Universidad de las Ciencias Informáticas

Facultad 6

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

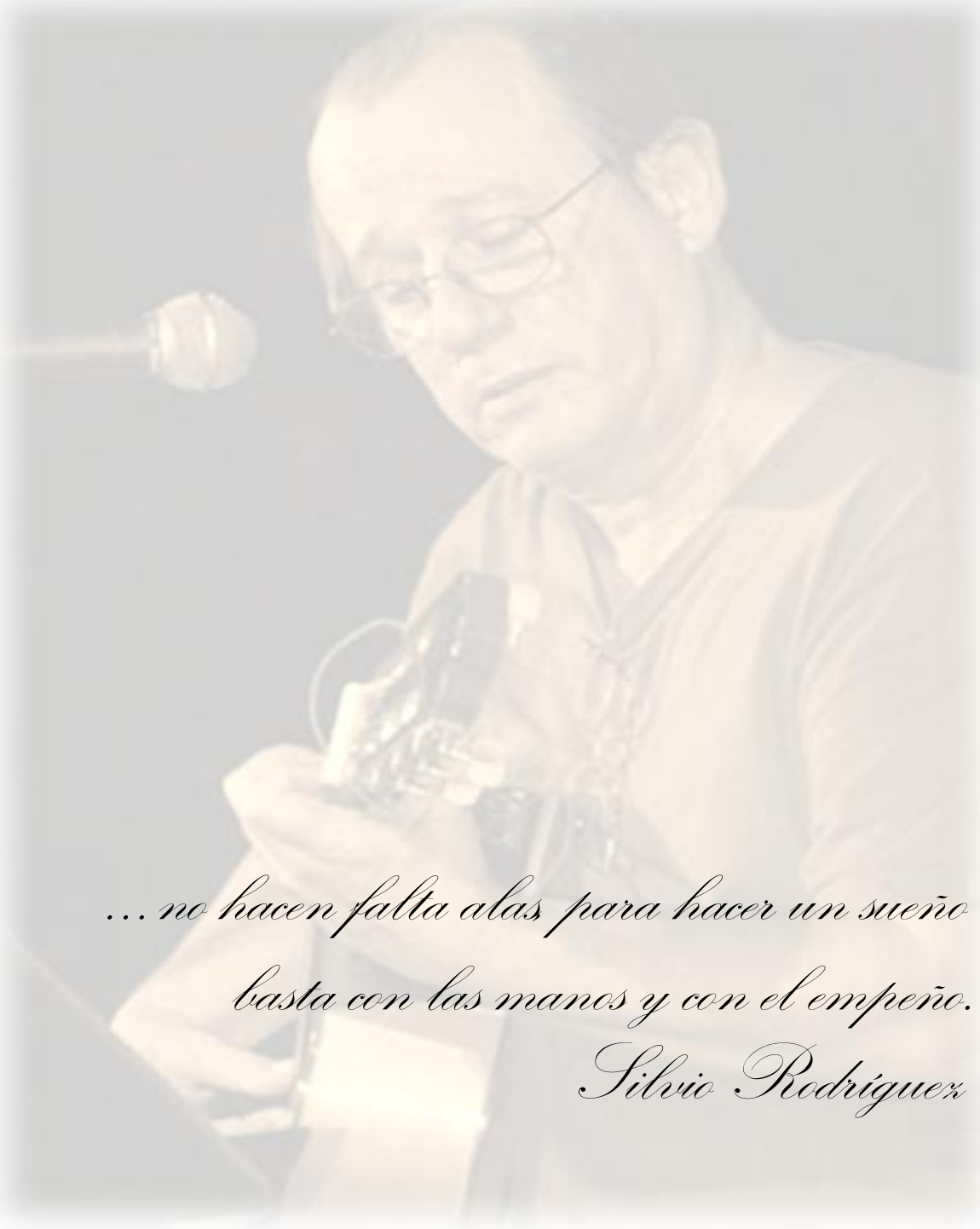
Título: *alasClínicas: Desarrollo de la Gestión de Datos de Ensayos Clínicos a partir del sistema OpenClinica.*

Autores: Erislán Martínez Jera
Landy González Enríquez

Tutores: Ing. Lucía Rodríguez García
Ing. Andrés Ballester Marsal

Cotutora: Ing. Martha D. Hernández Ramírez

Ciudad de La Habana, Junio 2009
"Año del 50 aniversario del triunfo de la Revolución"



... no hacen falta alas para hacer un sueño

basta con las manos y con el empeño.

Silvio Rodríguez

DECLARACIÓN DE AUTORÍA

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

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

Autores:

Erislán Martinez Jera Landy González Enríquez

_____ _____

Firma del Autor Firma del Autor

Tutores:

Ing. Lucía Rodríguez García Ing. Andrés Ballester Marsal

_____ _____

Firma de la Tutora Firma del Tutor

Cotutora:

Ing. Martha D. Hernández Ramírez

_____

Firma de la Cotutora

DATOS DE CONTACTO

Ing. Lucía Rodríguez García

Correo electrónico: lrodriguezg@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Ing. Andrés Ballester Marsal

Correo electrónico: aballester@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Ing. Martha D. Hernández Ramírez

Correo electrónico: mdhernandez@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Agradecimientos

Erislán Martinez Jera

Quiero agradecerle a toda mi familia por apoyarme siempre y por brindarme su guía.

A mi tío por ser una de las personas que más admiro.

A mis abuelos por brindarme su apoyo incondicional cada vez que lo necesité, por confiar siempre en mí y en mis decisiones, por sus buenos y siempre oportunos consejos.

A mi mamá por ser la luz que me mostró el camino, por ser tan exigente y cada vez exigirme más, demostrándome que siempre se puede dar más. Sin ella no hubiera llegado hasta aquí. Por ser más que mi madre, mí amiga, por darme y enseñarme la vida.

A todos mis maestros y profesores por aportar cada uno un granito de arena en mi enseñanza. De cada uno conservo un buen ejemplo que se que me ayudará a ser un buen profesional.

A todos los profes del proyecto.

A mis tutores Lucia y Ballester por ser los mejores tutores del mundo.

A todos mi amigos y compañeros, a los viejos y nuevos conocidos. Cada uno de ellos formó un hito en mi vida y en mi carrera.

A los de la UCI, por estos 5 años inolvidables, por permitir llevarme de cada uno una anécdota y muchos recuerdos.

A mi compañero de tesis Landy, por ser tan optimista, a todos los de mi grupo 2, los del 4 y también a mis amigos del IPVCE.

A Lexy por ser mi inspiración, un motivo por el cual esforzarme cada día más, un ejemplo a seguir y por hacer más alegres cada uno de los días vividos en esta escuela.

A todos muchas gracias.

A ti.

Landy González Enríquez

No creo poder incluir a todas las personas que de una u otra forma me han ayudado a recorrer el camino en tan poco espacio. Tampoco pienso que con hacer mención de aquellos que tan importante han sido para mí muestre una medida de lo que realmente quisiera expresar. Aún así a todos ellos debo y quiero hacer llegar mis agradecimientos.

Agradezco:

A mi familia por mostrarme el camino y ayudarme a recorrerlo, por apoyarme e inculcarme un grupo de valores sin los cuales hoy no sería quien soy. Por ser mí ejemplo a seguir, mi inspiración y mi fuerza. Por cultivar mi carácter y cosechar mis logros, que mucho más que míos son nuestros logros. A mi madre por su paciencia, su bondad, su comprensión y por guiarme constantemente. A mi padre por todas sus lecciones y la sabiduría que todos estos años me ha transmitido. A mi hermana por ser única, espléndida, por impulsarme. A ellos, agradezco.

A mis amigos de la UCI por confiar en mí y darme la seguridad de confiar en ellos. Por llenar cada uno de los recuerdos que me llevo. Por sus consejos, sus charlas, por tantos ratos inolvidables, a ellos, agradezco. A mis amigos del barrio que desde que era niño han estado ahí. A mis amigos del cuarto, después de cinco años somos casi familia. A mis amigos canchistas por acumular juntos tantas horas de alegría y la cancha por ser una amiga más. A ellos, por ser parte de mi vida, agradezco.

A mi amigo y compañero de tesis por ser alguien excepcional desde el primero de estos 5 años y hasta hoy. Por hacer en esta última etapa menos compleja la tarea, agradezco.

A mis compañeros que tantos días hemos compartido. Nada sería igual sin ustedes.

A mis tutores y los profesores del proyecto y todos los profes que me han ayudado, por su entrega, por el conocimiento que han transmitido, por guiarme en este difícil camino, por compartir sus madrugadas, agradezco.

A todos los que alguna vez me preguntaron: -¿y la tesis…?

Dedicatoria

A toda mi familia que siempre apostó por mí.

A mis profes y maestros.

A mis amigos.

A Lexy.

Erislán Martinez Jera

A mi familia.

A mis amigos.

A mis profesores.

Landy González Enríquez

Resumen

Este Trabajo de Diploma surge en el marco de trabajo del proyecto Ensayos Clínicos, perteneciente al polo científico Gestión de Información Biomédica de la Facultad 6 en la Universidad de las Ciencias Informáticas, en colaboración con el Centro de Inmunología Molecular. Este proyecto tiene el objetivo de desarrollar una aplicación web llamada ***alasClínicas***, la cual gestiona información asociada a varios Ensayos Clínicos. Esta aplicación tiene varios módulos, uno de ellos es Enviar Datos y se encarga, entre otras cosas, de la **Gestión de Datos de Ensayos Clínicos,** cuyo desarrollo se muestra en este trabajo a partir del sistema OpenClinica.

La aplicación alasClínicas garantiza toda la gestión de pacientes por estudios y la gestión de los datos de cada hoja de CRD por momento de seguimiento; estos a su vez deben de estar asociados a los pacientes. Para lograr esto es la **Gestión de Datos de Ensayos Clínicos,** la cual brinda a los usuarios un grupo de interfaces para llevar a cabo estas actividades de forma más rápida y fácil. En este documento quedará evidenciado, a través de la descripción de la gestión de datos de ensayos clínicos y un grupo de diagramas, ilustraciones y conceptos, todo lo referente al proceso de desarrollo de la gestión de datos de ensayos clínicos, enfocado propiamente a la forma en la que se realiza en Cuba y como se gestiona utilizando la aplicación alasClínicas.

PALABRAS CLAVE: alasClínicas, Cronograma Específico, Ensayos Clínicos, Gestión de Datos, Hoja de CRD, Momento de Seguimiento, OpenClinica, Paciente.

Índice

Índice de Figuras

Introducción

Existen en el mundo numerosos centros de investigaciones científicas, los cuales se dedican a la creación de fármacos para la prevención y/o tratamiento de las enfermedades que hasta hoy no tienen cura. Para lograr la creación de cada uno de estos medicamentos realizan los Ensayos Clínicos (EC), los cuales son una parte fundamental en el proceso de desarrollo, aprobación e introducción en el mercado de nuevos fármacos y tratamientos. Descubrir si los nuevos agentes son seguros y eficaces, es el principal objetivo de la mayor parte de ellos. Sin embargo, también pueden estar destinados a aprender cómo prevenir la enfermedad, diagnosticar precozmente la patología o bien mejorar la calidad de vida de los enfermos[1].

Aunque la inmensa mayoría de estos centros se encuentra en países con un alto nivel de desarrollo, Cuba también cuenta con sus propios centros para la creación de fármacos, uno de ellos es el Centro de Inmunología Molecular (CIM), inaugurado el 5 de Diciembre de 1994 en el oeste de la Habana con el objetivo de obtener y producir nuevos biofármacos, destinados al tratamiento del cáncer y otras enfermedades crónicas no transmisibles e introducirlos en la Salud Pública Cubana, además de hacer la actividad científica y productiva económicamente sostenible y realizar aportes importantes a la economía del país[2].

Este centro actualmente lleva a cabo la conducción de varios EC, los cuales a su vez tienen asociada una gran cantidad de información que puede ser: datos del ensayo, imágenes, aprobaciones, modificaciones, además de los datos del tratamiento de cada paciente, los cuales deben ser recogidos durante el estudio de cada uno. Toda esta documentación es necesaria para cumplir con las buenas prácticas clínicas exigidas por todas las agencias reguladoras a nivel mundial, y debe ser almacenada por no menos de 15 años posterior al cierre del estudio de modo que pueda ser inspeccionada en cualquier momento por cualquiera de las agencias reguladoras.

Actualmente realizar la gestión de toda esta información es muy difícil debido a que cada estudio tiene asociado varios pacientes de los cuales se recogen gran cantidad de datos que son almacenados en Cuadernos de Recogida de Datos (CRD), de modo que cuando se cierra un estudio, lo que se tiene es un número de CRD los cuales tienen una gran cantidad de datos y como fueron manuscritos puede que tengan errores.

A medida que crece el número de estudios se va haciendo más difícil realizar la gestión de toda esta información, por lo es necesario utilizar alguna herramienta que permita la gestión de estos estudios para de esta forma lograr hacer más fácil el manejo de varios ellos a la vez, que permita además la gestión de los pacientes, los datos del tratamiento de estos, la validación y monitoreo de los datos de las hojas de los CRD y que permita también procesar los datos fácilmente una vez concluido alguno de ellos.

Esta herramienta permitiría el país reducir el costo de la conducción de EC en cuanto a recursos y esfuerzo, disminuir el tiempo de la conducción de estos estudios y así poder terminar a tiempo la creación de un medicamento que se estaba importando o poner en el mercado internacional nuevos medicamentos. Además es una tendencia actual, e incluso ya es está haciendo oficial que en algunos países no acepten EC que no tengan un sistema con el cual gestionar sus datos.

Uno de los Sistemas de Manejo de Datos de Ensayos Clínicos (CTMS por sus siglas en inglés) que cumplen con las características de la herramienta que se necesita es OpenClinica, el cual tiene la ventaja de ser un software libre y estar desarrollado con un grupo de herramientas libres también, a diferencia de muchos otros que son software privativo y bastante costoso. Otra ventaja de este es que logra de manera eficiente brindar muchas de las funcionalidades que se necesitan con la integración de sus 4 módulos: Enviar Datos, Extraer Datos, Gestionar Estudio y Administrar Empresa.

