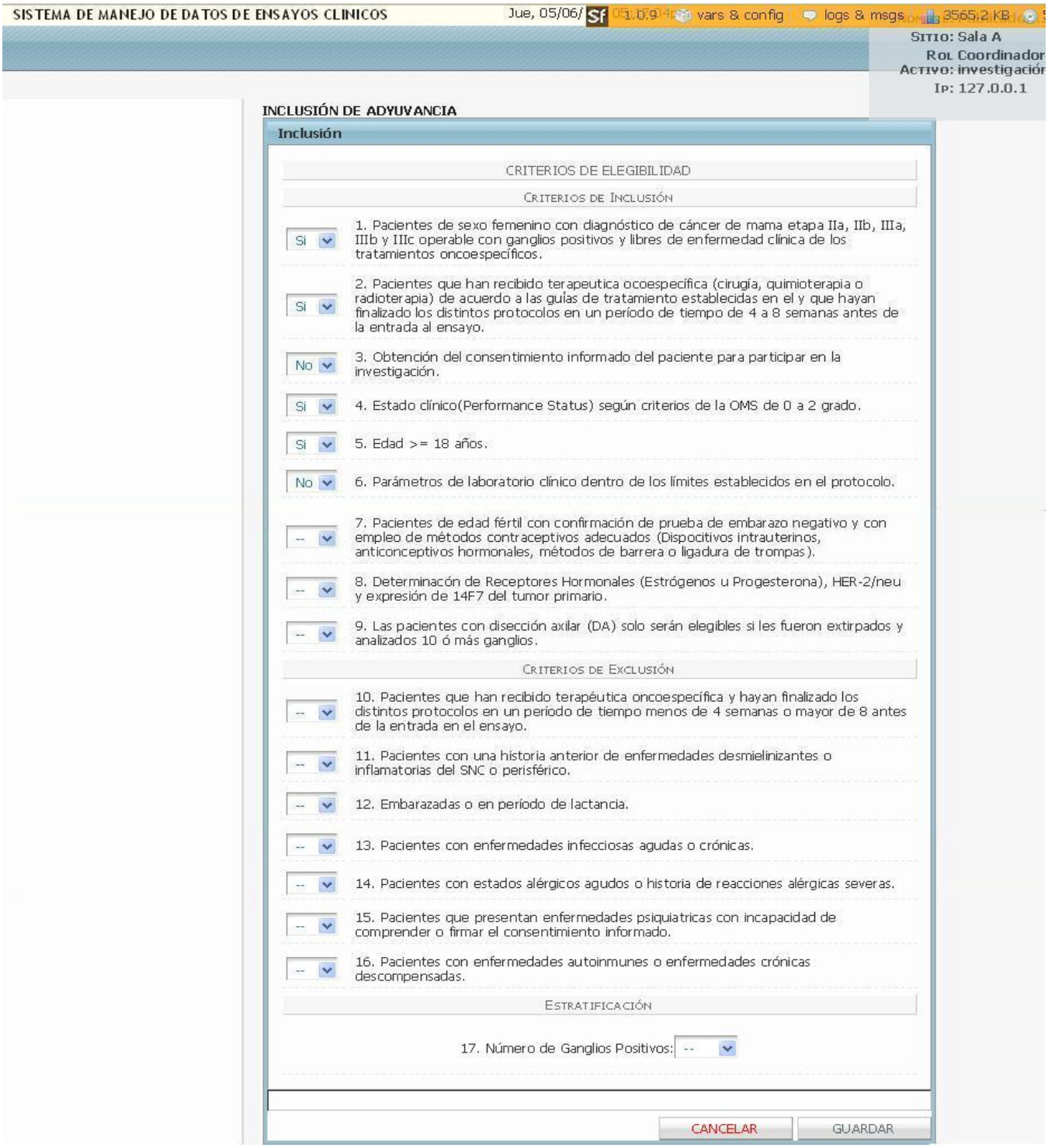
Prototipo de interfaz de usuario: Listado de investigadores de un sitio