El módulo de Enviar Datos es el encargado de la Gestión de Datos de Ensayos Clínicos (GDEC) y del Monitoreo de estos datos. Esta investigación solo se centra en la GDEC, que es la encargada de la gestión de pacientes por estudios y de la gestión de los datos del tratamiento de estos pacientes por momentos de seguimientos. OpenClinica permite la GDEC ofreciendo un grupo de interfaces a través de las cuales se realizan todas las actividades que involucra de manera fácil y sencilla. Pero esta no cumple con todos los requisitos que necesita el CIM para llevar a cabo la conducción de sus EC, por lo que se plantea el siguiente **problema científico:**

> ¿Cómo adaptar la Gestión de Datos de Ensayos Clínicos del sistema OpenClinica al proceso cubano de conducción de Ensayos Clínicos?

El **objeto de estudio** en este trabajo es:

> Proceso cubano de conducción de Ensayos Clínicos en el sistema OpenClinica.

Enmarcado en el **campo de acción**:

Gestión de Datos de Ensayos Clínicos Cubanos en el sistema OpenClinica.

Después de definir el problema científico, objeto de estudio y campo de acción en que se enmarca la investigación, se propone como **objetivo general:**

Desarrollar funcionalidades para la Gestión de Datos de Ensayos Clínicos Cubanos, en el sistema OpenClinica.

Los **objetivos específicos** en los que se divide el objetivo general son:

- ✓ Definir las funcionalidades que serán agregadas o modificadas para la Gestión de Datos de Ensayos Clínicos Cubanos, en el sistema OpenClinica.
- ✓ Diseñar las clases que serán agregadas o modificadas para la Gestión de Datos de Ensayos Clínicos Cubanos, en el sistema OpenClinica.
- ✓ Implementar las funcionalidades que serán agregadas o modificadas para la Gestión de Datos de Ensayos Clínicos Cubanos, en el sistema OpenClinica.

Para el cumplimiento de estos objetivos se trazan las siguientes **tareas:**

- ✓ Estudio del sistema OpenClinica.
- ✓ Modelación del dominio.
- ✓ Levantamiento de requisitos a adicionar o modificar al sistema OpenClinica.
- ✓ Definición y descripción de los casos de uso del sistema.
- ✓ Realización de los diagrama de clases del diseño.
- ✓ Realización de los diagrama de interacción.
- ✓ Realización del modelo de datos.
- ✓ Realización de los diagramas de componentes.
- ✓ Implementación de las funcionalidades para el sistema.

El presente trabajo está estructurado en 4 capítulos:

En el **CAPÍTULO 1: Fundamentación Teórica** se realiza un estudio de la GDEC del sistema OpenClinica desde el punto de vista de la conducción de EC en Cuba, se explica además la metodología de desarrollo a utilizar y se describen las tecnologías y herramientas utilizadas para dar solución al problema planteado.

En el **CAPÍTULO 2: Características del Sistema** se describe el funcionamiento de la aplicación a través del modelo del dominio, se definen los requisitos funcionales y no funcionales del sistema, se describe la estructuración del sistema a través del diagrama de casos de uso del sistema y las descripciones de los casos de uso para comprender mejor el funcionamiento de la aplicación.

En el **CAPÍTULO 3: Diseño del Sistema** se describe el diseño de la GDEC a través de varios elementos como los diagramas de clases del diseño, los diagramas de interacción, el modelo de datos y el diagrama de despliegue. La realización de este capítulo es de suma importancia ya que de él depende la implementación correcta del sistema.

En el **Capítulo 4: Implementación del Sistema** se comienza precisamente con los resultados arrojados por el diseño del sistema, en este capítulo se realiza la implementación del sistema logrando de esta manera alcanzar el objetivo general del trabajo y así darle solución al problema científico.

CAPÍTULO 1: Fundamentación Teórica

En este capítulo se realiza un análisis sobre la Gestión de Datos de Ensayos Clínicos y como funciona este proceso actualmente en el Centro de Inmunología Molecular. Se justifica la necesidad de un sistema informático y para ello se realiza un estudio sobre el sistema OpenClinica. Además se explica la metodología de desarrollo, tecnologías y herramientas utilizadas para dar solución al problema planteado.

1.1. Situación Actual

Actualmente el proceso de recogida de datos de un Ensayo Clínico (EC) en Cuba se hace de forma manual en los hospitales que son autorizados para esta tarea. Los datos de los pacientes en los momentos de seguimientos, se recogen y después se almacenan en las hojas que conforman los Cuadernos de Recogida de Datos (CRD) y paulatinamente se llevan al Centro de Inmunología Molecular (CIM), donde dos operadores se encargan de introducirlos a una base de datos utilizando interfaces que lejos de brindar algún tipo de información solo sirven de entrada para los datos.

Se necesita un sistema para la conducción de EC que permita almacenar la información contenida en estas hojas de CRD, pero que además constantemente vaya notificando un conjunto de criterios a los conductores del estudio, sobre un grupo de datos tanto de pacientes y hojas de CRD, como también de la asociación de ellos en un momento de seguimiento, y de esta forma facilitar el curso y la toma de decisiones en la conducción de un estudio.

1.2. ¿Qué es Gestión de Datos de Ensayos Clínicos?

La Gestión de Datos de Ensayos Clínicos (GDEC) garantiza toda la gestión de pacientes por estudios y la gestión de los datos de cada hoja de CRD por momento de seguimiento de estos pacientes. Esto incluye las actividades de incluir pacientes, excluirlos, chequear sus datos, asociarle una planificación de momento de seguimiento, llenar sus hojas de CRD por momentos de seguimientos y corregir esos datos una vez que se detecten errores en la revisión que hace el Monitor, entre otras actividades.

En el Capítulo 2 se describe con más detalles cada una de las funcionalidades que brinda la GDEC en el sistema ***alasClínicas***.

1.3. Estudio de OpenClinica.

El Sistema de Manejo de Datos de Ensayos Clínicos (SMDEC) o Clinical Trials Management System (CTMS) en inglés, OpenClinica, es un software libre y está liberado bajo la licencia LGPL[3]. Cuando se habla de software libre se refiere a la libertad de uso, no la ausencia de coste, por lo tanto con OpenClinica 2.5.3 se puede disfrutar de las 4 libertades que ofrece cualquier software libre:

- ✓ Libertad de ejecutar el programa para cualquier propósito.
- ✓ Libertad de estudiar cómo trabaja el programa, y adaptarlo a sus necesidades.
- ✓ Libertad de redistribuir copias.
- ✓ Libertad de mejorar el programa y publicar sus mejoras, y versiones modificadas en general[4].

La licencia LGPL por su parte está diseñada para asegurar que los usuarios tengan la libertad de distribuir copias de software libre (y cobrar por el servicio si quieren), que reciban el código fuente o puedan obtenerlo, que puedan modificar el software y usar piezas de él en nuevos programas libres y que estén informado de que pueden hacer todas estas cosas[5]. Por lo que se puede adaptar OpenClinica a las necesidades del CIM sin ningún tipo de conflicto legal.

Está desarrollado por Akaza Research y tiene una comunidad de más de 5000 usuarios de 76 países que constantemente actualizan y liberan estas actualizaciones hasta lograr que se haya convertido en el líder de la comercialización de software libre para EC[6].

OpenClinica está estructurado en 4 módulos: Enviar Datos, Extraer Datos, Gestionar Estudio y Administrar Empresa. Cada uno de ellos brinda un grupo de funcionalidades que en general logran de manera eficiente, flexible y con bajo costo, que se gestionen varios estudios a la vez. De estos Enviar Datos es el que se encarga de la GDEC y del Monitoreo de estos datos, pero en este trabajo solo nos centraremos en la GDEC.

La GDEC en OpenClinica permite incluir pacientes al estudio, excluirlos, planificar sus momentos de seguimiento uno a uno, gestionar los datos de las hojas de CRD por momentos de seguimiento, entre otras cosas. Según las necesidades del CIM, hay algunas funcionalidades deben ser agregadas al este sistema y hay otras que ser cambiadas porque no cumplen con las políticas necesarias para la conducción de EC en Cuba. Por ejemplo, este sistema, según las necesidades del CIM, debe de permitir planificar los momentos de seguimiento en un cronograma general, el cual debe de ser

cumplido por todos los pacientes que pertenezcan a ese estudio y en vez de programarse uno por uno, deben de programarse automáticamente todos a partir de una fecha indicada por el Coordinador de Investigación Clínica y la planificación indicada para cada uno en el cronograma general, en este sistema se recogen datos innecesarios a la hora de vincular pacientes a un estudio, no se permite la creación de notas del sitio y según el CIM estas son necesarias, por solo mencionar algunos ejemplos.

Por todo esto es que el proyecto Ensayos Clínicos decide adaptar OpenClinica a las necesidades del CIM y esa decisión se ratifica en esta investigación.

1.4. Metodología, tecnologías y herramientas.

La propuesta de herramientas, tecnologías y metodología que se muestra a continuación surge inicialmente a partir de haber determinado modificar a OpenClinica 2.5.3 para dar cumplimiento al objetivo propuesto, pero además por sus características, OpenClinica es una aplicación web, está programada en Java utilizando Eclipse como entorno de desarrollo, el servidor web que utiliza es Jakarta Tomcat y la base de datos esta creada en PostgreSQL. Todas son herramientas y tecnologías libres, de ahí que sean las propuestas para el desarrollo de la aplicación ***alasClínicas*** a partir del sistema OpenClinica.

1.4.1. Metodología: RUP

Investigaciones precedentes a esta han propuesto a ***Rational Unified Process (RUP)*** como la metodología más apropiada para el desarrollo de este sistema, así que esa será la metodología que se usará. Además es apropiada para proyectos grandes y es el proceso de desarrollo de software que forma parte, junto con el Lenguaje Unificado de Modelado (UML), de la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Una de las ventajas de RUP es que no se trata de proceso con pasos firmemente establecidos e inamovibles, sino de un conjunto de metodologías que se pueden adaptar al contexto y necesidades de cada organización, lo cual lo hace ideal para proyectos de desarrollo de cualquier tamaño y alcance que sean requeridos en pequeñas, medianas y grandes empresas. Además RUP permite:

- ✓ Mitigación temprana de posibles riesgos altos.
- ✓ Progreso visible en las primeras etapas.
- ✓ Temprana retroalimentación que se ajuste a las necesidades reales.

- ✓ Gestión de la complejidad.
- ✓ El conocimiento adquirido en una iteración puede aplicarse de iteración a iteración.

1.4.2. Herramienta CASE: Visual Paradigm para UML

Visual Paradigm for UML es una herramienta CASE profesional 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 CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Lista de algunas características:

- ✓ Soporte de UML versión 2.1
- ✓ Modelado colaborativo con CVS y Subversion (nueva característica)
- ✓ Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).
- ✓ Ingeniería de ida y vuelta
- ✓ Ingeniería inversa Java
- ✓ Generación de código - Modelo a código, diagrama a código
- ✓ Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso
- ✓ Diagramas EJB - Visualización de sistemas EJB.
- ✓ Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
- ✓ Soporte ORM - Generación de objetos Java desde la base de datos

- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos
- ✓ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación
- ✓ Generador de informes para generación de documentación
- ✓ Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML
- ✓ Importación y exportación de ficheros XMI

Se utiliza Visual Paradigm for UML en su versión 6.4 para el desarrollo del sistema **alasClínicas.**

1.4.3. Tecnología utilizada para la programación: J2EE

Java es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su ***plataforma J2EE***. La tecnología Java está compuesta básicamente por dos elementos: el lenguaje Java y su plataforma. Plataforma se refiere a la máquina virtual de Java (JVM por sus siglas en inglés).

Una de las principales características que favoreció el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que el mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además es un lenguaje orientado a objetos que resuelve los problemas en la complejidad de los sistemas. Toma mucha de la sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Debido a la necesidad del mercado de desarrollo de software, respecto a contar con medios y herramientas que permitan construir aplicaciones “empresariales”, se diseñó la plataforma abierta y estándar de Java para este mejor conocida como Java 2 Enterprise Edition, (J2EE). Se le denomina plataforma porque proporciona especificaciones técnicas que describen el lenguaje pero, además, provee las herramientas para implementar productos de software (aplicaciones) basados en dichas especificaciones.

J2EE ha sido diseñada para aplicaciones distribuidas que son construidas con base en componentes (unidades funcionales de software), los cuales interaccionan entre sí para formar parte de una aplicación J2EE. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea en la parte del servidor J2EE que le ofrece al componente ciertos servicios de bajo nivel y de sistema (tales como seguridad, manejo de concurrencia, persistencia y transacciones)[7].

Los componentes definidos por J2EE son:

- ✓ Servlet: Es un componente desarrollado con el objetivo de procesar peticiones de un cliente (requests) y generar respuestas con contenidos web dinámicos. Para ser ejecutados es necesaria la utilización de un servidor que de soporte a servlets y su contenedor (container).
- ✓ Páginas Servidoras de Java (JSP por sus siglas en ingles): Es un componente para construir fácilmente aplicaciones con contenido web como HTML, DHTML, XHTML y XML, en forma dinámica, con gran poder y flexibilidad. JSP se basa en los siguientes conceptos:
  - ➢ Plantillas. Una parte importante del contenido está compuesto por una plantilla. Típicamente en esta plantilla se encuentran fragmentos HTML o de texto.
  - ➢ Contenido dinámico. JSP provee una forma simple de agregar datos dinámicos a cada plantilla al permitir incrustar instrucciones de programación en este. El lenguaje es generalmente Java, aunque se puede utilizar otro lenguaje que sea soportado por el contenedor (container) JSP.
  - ➢ Encapsulación de funcionalidad. JSP provee dos formas distintas para encapsular funcionalidad: componentes JavaBeans y bibliotecas de etiquetas (taglibs)[8].
- ✓ Enterprise JavaBeans (EJB): Es una arquitectura que permite la creación de componentes de aplicaciones distribuidas y orientadas a transacciones. Las aplicaciones escritas utilizando EJB son escalables, transaccionales y multiusuario.