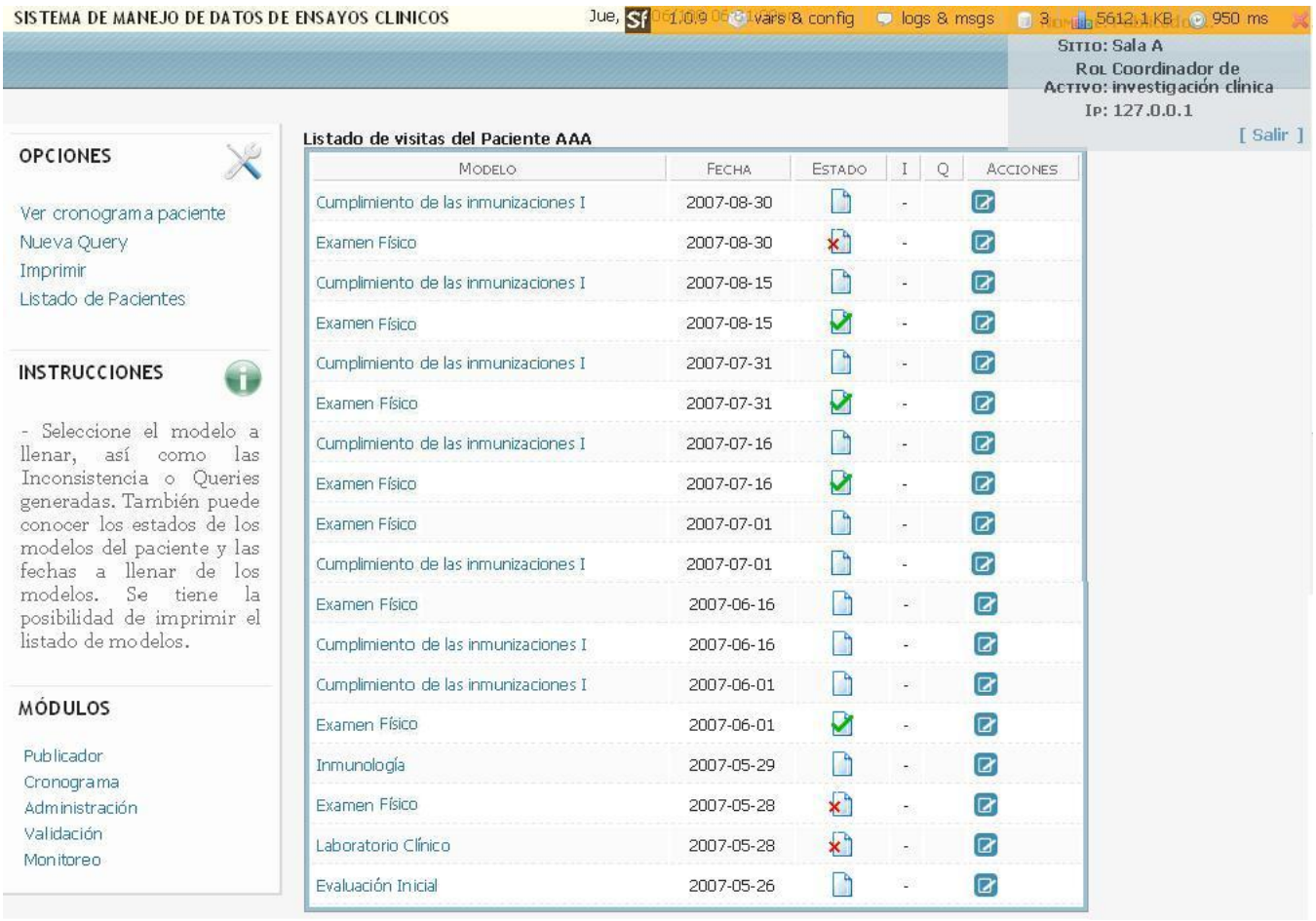
Prototipo de interfaz de usuario: Listado de pacientes de un sitio