J2EE no es sólo una tecnología, sino un estándar de desarrollo, construcción y despliegue de aplicaciones. J2EE es una especificación que define una plataforma para crear aplicaciones empresariales utilizando un modelo multicapa, dividiendo la aplicación en diferentes niveles, cada uno especializándose en una tarea en particular.

1.4.4. Entorno de Desarrollo Integrado: Eclipse

En la web oficial de ***Eclipse***, se define como un IDE para todo y para nada en particular ("An IDE for everything and nothing in particular"). Eclipse es, en el fondo, únicamente un armazón sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los complementos (plugins) adecuados.

La arquitectura de complementos (plugins) de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML como el Visual Paradim for UML, editores visuales de interfaces, ayuda en línea para librerías, herramientas para el control de versiones y otras.

El Proyecto Eclipse aúna tanto el desarrollo del IDE Eclipse como de algunos de los complementos (plugins) más importantes (como el JDT, plugin para el lenguaje Java, o el CDT, complemento (plugin) para el lenguaje C/C++).

Este proyecto también alcanza a las librerías que sirven como base para la construcción del IDE Eclipse (pero pueden ser utilizadas de forma completamente independiente), como por ejemplo, la librería de widgets SWT, la cual es conjunto de componentes para construir interfaces gráficas en Java.

Se utiliza Eclipse en su versión 3.4 para el desarrollo del sistema **alasClínicas.**

1.4.5. Servidor web: Apache Tomcat

Tomcat (también llamado ***Jakarta Tomcat*** o Apache Tomcat) es un servidor web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets y el contenedor de servlets Catalina.

El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Hoy en día es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Como fue escrito en Java, funciona en cualquier sistema operativo que disponga de la JVM. Es mantenido y desarrollado por miembros de la Fundación de Software Apache (Apache Software Foundation) y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Licencia de Software Apache (Apache Software

Licence). Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 6.x, que implementan las especificaciones de Servlet 2.5 y de JSP 2.1. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina y en la versión 5.x que esta implementado a partir de las especificaciones Servlet 2.4 y JSP 2.0 incorpora:

- ✓ Recolección de basura reducida.
- ✓ Capa envolvente nativa para Windows y Unix para la integración de las plataformas.
- ✓ Análisis rápido JSP.

Este servidor tiene grandes ventajas ya que es gratis, es fácil de instalar, se ejecuta en máquinas pequeñas y es compatible con las API más recientes de Java. Puede descargarse, instalarse y probarse en muchos servidores en menos de una hora. Tomcat ocupa muy poco espacio, teniendo su código binario (todo clases de Java) un tamaño total de apenas un megabyte, de modo que no es raro que se ejecute tan deprisa.

Otra ventaja de Tomcat es que es muy fiable, innumerables empresas utilizan Tomcat, y como dijo Linus Torvalds acerca del código libre, "Dado un número suficiente de ojos, todos los errores son irrelevantes". Dicho de otra forma, la solidez de Tomcat se basa en que miles de desarrolladores contribuyen a mejorar su código.

Se utiliza Apache Tomcat en su versión 5.5 para el desarrollo del sistema **alasClínicas.**

1.4.6. Sistema Gestor de Base de Datos: PostgreSQL

PostgreSQL es un sistema gestor de base de datos relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el motor de bases de datos de código abierto más potente del momento y en sus últimas versiones empieza a no tener que envidiarle nada a otras bases de datos comerciales.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. En los últimos años se han concentrado mucho en la velocidad de proceso y en características demandadas en el mundo empresarial.

La última serie de producción es la 8.4, siendo la última versión disponible la 8.4 Beta. PostgreSQL se puede ejecutar en la gran mayoría de sistemas operativos existentes en la actualidad, entre ellos Linux y UNIX y Windows. Las características más importantes y soportadas son:

- ✓ Es una base de datos 100% ACID.
- ✓ Llaves ajenas (foreign keys).
- ✓ Uniones (Joins).
- ✓ Vistas (Views).
- ✓ Disparadores (Triggers).
- ✓ Reglas (Rules).
- ✓ Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de oracle).
- ✓ Numerosos tipos de datos, posibilidades de definir nuevos tipos.
- ✓ Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido,...).
- ✓ Herencia de tablas (Inheritance).
- ✓ Unicode.
- ✓ Juegos de caracteres internacionales.
- ✓ Control de Concurrencia Multiversiones (Multi-Version Concurrency Control o MVCC).
- ✓ Acceso encriptado vía SSL.
- ✓ APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP y muchos otros lenguajes.
- ✓ Completa documentación.

Otra característica muy a tener en cuenta es lo bien que PostgreSQL funciona con grandes cantidades de datos y una alta concurrencia, con muchos usuarios accediendo a la vez el sistema.

Se utiliza PostgreSQL en su versión 8.2.4 para el desarrollo del sistema **alasClínicas,** este es uno de los sistemas gestores de base de datos que proponen los desarrolladores de OpenClinica, el otro es Oracle, pero como es privativo escogemos a PostgreSQL**.**

1.4.7. Herramienta para el Control de Versiones: Subversion

Subversion es un controlador de versiones empleado en la administración de archivos utilizados o contenidos en el desarrollo de software.

Un repositorio, depósito o archivo es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

Subversion es un repositorio en forma de árbol con una jerarquía de directorios y archivos, de ahí que constituya un sistema centralizado para compartir información que permite gestionar archivos y directorios, y sus cambios a través del tiempo así como recrear un proyecto desde cualquier momento en su historia[9]. El objetivo de un sistema de control de versiones es el de permitir editar de forma colaborativa y compartir información.

- ✓ Subversion recuerda cada cambio que se haya realizado en el repositorio.
- ✓ Recuerda cambios realizado a cada archivo así como cambios en el árbol de directorios:
  - ➢ Archivos y directorios nuevos
  - ➢ Archivos y directorios borrados
  - ➢ Archivos y directorios modificados o cambiados de lugar.
- ✓ Gestiona cambios a través del tiempo
  - ➢ Generalmente un cliente lee la versión más reciente del árbol de directorios y archivos.
  - ➢ Subversion provee la habilidad de leer estados anteriores del sistema de archivos.
  - ➢ Se pueden conocer los cambios realizados, cuándo se realizaron, y quién realizó dichos cambios.

Además Subversion presenta un grupo de características que lo destacan entre otros controladores de versiones como su antecesor CVN, ellas son:

- ✓ *Fuerte integración con Apache*: Esto permite definir controles de acceso avanzados y navegación vía web para consultar el depósito de archivos.
- ✓ *Transparencia al eliminar y cambiar nombres de archivos:* Facilita el proceso sin requerir de intervención manual en el repositorio para lograrlo.
- ✓ *Copias ligeras sobre ramificaciones:* Independientemente del número de ramificaciones creadas, mantiene un árbol diferencial de cambios, minimizando así el espacio consumido en el depósito.
- ✓ *Copias diferenciales de archivos binarios:* Basado en el mismo principio de copias ligeras, Subversion es capaz de mantener un control diferencial sobre cualquier archivo binario del depósito así reduciendo el consumo de espacio.

Se utiliza Subversion en su versión 1.6 para el desarrollo del sistema **alasClínicas.**

1.5. Conclusiones

La decisión del proyecto Ensayos Clínico de la Facultad 6 de la Universidad de las Ciencias Informáticas de adaptar OpenClinica a las necesidades del CIM está dada porque este es un sistema que cumple con muchas de las demandas del CIM pero a la vez le faltan algunas o hay que modificarle otras, además es un sistema robusto, bastante probado y libre; está desarrollado con un grupo de herramientas libres, las cuales fueron caracterizadas en este capítulo; y su modificación está permitida por los desarrolladores del mismo.

CAPÍTULO 2: Características del Sistema

Como el problema planteado es "¿Cómo adaptar la GDEC del sistema OpenClinica al proceso cubano de conducción de EC?" no es posible identificar procesos de negocio porque no existe un proceso para lograr esta adaptación, por lo que se realizará el modelo de dominio para dar paso a definir los requisitos funcionales y no funcionales del sistema y así lograr describir mejor la estructuración del sistema a través del diagrama de casos de uso (CU) del sistema y las descripciones de los CU para comprender mejor el funcionamiento de la aplicación.

2.1. *Modelado del Dominio*

2.1.1. Modelo de Dominio

El modelo de dominio representa un modelo de conceptos y no un modelo de objetos. Es un diccionario visual de términos importantes del dominio y sus asociaciones que utiliza la notación UML de diagrama de estructura estática[10]. Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema de software.

2.1.2. Diagrama de conceptos del dominio

A continuación se muestra el modelo de dominio que recogen los principales conceptos que se identifican en la GDEC y como se relacionan entre sí. Además se agrupan en el paquete Gestión de Datos – OpenClinica los conceptos que están presentes en el sistema OpenClinica, para delimitar de esta forma aquellos que deben ser adicionados para obtener el sistema alasClínicas.

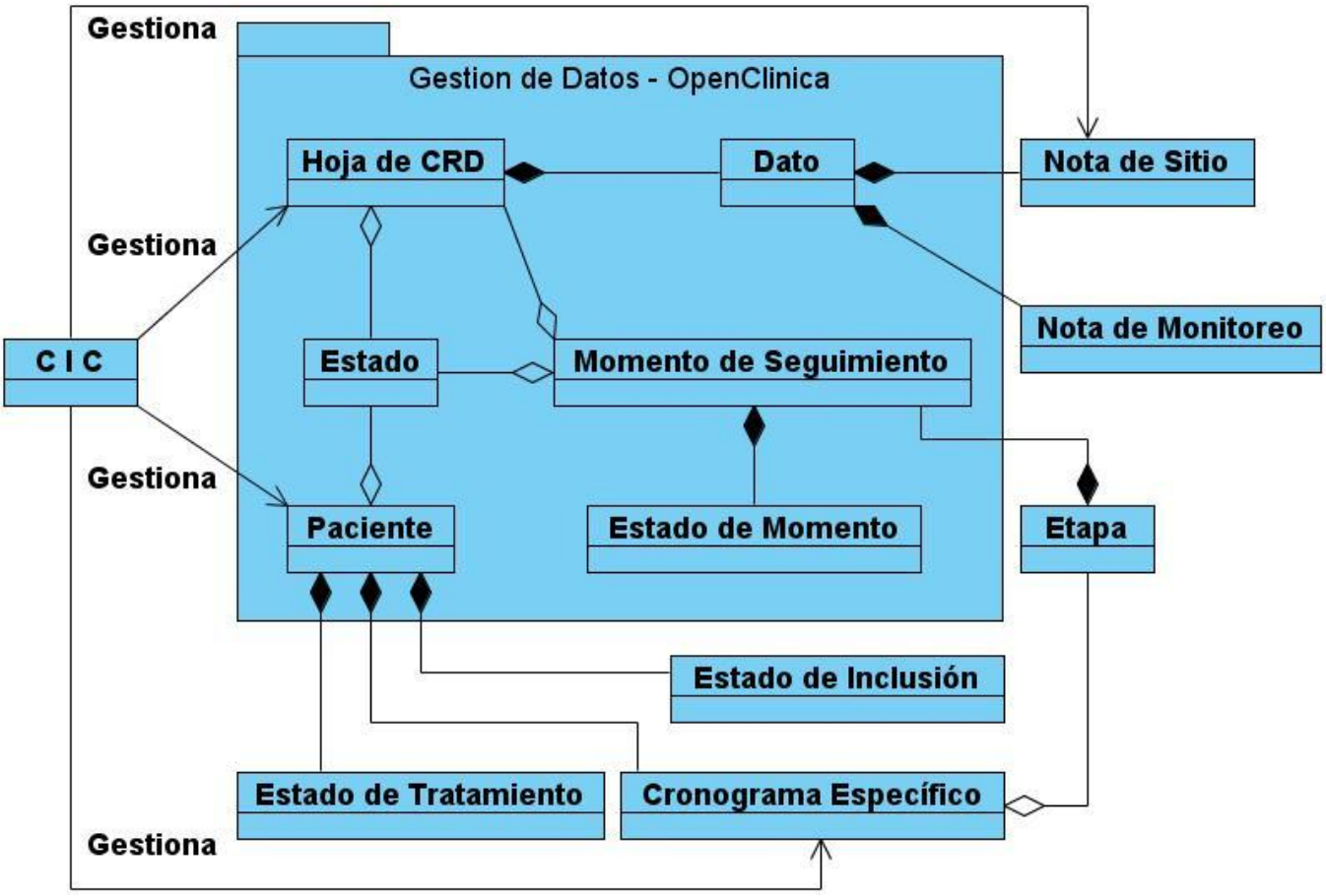


Fig. # 1 Modelo de dominio.

2.1.3. Definición de los conceptos del modelo del dominio

CIC

Es una representación del rol Coordinador de Investigación Clínica. El cual es el responsable de gestionar pacientes, los datos de las hojas de CRD asociadas a los momentos de seguimiento definidos para los pacientes bajo su responsabilidad, las notas del sitio, y además, gestionar el cronograma específico de cada paciente.

Paciente

Representa la asociación de un sujeto a un estudio, es decir, a un sujeto se le llama paciente una vez que es incluido en un estudio.

Cronograma Específico

Es un concepto nuevo en OpenClinica que representa la asociación de una planificación general de momentos de seguimiento a un paciente específico. Contiene la fecha para la cual se planificó cada momento de seguimiento, la etapa a la que pertenece cada uno y el estado.

Etapa

Representa un periodo en el tratamiento de un paciente. Tiene un día inicial y un día final.

Momento de Seguimiento

Es una asociación de una o varias hojas de CRD a un paciente en una fecha, con un plazo de tiempo para su llenado.

Hojas de CRD

Representa el conjunto de datos pertenecientes a un examen definido por los conductores del estudio que debe de ser llenado en algún momento durante la conducción del mismo.

Dato

Representa el valor de un elemento o variable perteneciente a una hoja de CRD de un paciente, que se recogió en algún momento de seguimiento.

Estado

Es una propiedad que tienen casi todos los conceptos del sistema. En el caso de las hojas de CRD representa el estado de cada una respecto a los datos que le son introducidos (No Iniciada, Iniciada, Completada, Firmada, Monitoreado Iniciado, Monitoreado Completado o Atrasada). Si se está hablando de momento de seguimiento, representa el estado que tiene este respecto a su disponibilidad en el estudio, es decir, si existe o esta borrado. Así mismo pasa con los pacientes.

Estado de Momento

Representa el estado que tiene el momento de seguimiento respecto al estado de las hojas de CRD asociadas a él (No Iniciado, Iniciado, Completado, Firmado, Monitoreado Iniciado, Monitoreado Completado o Atrasado).

Estado de Tratamiento

Representa el estado que tiene el tratamiento del paciente según la etapa del estudio en que se encuentre el dicho paciente (Evaluación, Tratamiento, Seguimiento o Interrumpido).

Estado de Inclusión

Representa el estado que tiene el paciente en el estudio de acuerdo a su inclusión.

Nota de Sitio

Representa una aclaración que se le agrega a un dato específico en el llenado de una hoja de CRD por el CIC.

Nota de Monitoreo

Representa una aclaración que se le agrega a un dato específico durante el proceso de monitoreo de los datos por el Monitor.

2.2. Modelado del sistema

2.2.1. Requisitos Funcionales

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

A continuación se presenta el listado de todas las funcionalidades que brinda la GDEC, en la descripción de CU del sistema, se especificará cuales realmente son las que se agregan o modifican al sistema OpenClinica.

R1 Generar cronograma específico.

A partir de una fecha indicada por el CIC, se le asigna una fecha a todos los momentos de seguimiento programados de un paciente, pertenecientes a las etapas de “Tratamiento” y “Seguimiento”, según el orden de la planificación de los mismos en el cronograma general.

R2 Adicionar momentos de seguimiento no programados.

A partir de la fecha de inclusión de un paciente al estudio se le puede asignar un momento de seguimiento no programado, indicando como fecha de inicio la fecha actual y un día como plazo de tiempo para su llenado.

R3 Eliminar momentos de seguimiento no programados.

Durante el curso de un estudio se puede eliminar un momento de seguimiento no programado en estado “Atrasado” a cualquier paciente.

R4 Visualizar cronograma específico de un paciente.

Se muestra de todos los momentos de seguimiento definidos para un paciente, el nombre, el estado, fecha de inicio y etapa asociada a los mismos.

R5 Visualizar momentos de seguimiento programados.

Se muestra de todos los momentos de seguimiento programados definidos para un paciente con fecha de inicio menor o igual a la fecha actual, el nombre, el estado, fecha de inicio, plazo de tiempo para su llenado, cantidad de notas de monitoreo con estado “Nueva” o “Actualizada” y las acciones asociadas a los mismos.

R6 Visualizar momentos de seguimiento no programados.

Se muestra de todos los momentos de seguimiento no programados definidos para un paciente con fecha de inicio menor o igual a la fecha actual, el nombre, el estado, fecha de inicio, cantidad de notas de monitoreo con estado “Nueva” o “Actualizada” y las acciones asociadas a los mismos.

R7 Establecer estado de cada momento de seguimiento.

Se le asigna a cada momento de seguimiento un estado, según el estado de las hojas de CRD asociadas a él.

R8 Establecer estado a cada hoja de CRD.

Se le asigna un estado a cada hoja de CRD asociada a un momento de seguimiento según los datos que se le hayan introducido.

R9 Visualizar listado de hojas de CRD asociado a un momento de seguimiento.

Se muestra de todas las hojas de CRD asociadas a un momento de seguimiento, el nombre, el estado, la fecha de llenado, la cantidad de notas de monitoreo con estado “Nueva” o “Actualizada” y las acciones asociadas a las mismas.

R10 Introducir datos a cada hoja de CRD asociada a un momento de seguimiento.

El CIC introduce los datos pertenecientes a una hoja de CRD asociada a un momento de seguimiento definido para un paciente y posteriormente son almacenados por el sistema.

R11 Mostrar datos de cada hoja de CRD asociada a un momento de seguimiento.

Se muestran los datos pertenecientes a una hoja de CRD asociada a un momento de seguimiento definido para un paciente.

R12 Modificar datos de cada hoja de CRD asociada a un momento de seguimiento.

El CIC introduce los datos a modificar de una hoja de CRD asociada a un momento de seguimiento definido para un paciente y posteriormente el sistema actualiza el valor de los datos que ya tenía almacenado de esa hoja de CRD.

R13 Crear nota del sitio.

El CIC puede crear una nota del sitio en el llenado de un dato en caso de ser necesaria alguna aclaración o algún comentario sobre un dato.

R14 Mostrar nota del sitio.

Se muestra la nota del sitio creada por el CIC en el llenado de un dato.

R15 Visualizar listado de pacientes.

Se muestra de todos los pacientes que están asociados a un estudio determinado, el nombre del paciente, estado de inclusión, estado de tratamiento, próximo momento de seguimiento, último momento de seguimiento y las acciones asociadas a los mismos.

R16 Insertar pacientes.

Se insertan pacientes a un estudio determinado.

R17 Adicionar momentos de seguimiento de la etapa de evaluación al insertar un paciente.

Al insertar un paciente se le asocian los momentos de seguimiento definidos para la etapa de “Evaluación” en el cronograma general. La fecha de inicio de cada uno de ellos se genera a partir de la fecha de inclusión del paciente.

R18 Mostrar datos del paciente.

Se muestra de un paciente determinado todos sus datos, incluido el nombre del estudio al que pertenece.

R19 Interrumpir paciente.

Se le establece el estado de tratamiento “Interrumpido” al paciente y el sistema no obligará el llenado de los momentos posteriores al último momento de seguimiento llenado.

R20 Establecer estado de tratamiento del paciente.

Se le establece un estado de tratamiento al paciente de acuerdo a la etapa del estudio en que se encuentre el mismo.

2.2.2. Requisitos no Funcionales

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, entre otros, representando aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica[11]. A continuación se describe un grupo de requisitos no funcionales que incluirá el sistema alasClínicas.

Apariencia o Interfaz Externa

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

Usabilidad

- ✓ La aplicación tendrá un ambiente sencillo y será fácil de manejar para los usuarios.

Fiabilidad

- ✓ La aplicación tendrá un sistema de trazas que registran el flujo constante de los datos y los responsables de sus cambios.
- ✓ La seguridad de la información será garantizada evitando su eliminación de la base de datos, permitiendo solo modificar su estado o su acceso.

Soporte

- ✓ Una vez terminada la aplicación se instalará en el CIM para realizar pruebas piloto del software y pruebas de despliegue.

Seguridad

- ✓ El acceso a cualquier manipulación del sistema, tanto entrada como análisis de datos estará sometido a un proceso de autentificación del usuario donde será especificado el rol, usuario y contraseña.
- ✓ Cada usuario tendrá asignado uno o varios roles en el sistema. Cada rol definido tendrá niveles de acceso al Software.
- ✓ Todo cambio o modificación en el sistema será atribuible a un usuario particular según su autenticación.

Software

- ✓ Para la instalación de la aplicación se debe disponer del sistema operativo Windows o GNU Linux.
- ✓ Servidor de Base de Datos Postgres 8.2.4.
- ✓ Navegador: Internet Explorer 5.0 o superior, Mozilla Firefox 1.0 o superior.
- ✓ Web Server: Apache Tomcat 5.5 o superior.
- ✓ Debe estar instalada la máquina virtual de Java SDK 1.5.x.

Hardware

- ✓ Microprocesador PENTIUM IV (o superior).

- ✓ Mínimo memoria RAM 256 (1Gb Recomendado).
- ✓ El servidor de Base de Datos debe tener 1 a 3 TB de espacio disponibles pues el volumen de información es bastante grande y perdura en el tiempo hasta 15 años.

Disponibilidad

- ✓ La aplicación debe mantenerse funcionando las 24 horas del día y los siete días de la semana.
- ✓ El servidor de aplicación debe soportar un aumento de usuarios concurrentes por minuto de 1 a 400.

Licencia

Se estará usando una herramienta de software libre, licencia de código abierto GNU/GPL (¡sin costes por licencia!). La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se la denomina contrato de licencia o acuerdo de licencia.

2.2.3. Patrón de Casos de Uso: CRUD

Un patrón de casos de uso captura técnicas para que un modelo sea fácil de mantener, reusable y entendible. Los patrones de casos de uso capturan mejores prácticas para modelar casos de uso. La aplicación de este tipo de patrones trae como beneficios:

- ✓ Aumentar la productividad.
- ✓ Reutilizar elementos existentes.
- ✓ Evitar el retrabajo por errores.
- ✓ No invertir tiempo en resolver problemas ya resueltos.

Para la realización del diagrama de casos de uso del sistema se utilizó el patrón de casos de uso CRUD, este tipo de patrón se utiliza cuando se quiere realizar altas, bajas, cambios y consultas a alguna entidad del sistema. Su nombre es un acrónimo de las palabras Crear, Mostrar, Modificar y Eliminar (en inglés: Create, Read, Update y Delete). El mismo consiste en un caso de uso completo para administrar la información[12]. El patrón CRUD puede ser parcial o completo dependiendo de la totalidad de la información gestionada. Para la confección del diagrama de casos de uso se utilizó el

CRUD parcial y su uso queda evidenciado en el caso de uso Gestionar datos de hoja, en este caso de uso se agrupan las funcionalidades encargadas de introducir, mortar y modificar datos, pero no permite eliminar los datos.

2.2.4. Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso del sistema describe las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso. Estos diagramas sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen qué es lo que debe hacer el sistema, el cómo será descrito posteriormente.

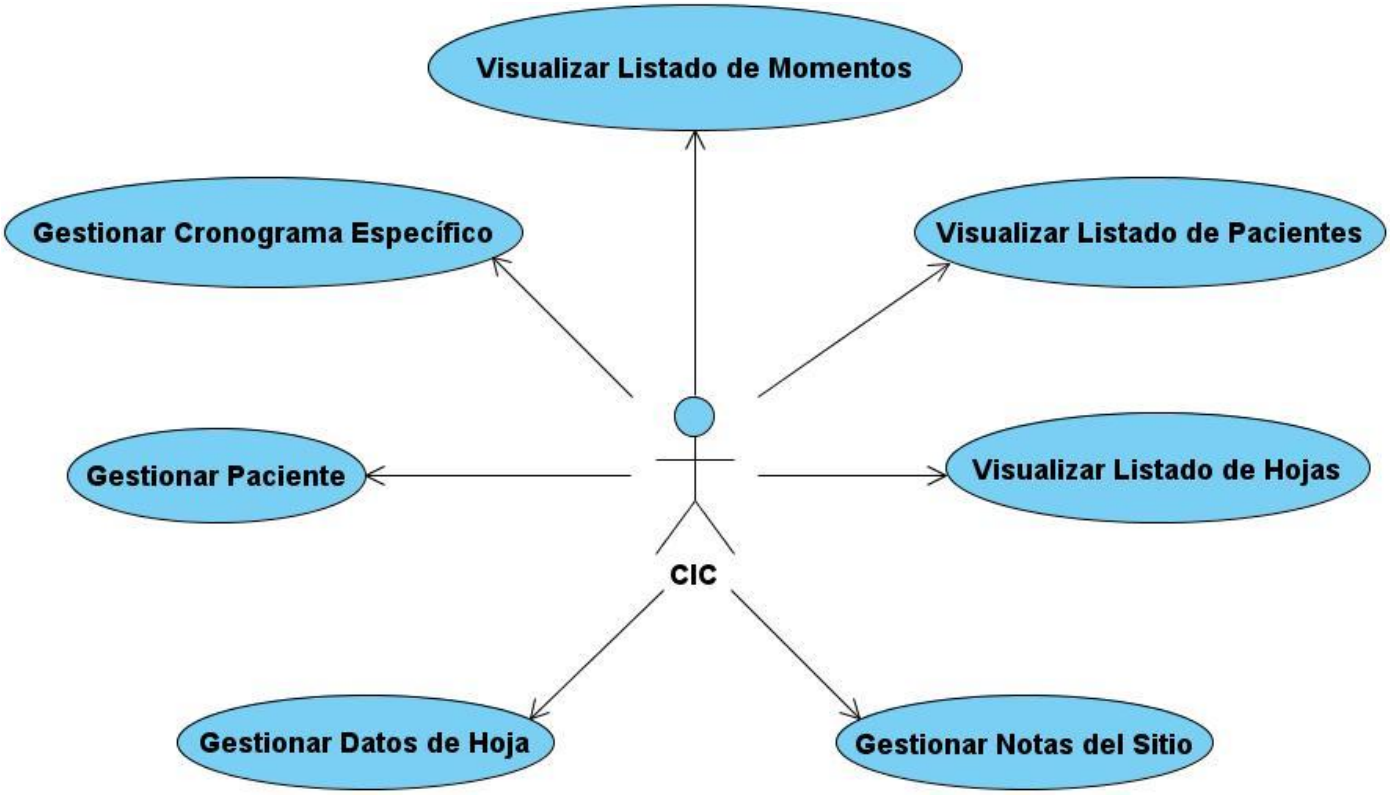


Fig. # 2 Diagrama de casos de uso del sistema.

2.2.5. Actor del Sistema

Actor	Descripción

CIC	El Coordinador de Investigación Clínica es el responsable de gestionar pacientes por Estudios, los datos de las hojas de CRD asociadas a los momentos de seguimiento definidos para estos pacientes bajo su responsabilidad, gestionar las notas del sitio, y además gestionar el cronograma específico de dichos pacientes.

2.2.6. Descripción de Casos de Uso del Sistema

2.2.6.1. Visualizar listado de pacientes.

Este CU ya existe en el sistema OpenClinica pero será modificado porque los datos que se muestra en este no son relevantes para el CIM. En su lugar se mostrarán los que son verdaderamente importantes para la conducción de EC en Cuba, como a continuación se describe.