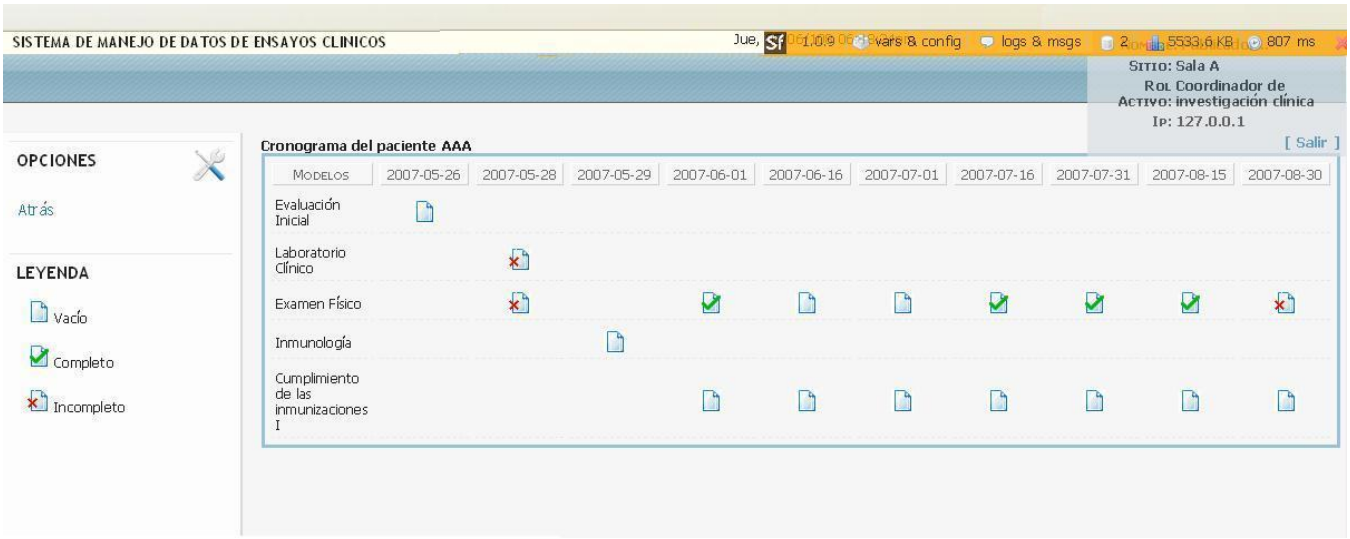
Prototipo de interfaz de usuario: Insertar paciente