Caso de Uso:	Visualizar listado de pacientes
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el CIC indica ver el listado de pacientes de un estudio. El sistema muestra el listado de pacientes con un grupo de datos de interés asociados a cada uno.
Precondiciones:	
Referencias	R15
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El CIC indica ver todos los pacientes de un estudio en el menú principal.	2 El sistema busca todos los pacientes del estudio que está activo. 3 El sistema comprueba si el cronograma específico de cada

	paciente ya está generado. 4 El sistema muestra al CIC el listado de pacientes especificando el identificador del paciente en el estudio, el estado de la inclusión, estado del tratamiento, la fecha del último momento de seguimiento, del próximo momento de seguimiento y acciones que puede hacer el CIC con cada uno de ellos (Ver Datos del paciente, Ver listado de momentos de seguimientos o Interrumpir paciente si el cronograma específico del paciente está generado) y finaliza así el caso de uso.

Prototipo de Interfaz

Poscondiciones	

2.2.6.2. Visualizar listado de momentos.

Este CU ya existe en el sistema OpenClinica pero será modificado porque la información que se muestra de los momentos de seguimientos no cumplen con todas las necesidades del CIM, por lo que se mostrarán los datos que realmente son importantes para la conducción de EC en Cuba.

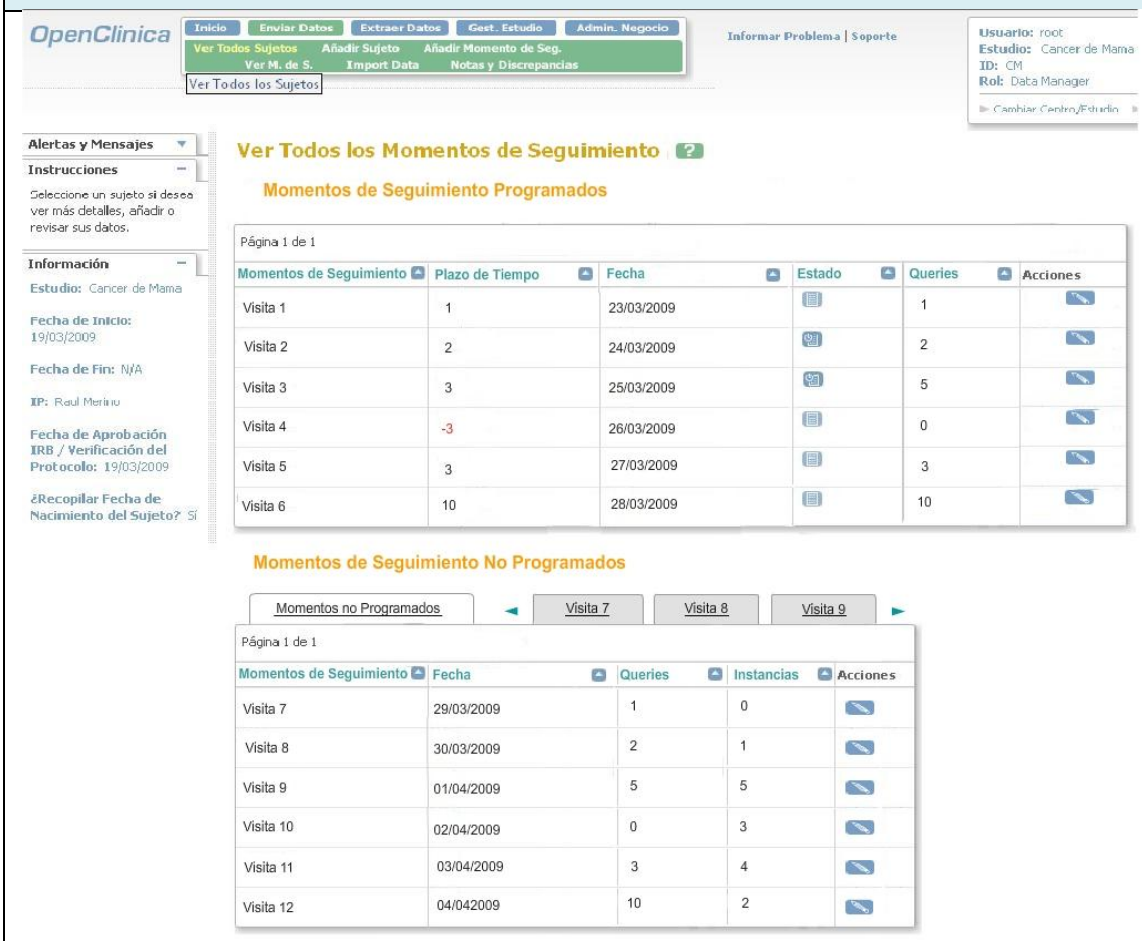
Caso de Uso:	Visualizar listado de momentos
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el CIC indica ver el listado de momentos de seguimiento definidos para un paciente. El sistema muestra dos listados con los momentos de seguimiento programados y los momentos de seguimiento no programados asociados a un paciente.
Precondiciones:	
Referencias	R5, R6
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El CIC indica ver el listado de momentos de seguimiento definidos para un paciente en el listado de pacientes.	2 El sistema busca el listado de momentos de seguimiento asociados a un paciente con fecha de llenado anterior o igual a la fecha actual. 3 El sistema muestra un listado con todos los momentos de seguimiento programados, especificando de

	cada uno el nombre, el plazo de tiempo para su llenado (si el estado de momento de seguimiento es atrasado el plazo de tiempo se será negativo y en rojo), la fecha de inicio, la cantidad de notas de monitoreo con estado de “Nuevas” o “Actualizadas” y se visualizarán las acciones que se pueden realizar con cada uno de ellos para cada uno de ellos (Ver caso de uso Listado de hojas). 4 El sistema muestra otro listado con todos los momentos de seguimiento no programados, especificando de cada uno el nombre, la fecha de inicio, la cantidad de notas de monitoreo con estado de “Nuevas” o “Actualizadas” y las acciones que se puede hacer con cada uno de ellos (Ver listado de hojas, o Eliminar el momento de seguimiento, en caso de estar atrasado). En este listado el CIC puede, además, seleccionar una de las definiciones de momento de seguimiento No Programado y así en este listado solo aparecerán los momentos de seguimiento que tienen como definición la seleccionada, y el sistema mostrará además la opción de adicionar un nuevo momento de seguimiento

	para esta definición. 5 El sistema comprueba si está generado el cronograma específico de ese paciente. 6 Si el cronograma específico está generado finaliza así el caso de uso.

Prototipo de Interfaz



Flujos Alternos

Acción del Actor	Respuesta del Sistema

	6a) Si no está generado el cronograma específico el sistema además verifica que el estado de todas las hojas de CRD asociadas a los momentos de seguimientos de la etapa “Evaluación” del paciente sean distintos de “No Iniciada”, si es así se muestra una sección para generar el cronograma específico para dicho paciente a partir de una fecha a indicar por el CIC y finaliza así el caso de uso.

Prototipo de Interfaz

Poscondiciones	

2.2.6.3. Visualizar listado de hojas.

Con este CU pasa igual que el anterior que ya existe en el sistema OpenClinica, pero será modificado porque la información que se muestra de las hojas de CRD no cumple con todas las necesidades del

CIM, por lo que se mostrará los datos que son realmente importantes para la conducción de EC en Cuba.

Caso de Uso:	Visualizar listado de hojas
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el CIC indica visualizar el listado de hojas de CRD asociadas a un momento de seguimiento. El sistema un listado con las hojas de CRD asociadas a un momento de seguimiento.
Precondiciones:	
Referencias	R9
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El CIC indica ver el Listado de hojas de CRD asociadas a un momento de seguimiento en el Listado de momentos de seguimiento definidos para un paciente.	2 El sistema busca todas las hojas de CRD asociadas al momento de seguimiento seleccionado. 3 El sistema muestra al CIC el Listado de hojas de CRD asociadas al momento de seguimiento seleccionado especificando de cada una el nombre, la versión, el estado, la fecha de inicio, la cantidad de notas de monitoreo con estado de "Nueva" o "Actualizada" y las acciones que se puede realizar con cada una de ellas, finalizando así el caso de uso.

Prototipo de Interfaz	
	
Poscondiciones	

2.2.6.4. Gestionar cronograma específico.

Este CU se le agregará al sistema para cumplir con las necesidades de la conducción de EC en Cuba.

Caso de Uso:	Gestionar cronograma específico
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el CIC indica: – Generar cronograma específico de un paciente a partir del cronograma general. – Adicionar un momento de seguimiento no programado. – Eliminar un momento de seguimiento no programado. De acuerdo a la opción seleccionada el sistema se encarga de generar el cronograma específico de un paciente, adicionar un nuevo momento de seguimiento no programado

	a la definición de un momento de seguimiento no programado o eliminar un momento de seguimiento a la definición de un momento de seguimiento no programado.
Precondiciones:	El cronograma general del estudio debe estar firmado por el Investigador Promotor.
Referencias	R1, R2, R3
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
El CIC indica realizar una de las siguientes opciones en el listado de momentos de seguimiento de un paciente: - Generar cronograma específico de un paciente a partir del cronograma general. - Adicionar un momento de seguimiento no programado. - Eliminar un momento de seguimiento no programado.	En dependencia de la acción del actor el sistema realiza una de las siguientes secciones.

Flujo Normal de Eventos	
Sección "Generar cronograma específico"	
Acción del Actor	**Respuesta del Sistema**
1 El CIC especifica una fecha e indica generar el cronograma específico de	2. El sistema le asocia al paciente todos los momentos de seguimiento

un paciente a partir del cronograma general.	programados definidos en el cronograma general para las etapas de “Tratamiento” y “Seguimiento”. 3. El sistema le establece una fecha de inicio a cada uno de estos momentos de seguimiento a partir de la fecha indicada por el CIC. 4. El sistema le establece un estado a cada uno de estos momentos de seguimiento (“No Iniciado”) y finaliza así el caso de uso.

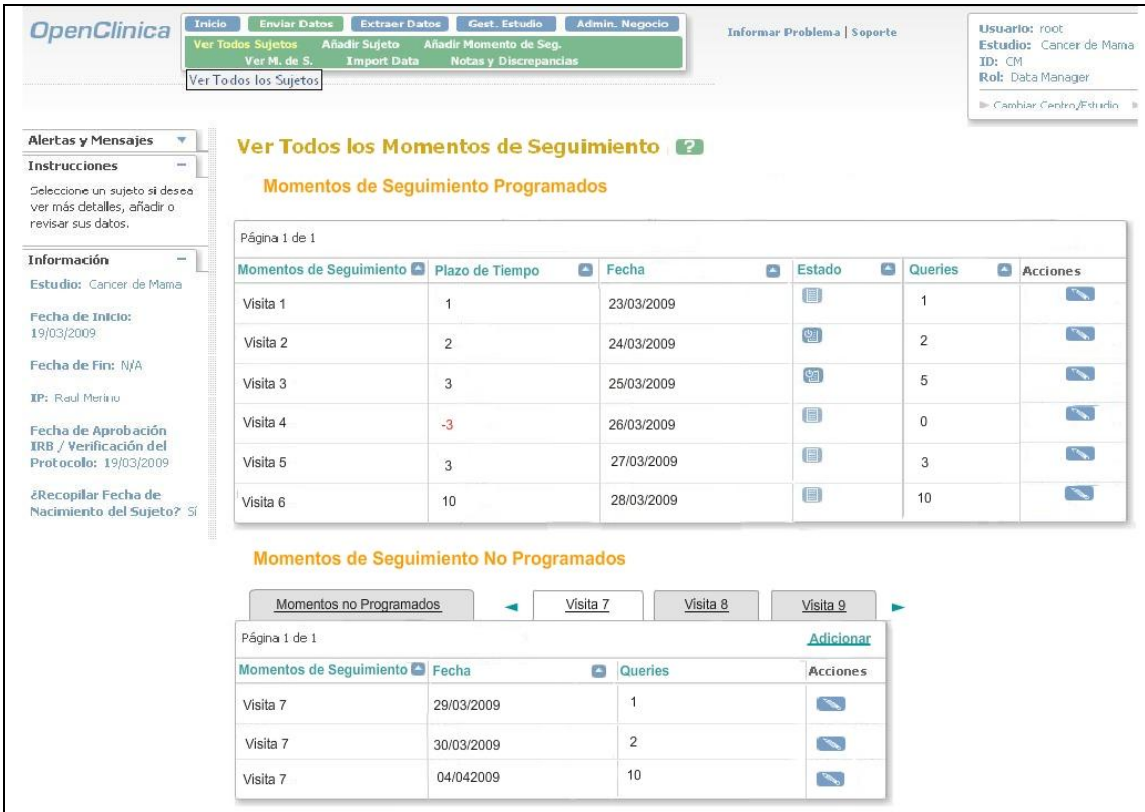
Prototipo de Interfaz

Flujo Normal de Eventos

Sección “Adicionar momento de seguimiento no programado”

Acción del Actor	Respuesta del Sistema
1. El CIC indica ver todas las instancias	2. El sistema busca todas las

asociadas a una definición de un momento de seguimiento no programado. 4. En el listado de las instancias de un momento de seguimiento no programado de un paciente el CIC indica adicionar una instancia a una definición de un momento de seguimiento no programado.	instancias asociadas a la definición de ese momento de seguimiento. 3. El sistema muestra todas las instancias asociadas a la definición de ese momento de seguimiento. 5. El sistema le asocia al paciente una nueva instancia de esta definición del momento de seguimiento no programado seleccionado, indicando como fecha de llenado la fecha actual. 6. El sistema muestra todas las instancias de esta definición del momento de seguimiento no programado asociadas al paciente, especificando de cada una la fecha de llenado, el estado, cantidad de notas de monitoreo con estado de "Nueva" o "Actualizada" y visualiza las posibles acciones a realizar por el CIC.
Prototipo de Interfaz	



Poscondiciones	Se le asocia a un paciente una planificación de momentos de seguimiento.

Flujo Normal de Eventos

Sección "Eliminar momento de seguimiento no programado"

Acción del Actor	Respuesta del Sistema
1. El CIC indica ver todas las instancias asociadas a una definición de un momento de seguimiento no programado.	2. El sistema busca todas las instancias asociadas a la definición de ese momento de seguimiento. 3. El sistema verifica el estado de las instancias de la definición de ese momento de seguimiento no programado.