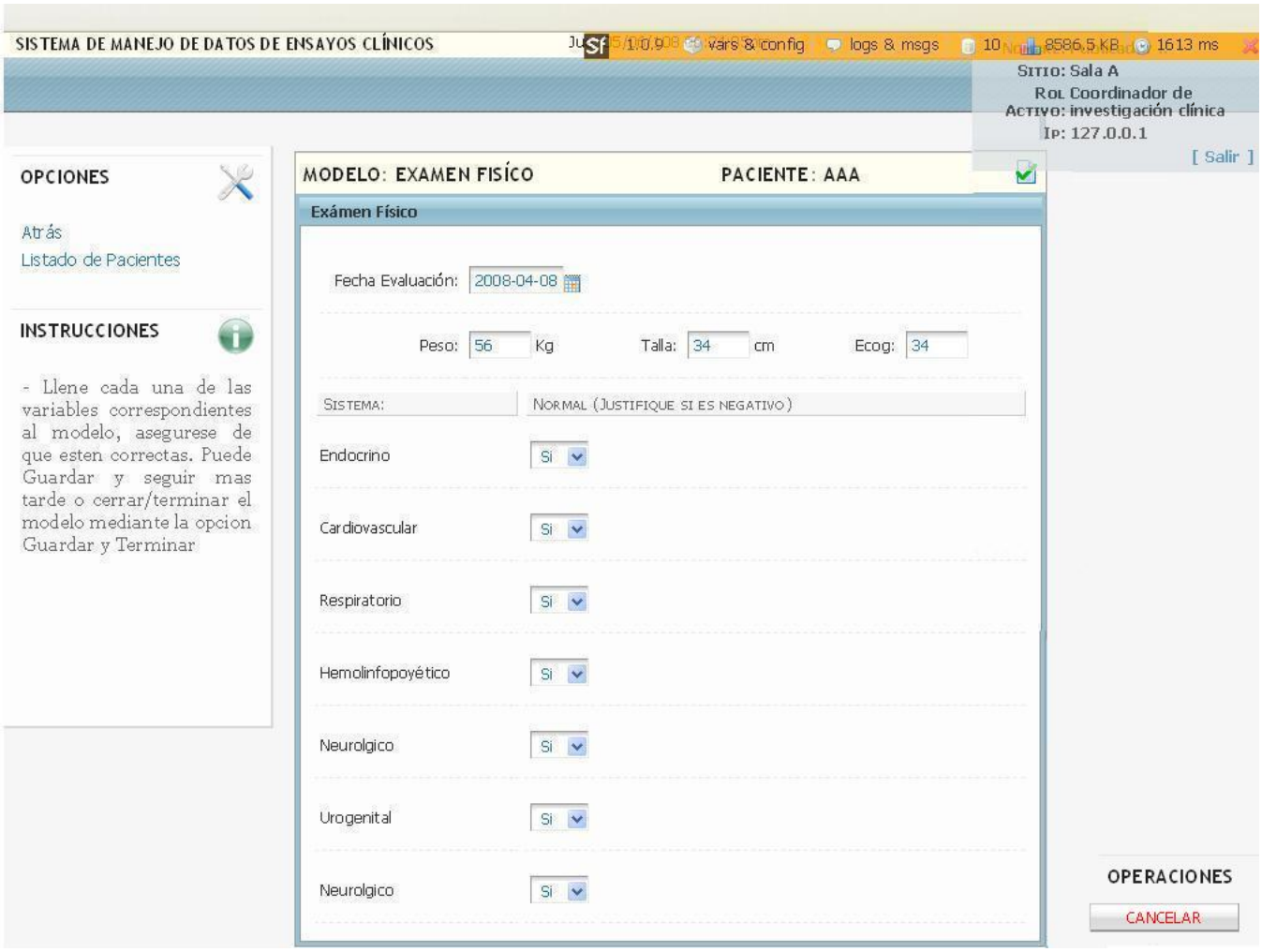
Prototipo de interfaz de usuario: Modelo de inclusión de un paciente