5. En el listado de las instancias de un momento de seguimiento no programado de un paciente el CIC indica eliminar una instancia a una definición de un momento de seguimiento no programado.	4. El sistema muestra todas las instancias asociadas a la definición de ese momento de seguimiento y un botón “Eliminar” si el estado de la instancia es “Atrasado”. 6. El sistema elimina la instancia a la definición del momento de seguimiento no programado seleccionado. 7. El sistema muestra todas las instancias de la definición del momento de seguimiento no programado asociadas al paciente, especificando de cada una la fecha de llenado, el estado, cantidad de notas de monitoreo con estado de “Nueva” o “Actualizada” y las acciones a realizar por el CIC, finalizando así el caso de uso.

2.3. Conclusiones

Con el desarrollo en este capítulo del modelo del dominio, de la descripción de los requisitos funcionales, los no funcionales, del diagrama de CU del sistema y de la descripción de CU del sistema se describe el funcionamiento del sistema, cómo debe hacerlo y las características necesarias para su correcto funcionamiento. Se identificaron 20 requisitos funcionales, los que se agruparon en 7 casos de uso.

CAPÍTULO 3: Diseño del Sistema

Al realizar el diseño de la Gestión de Datos de Ensayos Clínicos (GDEC) y de esta forma para obtener una solución bastante óptima surgen algunos elementos importantes, como los diagramas de clases del diseño, los diagramas de interacción, el modelo de datos, el diagrama de despliegue, la descripción del patrón arquitectónico adoptado y los patrones de diseño usados. Estos elementos serán descritos en el desarrollo de este capítulo.

3.1. Descripción de la Arquitectura

De acuerdo con la IEEE la arquitectura es:

- ✓ El nivel conceptual más alto de un sistema en su ambiente.
- ✓ La organización fundamental de un sistema descrita en:
  - Sus componentes.
  - Relación entre ellos y con el ambiente.
  - Principios que guían su diseño y evolución[13].

3.1.1. Patrón arquitectónico: Modelo Vista Controlador

La arquitectura envuelve un conjunto de decisiones estratégicas de diseño, lineamientos, reglas y patrones que restringen el diseño y la implementación de un software. El diseño arquitectónico define la relación entre los elementos estructurales principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. OpenClinica ha sido desarrollada bajo la utilización del patrón arquitectónico Modelo Vista Controlador.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web. Para aplicaciones J2EE, la arquitectura MVC

satisface las necesidades de la aplicación. El patrón indica que se deben establecer tres componentes o capas en la arquitectura, y que cada una de estas, solo se comunica con la adyacente.

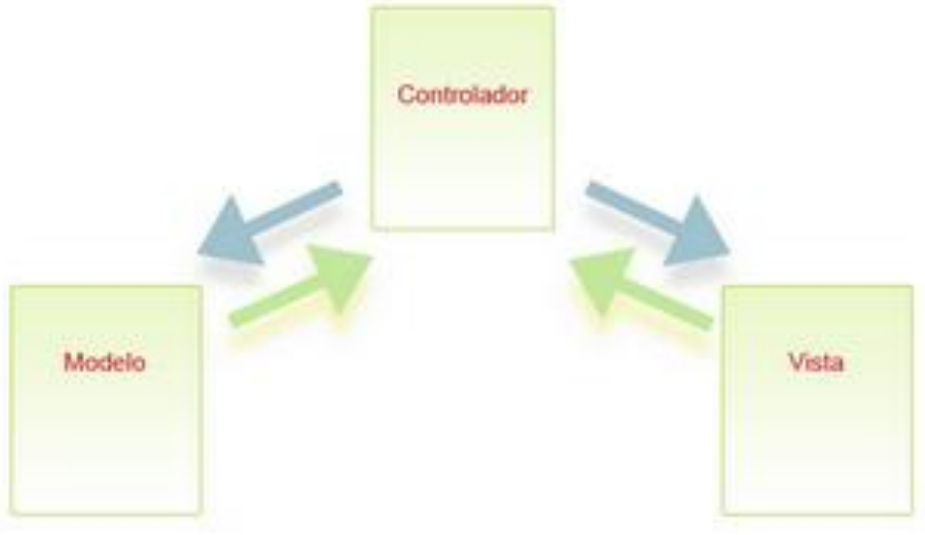


Fig. # 3 Patrón Modelo Vista Controlador.

Descripción de los componentes

- ✓ Modelo: Esta es la representación específica de la información con la cual el sistema opera. Básicamente contiene la lógica de negocio real, el dominio de la aplicación con sus clases, los métodos get y set y los objetos de acceso a datos (DAO) que implementen las operaciones CRUD. Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

- ✓ Vista: Este presenta el modelo en un formato adecuado para interactuar con el usuario. Responsable de la lógica de presentación y captura de datos de nuestro sistema. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

- ✓ Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Reciben las entradas, como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio (requests) para el modelo o la vista, traslada las peticiones

de la Capa Vista a la Capa Modelo, y según la respuesta, la reenvía o no a la Capa Vista. Carga objetos y opera con ellos.

Entre sus ventajas se puede señalar que MVC nos aporta una construcción de software muy mantenible, en la que se pueden localizar de forma ágil los errores. Supone un diseño modular, y muy poco acoplado, favoreciendo la reutilización. El uso de este patrón trae asociado directamente un grupo de patrones de diseño como son el patrón Bajo Acoplamiento y el patrón Alta Cohesión.

Al igual que la mayoría de las aplicaciones web actuales, el sistema OpenClinica fue desarrollado haciendo uso de este patrón arquitectónico y como el sistema alasClínicas será desarrollado a partir del sistema OpenClinica, se mantiene MVC como patrón arquitectónico, esto queda claramente evidenciado y se muestra posteriormente en la organización de los diagramas de clases del diseño y los diagramas de componentes, donde se agrupan todos los elementos en los tres paquetes principales que propone este patrón.

3.1.2. Patrones de Diseño

Refiriéndose a los patrones de diseño Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides mencionaron: “Un patrón de diseño nomina, abstrae e identifica los aspectos claves de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuando aplicarlo y si tiene sentido hacerlo teniendo otras restricciones de diseño, así como las consecuencias y las ventajas e inconvenientes de su uso. Por otro lado, como normalmente tendremos que implementar nuestros diseños, un patrón también proporciona código de ejemplo para ilustrar una implementación.”

Los patrones pretenden ser la solución al problema de la comunicación de experiencias entre profesionales del software. Cada patrón describe un problema concurrente en nuestro entorno, para describir después el camino a la solución a ese problema, de tal forma que pueda ser reutilizado en proyectos distintos**.** Un Patrón de diseño proporciona un esquema para refinar los subsistemas o componentes de un sistema software y las relaciones entre ellos. Describe estructuras recurrentes de comunicar componentes que resuelven un problema de diseño en un contexto particular. Son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema.

El Patrón Objeto de Acceso a Datos, **DAO** (de sus siglas en inglés Data Access Object) surge históricamente de la necesidad de gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento[14].

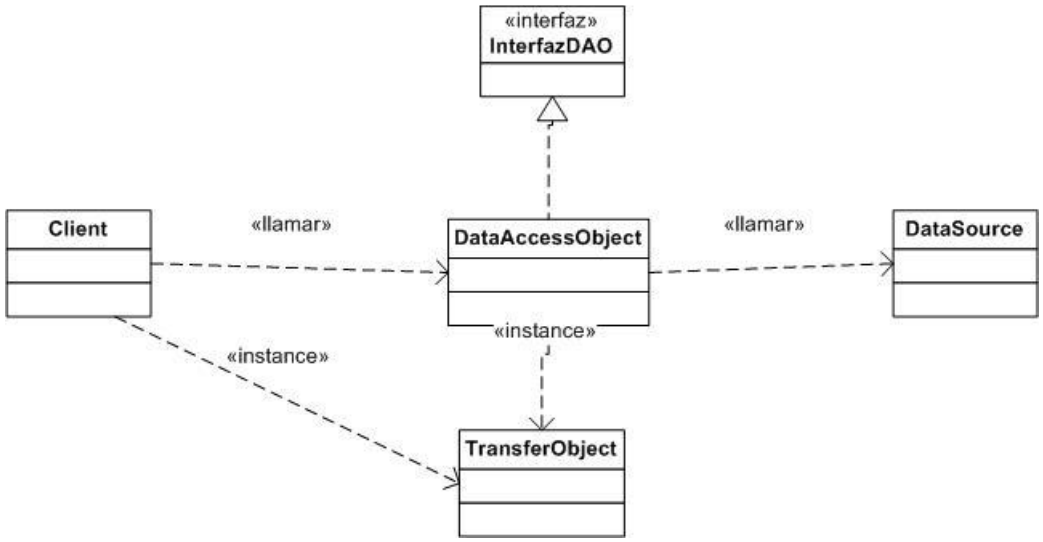


Fig. # 4 Estructura del DAO.

No es imprescindible, pero en proyectos de cierta complejidad resulta útil que el DAO implemente un interfaz. De esta forma los objetos cliente tienen una forma unificada de acceder a los DAO.

El uso de este patrón queda expuesto en los diagramas de clases del diseño y diagramas de componentes que se muestran más adelante en este documento. En cada diagrama se incluye en el paquete Modelo un grupo de clases DAO, un ejemplo constituye el siguiente perteneciente al Caso de Uso Mostrar listado de pacientes.

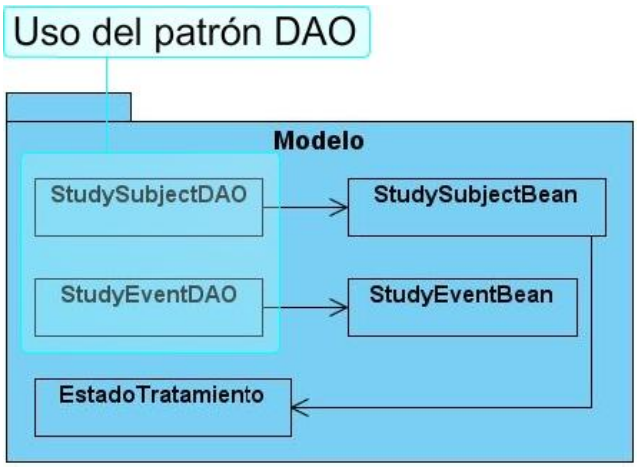


Fig. # 5 Uso del Patrón DAO.

Con la aparición del J2EE, todo un nuevo catálogo de patrones de diseño apareció. Desde que J2EE es una arquitectura por si misma que involucra otras arquitecturas, incluyendo Servlets, JSPs, EJBs, y más, merece su propio conjunto de patrones específicos para diferentes aplicaciones empresariales.

Método Plantilla *(Template Method):* Es un patrón de diseño que define una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Este patrón define una serie de pasos, en donde los pasos serán redefinidos en las subclases. Usando el Método Plantilla, se define una estructura de herencia en la cual la superclase sirve de plantilla de los métodos en las subclases. Una de las ventajas de este método es que evita la repetición de código, por tanto la aparición de errores. El sistema OpenClinica hace uso del mismo a través de la clase abstracta SecureController.java, todas las clases servlet que invocan cambios en el modelo o en la vista heredan de esta superclase y cada una se encarga de redefinir sus métodos.

Vista compuesta *(Composite View)*: Es un patrón de diseño en el cual un objeto vista está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras JSP usando la directiva include o el action include es un ejemplo de la aplicación del patrón Composite View.

Iterador *(Iterator)***:** Es otro patrón que se encarga de proveer una forma de acceso secuencial a los elementos de un objeto agregado sin tener que exponer su representación interna. El mismo define una interfaz mediante la cual se ofrece este acceso secuencial a un agregado y soportar múltiples recorridos de objetos agregados. En la implementación se usa mucho este patrón para recorrer los

ArrayList los cuales contienen en algunos casos las tuplas que se obtienen de las consultas a la base de datos. En las clases DAO contenidas en el paquete Modelo en los diagramas de clases del diseño y los diagramas de componentes se hace uso de este patrón. Estas clases son las encargadas de buscar la información almacenada en la base de datos. En ocasiones una petición devuelve por resultado varias tuplas de una entidad y es Iterator el responsable de asimilar todas las filas en un arreglo.

Bajo acoplamiento: El acoplamiento mide qué tan fuerte está una clase conectada con otras. Una clase con bajo o débil acoplamiento no depende de muchas otras clases. Una clase con alto o fuerte acoplamiento recurre a muchas otras clases. Este tipo de clase no es conveniente, pues: cambios en las clases relacionadas ocasionan cambios en la clase local; son más difíciles de entender; son más difíciles de reutilizar. Por lo que la solución planteada por este patrón es asignar una responsabilidad para mantener bajo acoplamiento.

Alta cohesión: Surge ante la problemática de cómo mantener la complejidad dentro de límites manejables. En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos. La solución es asignar una responsabilidad de modo que la cohesión siga siendo alta.

Experto: Consiste en asignar la responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Un ejemplo del uso de este patrón se evidencia en la clase EventCRFBean en el método getStage. En este método se utiliza la información de la base de datos y a través de la clase DataEntryStage se convierte el calor del estado del cada hoja de CRD, que es un entero, a un objeto de tipo DataEntryStage, y en este caso el responsable de hacer esta tarea es el experto en información EventCRFBean.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea muy común). La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Logrando así mayor reutilización. Ante la problemática ¿quién debería ser el responsable de la creación de una nueva instancia de alguna clase? El uso de este patrón se puede encontrar en todos los diagramas de clases del diseño y diagramas de componentes realizados. Un

ejemplo del mismo se aprecia entre las clases DAO y Bean, donde las clases DAO son las encargadas de la creación de objetos Bean.