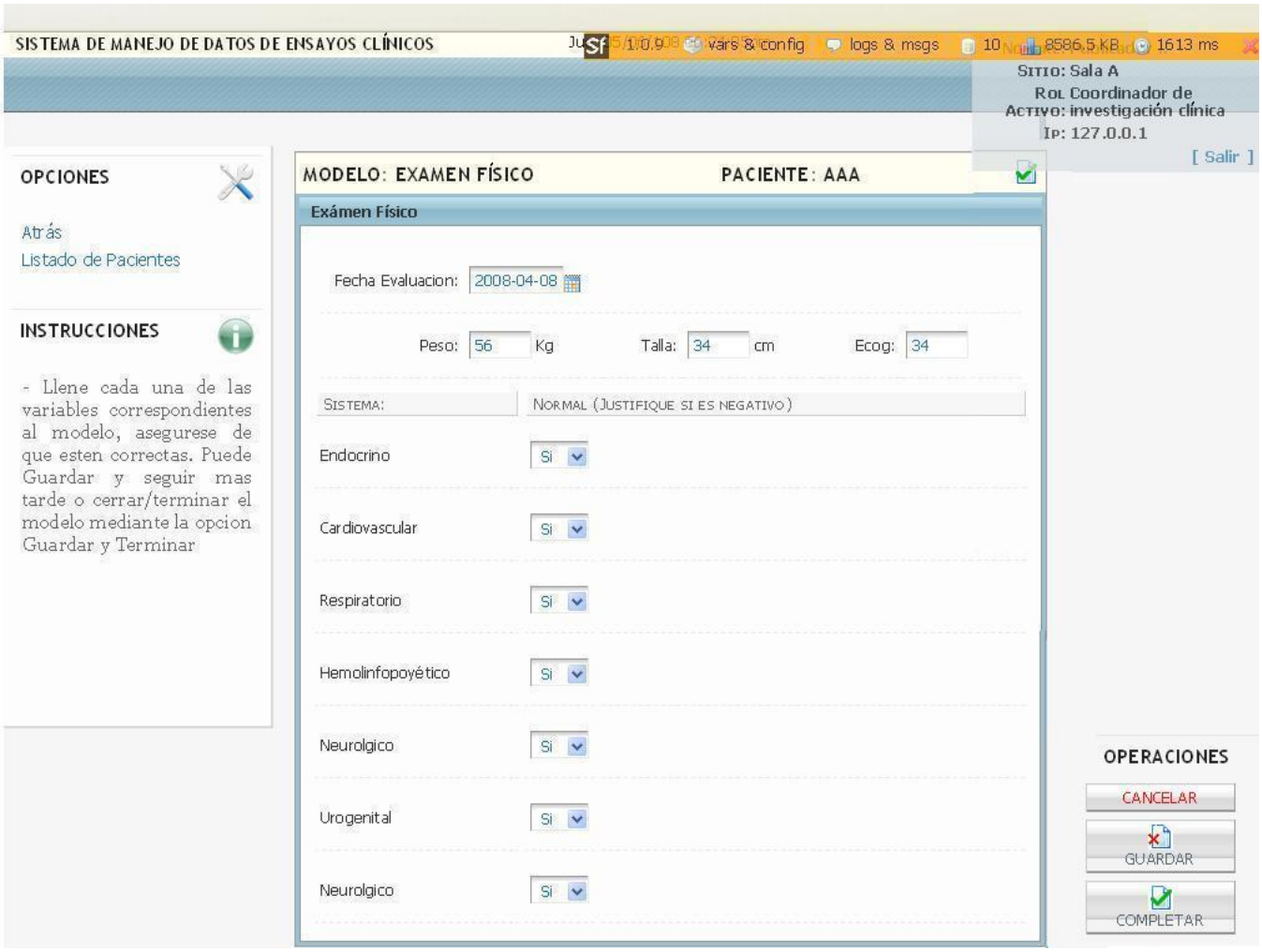
Prototipo de interfaz de usuario: Listado de visitas de un paciente



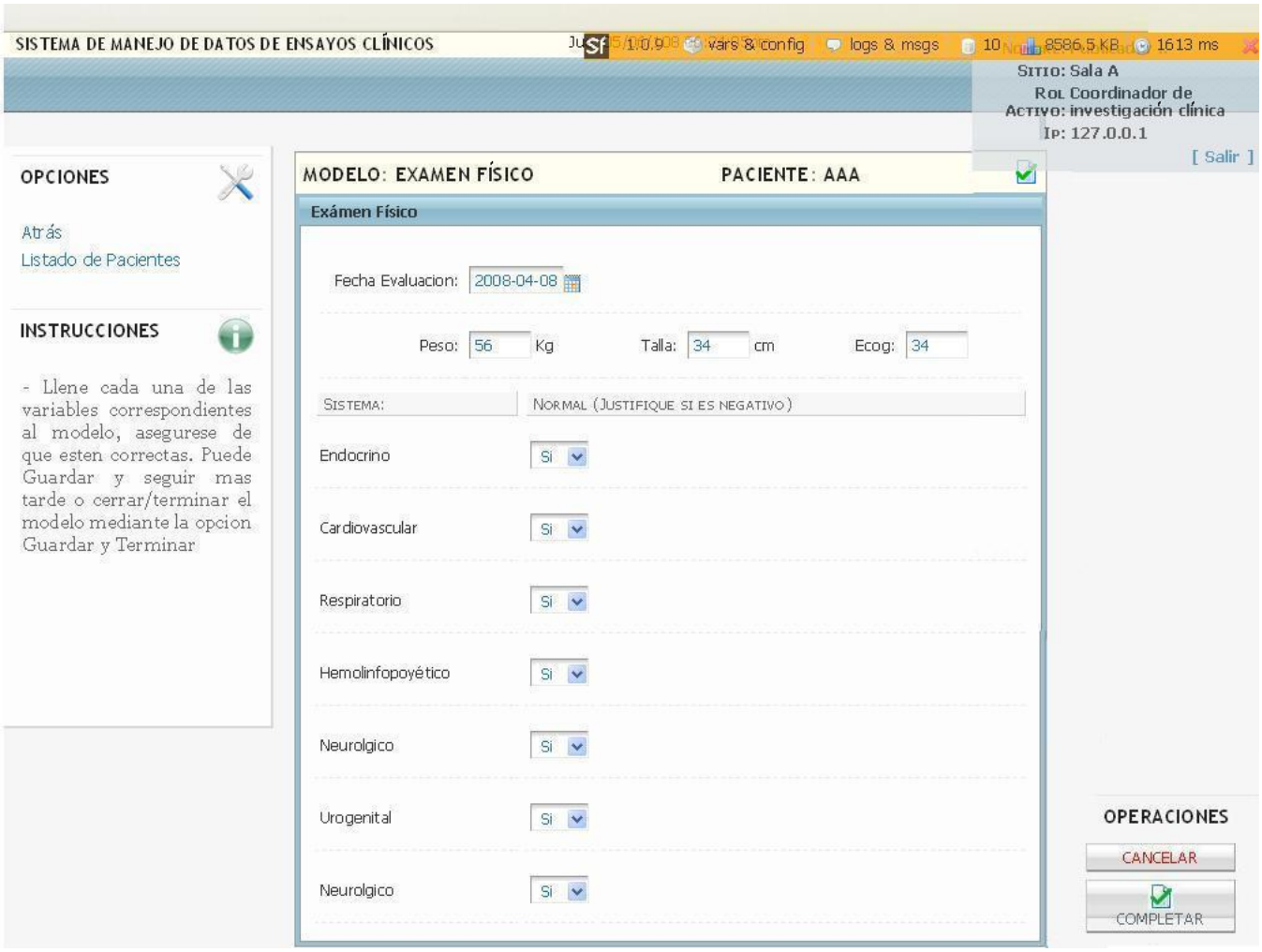
Prototipo de interfaz de usuario: Cronograma específico de un paciente