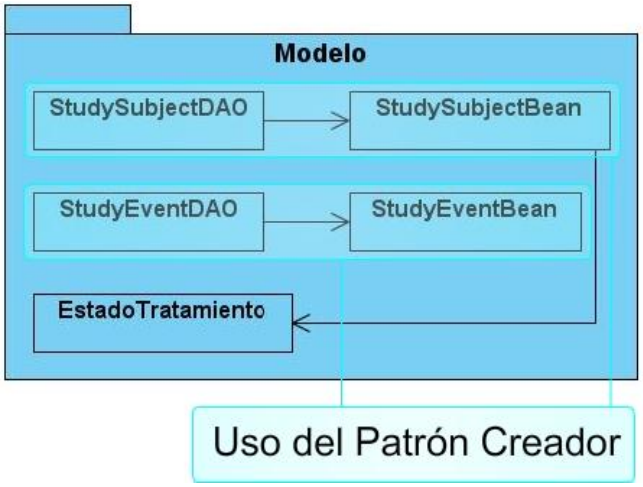


Fig. # 6 Uso del Patrón Creador.

Controlador: Se encarga de 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. Las clases Servlet son clases controladoras que se encargan de gestionar todos los eventos del sistema. Ante cualquier petición gestionan la información relacionada a la misma a través de las clases del paquete modelo, procesan esta petición y redireccionan a una página JSP la respuesta a esta solicitud. El uso del patrón Controlador se evidencia en todas las clases Servlet que conforman los diagramas de clases del diseño y diagramas de componentes.

3.2. Modelo de Diseño

3.2.1. Diagramas de clases de diseño

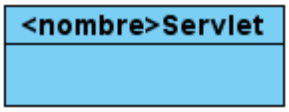
Los diagramas de clases se realizan por cada caso de uso y muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. A continuación se muestran los diagramas de clases del diseño confeccionados, se han agrupado las clases de acuerdo a su funcionalidad en tres grandes paquetes, haciendo además referencia al patrón arquitectónico expuesto anteriormente: Modelo Vista Controlador (MVC). Se muestran entonces los elementos que componen cada uno de estos paquetes y posteriormente se ilustran con los diagramas de clases del diseño realizados.

Paquetes:

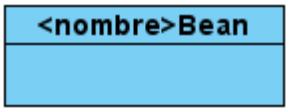
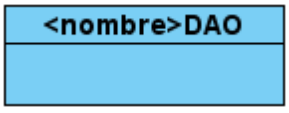
- ✓ Vista:

 *.jsp	 *.html	 form
Las páginas servidoras *.jsp son las encargadas de generar una página cliente *.html a partir de su código y el código html que traen incrustado.	La página cliente *.html es la interfaz a través de la cual interactúan los usuarios con el sistema.	A través de él se envían datos del cliente al servidor.

- ✓ Control:

	Clases controladoras que se encargan de la gestión de los datos del modelo y de enviárselos a la vista, de la misma forma, recibe los datos de la vista y los envía el modelo para ser almacenados.

- ✓ Modelo

	Clases entidades portadoras de los datos del sistema, cada objeto de ellas constituye una tupla de una tabla en la base de datos.
	Clases de acceso a datos, encapsulan la fuente de datos y ocultan la forma de acceder a ellos.

- ✓ Dependencias

	Paquete que contiene código utilizado para validar la información y construir la vista a mostrar en las páginas *.html mostradas a los usuarios.

	Grupo de páginas servidoras de las que dependen algunas *.jsp desarrolladas para crear la vista que se mostrará en las páginas *.html.
	Ficheros encargados de internacionalización de los textos que se muestran en las páginas *.html.
	Conjunto de API's de Java de las que depende un grupo de clases.

3.3.1.1. Visualizar listado de pacientes.

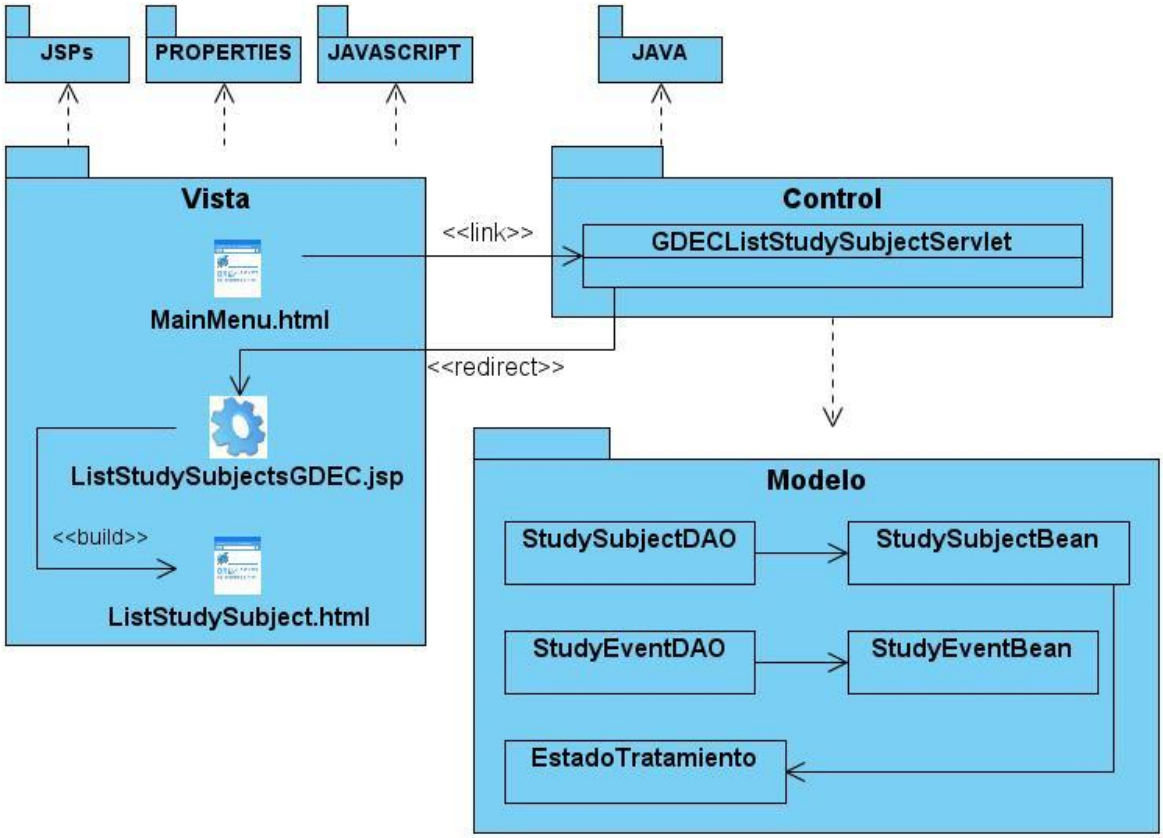


Fig. # 7 Diagrama de clases de diseño "Visualizar listado de Pacientes".

Caso de Uso: Visualizar listado de pacientes

Nombre: GDECListStudySubjectsSubmitServlet		
Tipo: Clase (Es la encargada de gestionar el listado de pacientes que se muestra al usuario, en el cual se visualiza el estado de inclusión, el estado de tratamiento, la próxima visita y la última visita asociada a cada paciente además de las acciones a realizar de acuerdo al rol que haya iniciado sesión).		
Atributo	**Tipo**	
Responsabilidades		
Nombre	**Tipo**	**Descripción**
myProceed()	void	Método que se encarga de verificar los roles para permitir o no el envío de datos.
processRequest()	void	Método que se encarga de gestionar los datos a mostrar en el listado de Pacientes a partir del id de un estudio.

3.3.1.2. Visualizar listado de momentos de seguimiento

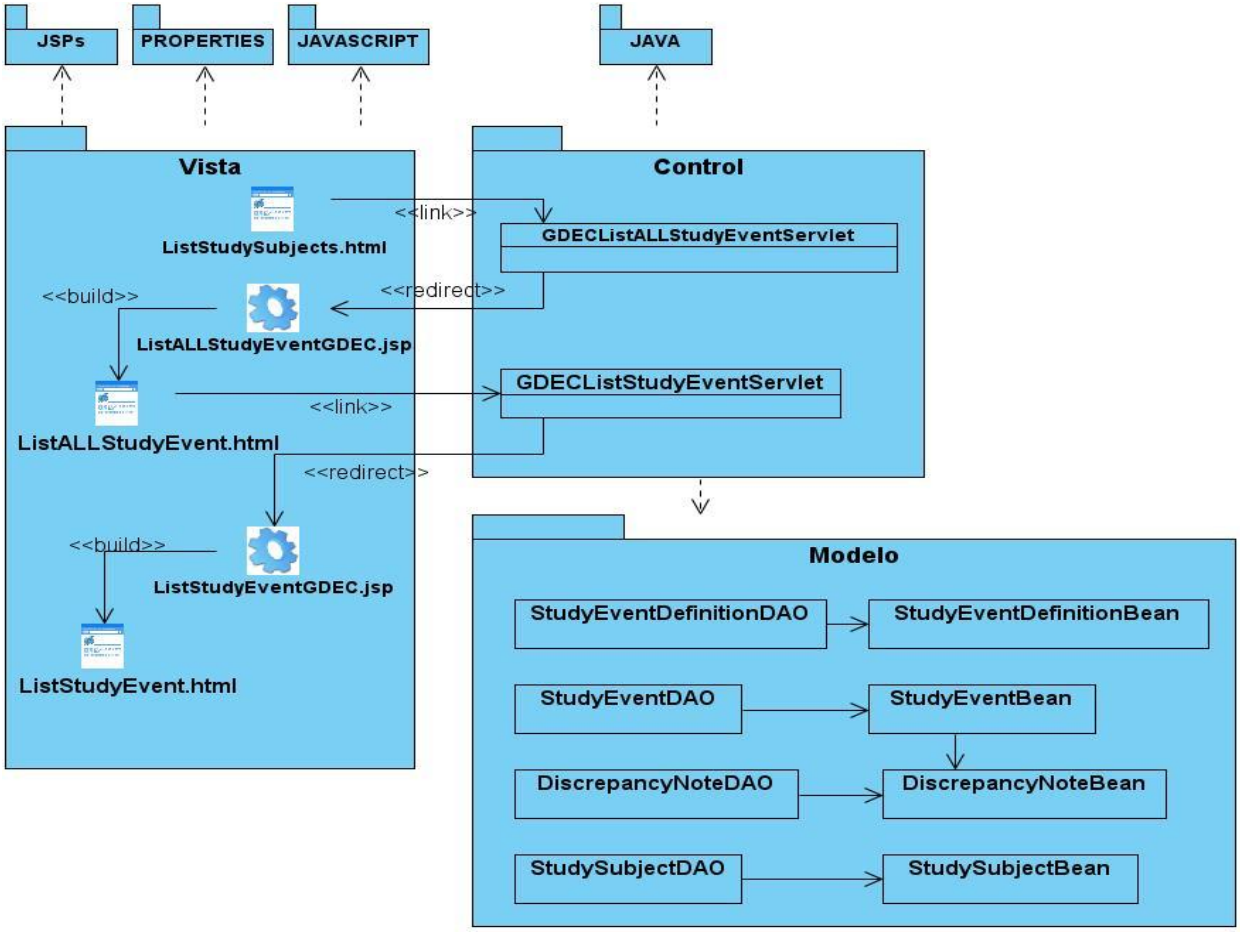


Fig. # 8 Diagrama de clases de diseño "Visualizar listado de momentos de seguimiento".

Caso de Uso: Visualizar listado de momentos de seguimiento		
Nombre: GDECListALLStudyEventServlet		
Tipo: Clase (Es la encargada de gestionar el listado de momentos de seguimiento asociados a un paciente que se muestra al usuario, en el cual se visualiza el estado, la fecha y plazo de llenado, la cantidad de notas de monitoreo con estado "Nueva" o "Actualizada" asociadas a cada momento de seguimiento además de las acciones a realizar de acuerdo al rol que haya iniciado sesión).		
Atributo	**Tipo**	
Responsabilidades		
Nombre	**Tipo**	**Descripción**
myProceed()	void	Método que se encarga de verificar los roles para

		permitir o no el envío de datos.
processRequest()	void	Método que se encarga de gestionar los datos a mostrar en el listado de momentos de seguimiento asociados a un paciente. Este listado estará compuesto realmente por 2 listados, el de los programados y el de los no programados.

Nombre: GDECListStudyEventServlet	
Tipo: Clase (Es la encargada de gestionar el listado de momentos de seguimiento asociados a un paciente que se muestra al usuario, en el cual se visualiza el estado, la fecha y plazo de llenado, la cantidad de notas de monitoreo con estado "Nueva" o "Actualizada" asociadas a cada momento de seguimiento además de las acciones a realizar de acuerdo al rol que haya iniciado sesión).	
Atributo	**Tipo**
Responsabilidades	

Nombre	**Tipo**	**Descripción**
mayProceed()	void	Método que se encarga de verificar los roles para permitir o no el envío de datos.
processRequest()	void	Método que se encarga de gestionar los datos a mostrar en el listado de momentos de seguimiento asociados a un paciente. Este listado estará compuesto realmente por 2 listados, el de los programados y el de los no programados que pertenecen a una definición especifica.