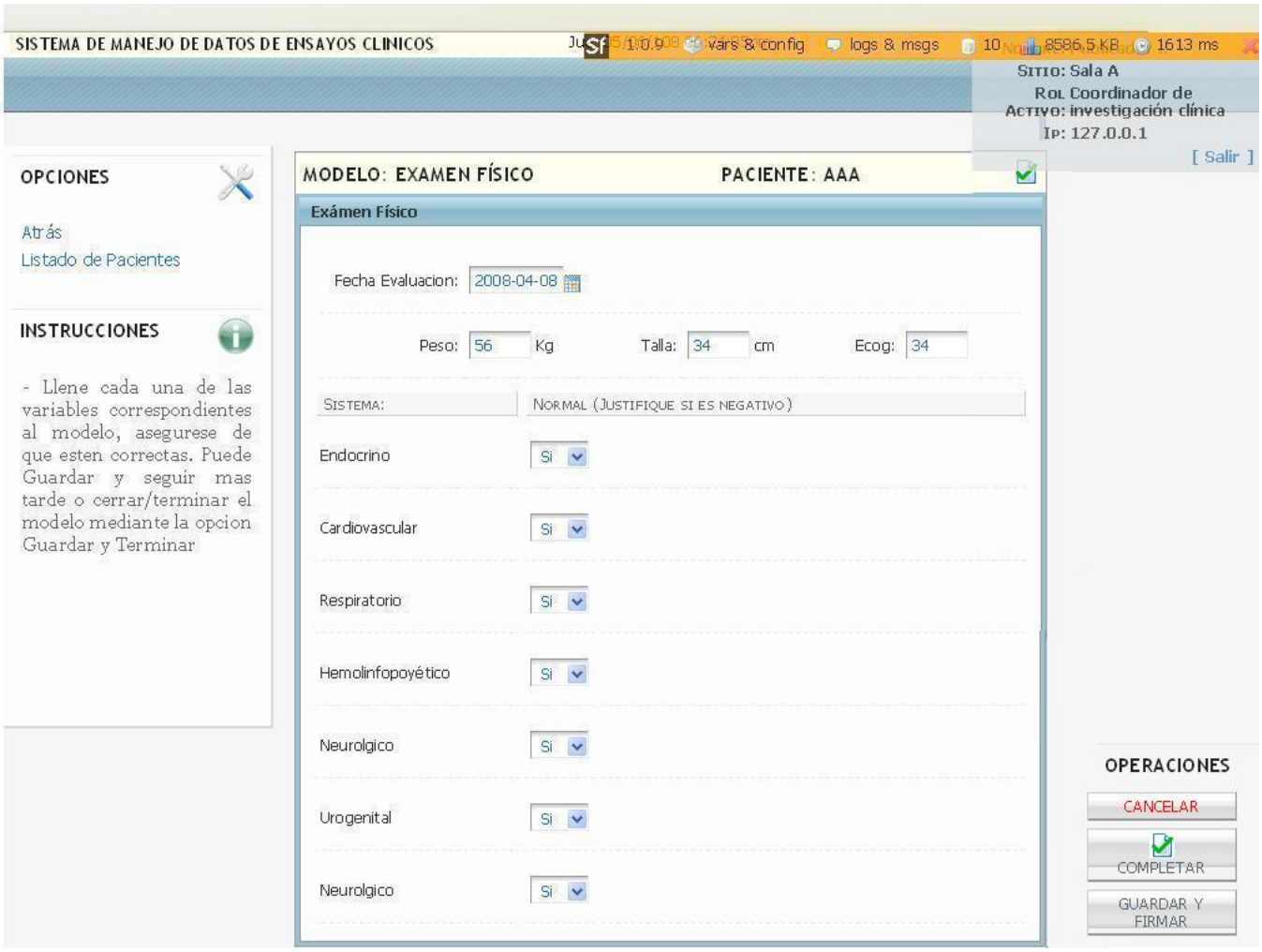
Prototipo de interfaz de usuario: Visualizar modelo



Prototipo de interfaz de usuario: Insertar datos de un modelo



Prototipo de interfaz de usuario: Modificar datos de un modelo



Prototipo de interfaz de usuario: Firmar modelo

Glosario

Anticuerpos monoclonales: son anticuerpos idénticos porque son producidos por un solo tipo de célula del sistema inmune, es decir, todos los clones proceden de una sola célula madre. Es posible producir anticuerpos monoclonales que se unan específicamente con cualquier molécula con carácter antigénico. Este fenómeno es de gran utilidad en bioquímica, biología molecular y medicina.

Centro biotecnológico: es una institución de desarrollo dinámico que le ha permitido alcanzar un alto nivel en la investigación, desarrollo, producción y comercialización de productos biológicos obtenidos a través de los métodos de la biotecnología moderna.

Cuaderno de Recogida de Datos: es un documento impreso, óptico ó electrónico diseñado para recoger toda la información que requiere un protocolo de investigación

Ensayo Clínico: proceso de aplicación de productos en prueba a los humanos junto a la recuperación de todos los datos de la evolución del medicamento en ellos.

Fármacos: término farmacológico para cualquier compuesto biológicamente activo, capaz de modificar el metabolismo de las células sobre las que hace efecto.

Framework: estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

Herramientas case: son 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.

Inmunología: es una rama amplia de la biología y de las ciencias biomédicas que se ocupa del estudio del sistema inmunitario en todos los organismos, entendiendo como tal al conjunto de órganos, tejidos y células que en los vertebrados tienen como función biológica el reconocer elementos extraños o ajenos dando una respuesta (respuesta inmunológica).

Inmunoterapia: es un tipo de tratamiento que se basa en la estimulación del propio sistema inmunológico del paciente para que sea capaz de reconocer y eliminar las células tumorales.

Queries: Constituyen una especie de consulta entre el Monitor y el Coordinador de la investigación clínica o el Investigador principal, su objetivo es comunicar y establecer un debate sobre alguna deficiencia encontrada relacionada con los pacientes, los modelos o el ensayo en general.

Release: Versiones que se generan de la aplicación tan pequeñas como sea posible como resultado de una metodología, pero que proporcionen un valor adicional claro, desde el punto de vista del negocio.