3.3.1.3. Visualizar listado de hojas de CRD

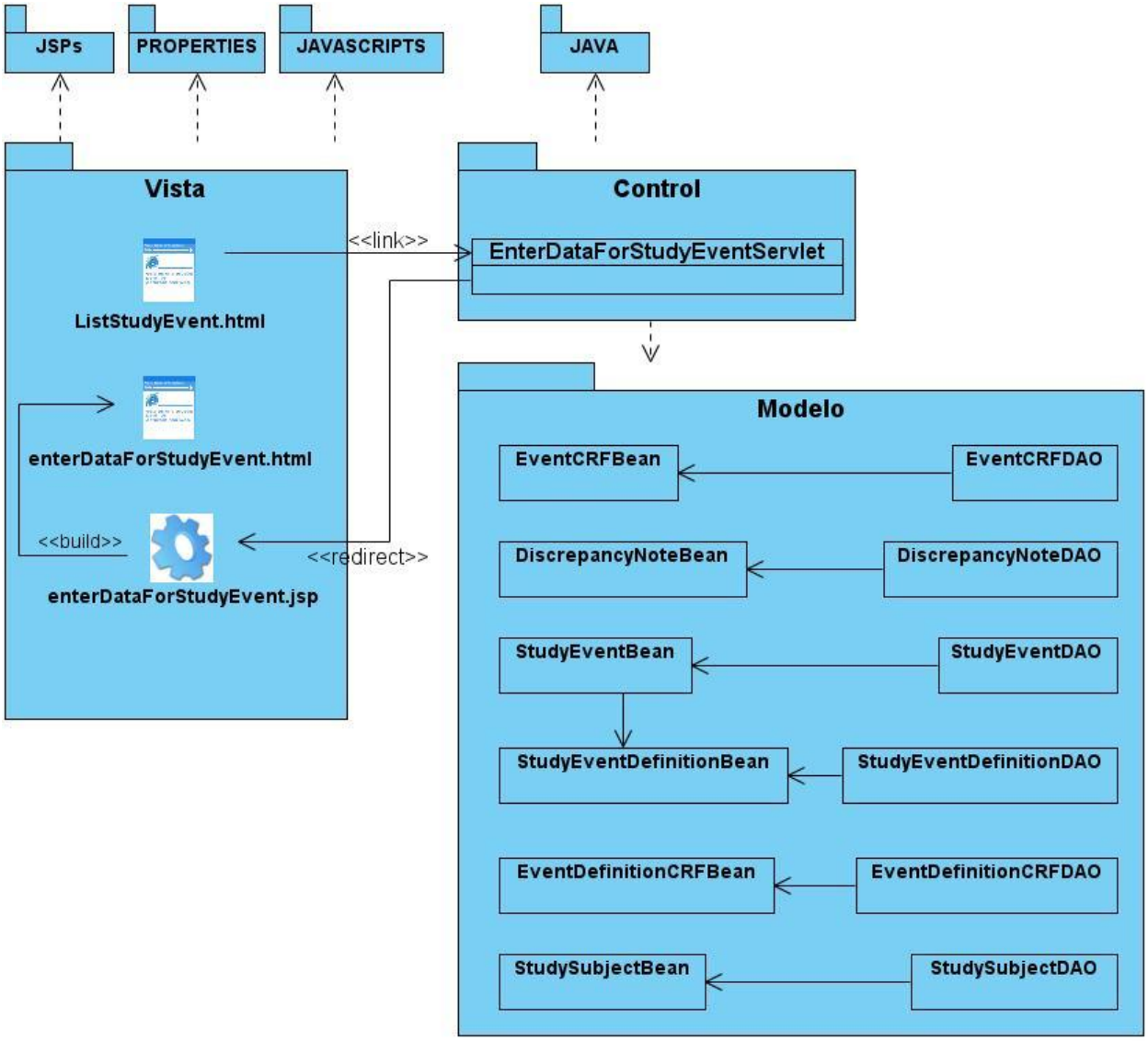


Fig. # 9 Diagrama de clases de diseño "Visualizar listado de hojas de CRD".

Caso de Uso: Visualizar listado de hojas de CRD
Nombre: GDECListEventCRDServlet
Tipo: Clase (Es la encargada de gestionar el listado de hojas de CRD asociadas a un momento de seguimiento que se muestra al usuario, en el que se visualiza el estado, la fecha, plazo de llenado, la fecha de última modificación y la cantidad de notas de monitoreo con estado "Nueva" o "Actualizada" asociadas a cada hoja de CRD además de las acciones a realizar de acuerdo al rol que haya iniciado sesión).

Atributo		**Tipo**
Responsabilidades		
Nombre	**Tipo**	**Descripción**
myProceed()	void	Método que se encarga de verificar los roles para permitir o no el envío de datos.
processRequest()	void	Método que se encarga de gestionar los datos a mostrar en el listado de hojas de CRD asociadas a un momento de seguimiento específico.
getStudyEvent	StudyEventBean	Método que se encarga de buscar un momento de seguimiento a partir de su identificador.
getUncompletedCRFs populateUncompletedCRFsWithAnOwner populateUncompletedCRFsWithCRFAndVersions getDisplayEventCRFs getDisplayEventCRFs (distintos parámetros)	ArrayList void void ArrayList ArrayList	Métodos que se encargan de gestionar un listado con todas las hojas de CRD asociadas al momento de seguimiento de un paciente.

3.3.1.4. Gestionar cronograma específico

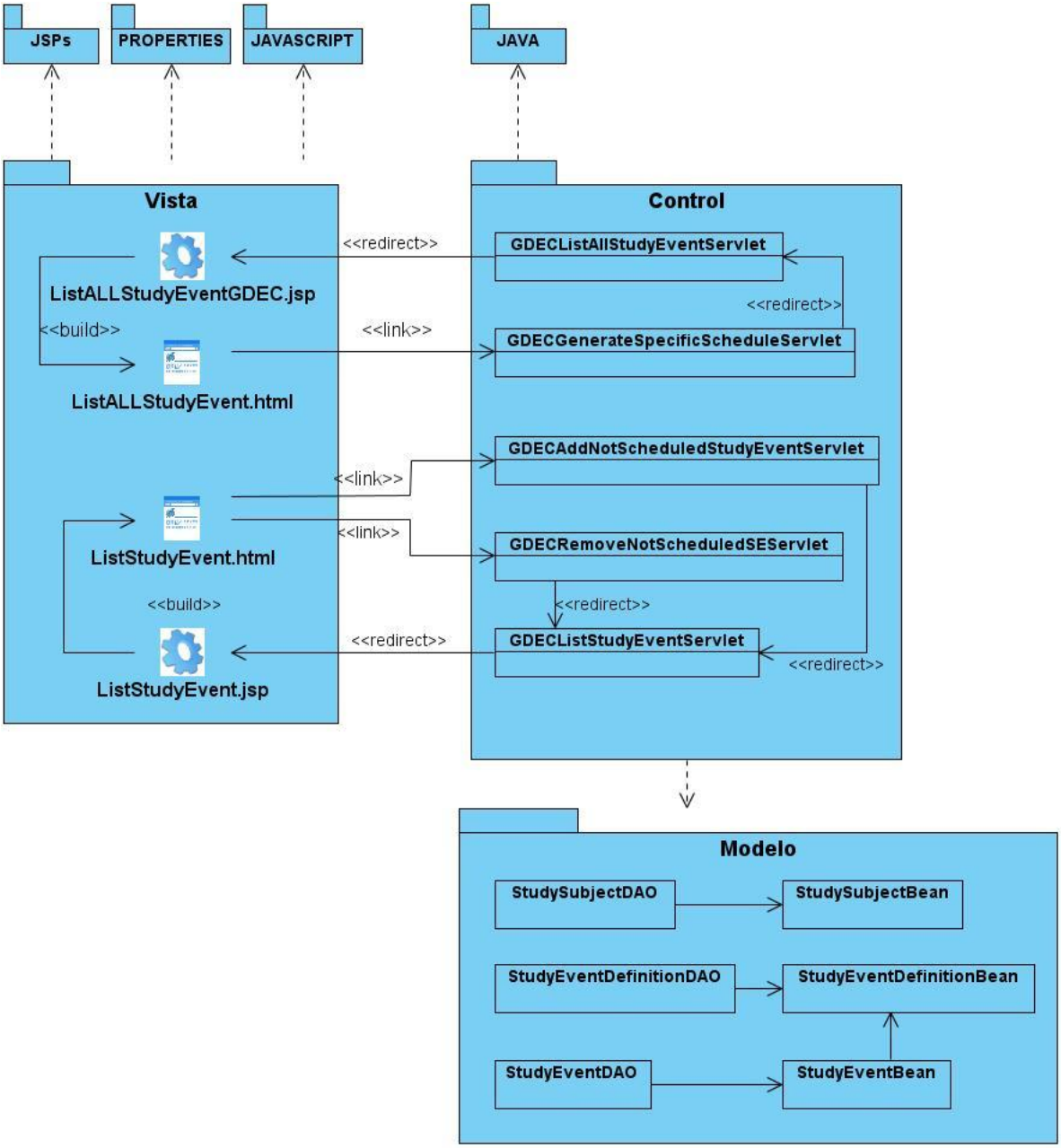


Fig. # 10 Diagrama de clases de diseño "Gestionar cronograma específico".

Caso de Uso: Gestionar cronograma específico
Nombre: GDECGenerateSpecificScheduledServlet
Tipo: Clase (Es la encargada de generar el cronograma específico de un paciente).

Atributo	**Tipo**	
Responsabilidades		
Nombre	**Tipo**	**Descripción**
myProceed()	void	Método que se encarga de verificar los roles para permitir o no el envío de datos.
processRequest()	void	Método encargado de planificar para un paciente especifico todos de momentos de seguimiento programados definidos en el cronograma general del estudio pertenecientes a las etapas “Tratamiento” y “Seguimiento”. La fecha de inicio y fin de cada uno, así como el plazo de llenado son generados a partir de la definición de cada momento de seguimiento.
Nombre: GDECAddNotScheduledStudyEventServlet		
Tipo: Clase (Es la encargada de adicionar instancias de una definición de un momento de seguimiento no programados a un paciente).		
Atributo	**Tipo**	
Responsabilidades		
Nombre	**Tipo**	**Descripción**
myProceed()	void	Método que se encarga de verificar los roles para permitir o no el envío de datos.
processRequest()	void	Método que se encarga de adicionar una instancia de una definición de un momento de seguimiento no programado a un paciente a partir del identificador del paciente y de la definición del momento de seguimiento no programado, asignándole como fecha de llenado la fecha actual.
Nombre: GDECRemoveNotScheduledStudyEventServlet		
Tipo: Clase (Es la encargada de eliminar instancias de una definición de un momento de seguimiento no programados de a un paciente).		
Atributo	**Tipo**	
Responsabilidades		

Nombre	Tipo	Descripción
myProceed()	void	Método que se encarga de verificar los roles para permitir o no el envío de datos.
processRequest()	void	Método que se encarga de eliminar una instancia de una definición de un momento de seguimiento no programado a un paciente a partir de los identificadores del paciente y de la definición del momento de seguimiento no programado.

3.2.2. Diagramas de interacción (Secuencia)

Los diagramas de secuencia son unos de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Ellos muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y contienen detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos. Se desarrollan tras examinar la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario.

3.2.2.1. Visualizar listado de pacientes.

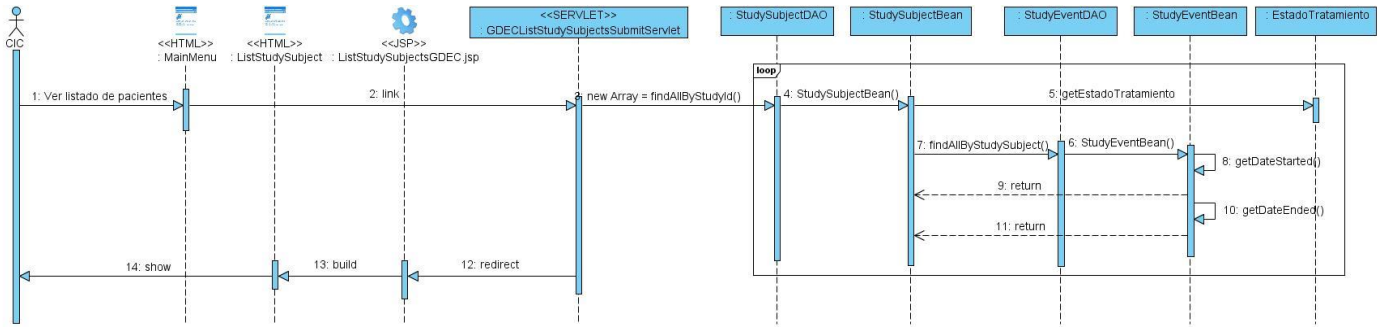


Fig. # 11 Diagrama de secuencia "Visualizar listado de pacientes".

3.2.2.2. Visualizar listado de momentos de seguimiento.

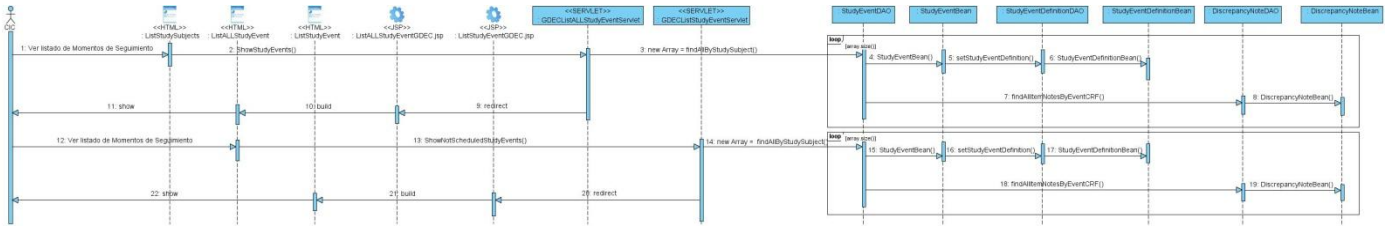


Fig. # 12 Diagrama de secuencia "Visualizar listado de momentos de seguimiento".

3.2.2.3. Visualizar listado de hojas de CRD.

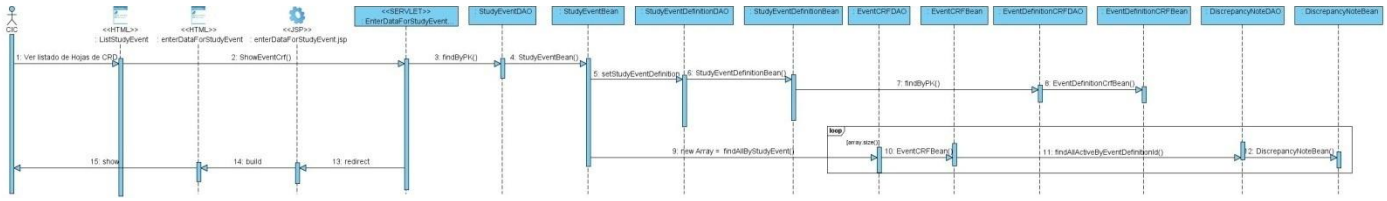


Fig. # 13 Diagrama de secuencia "Visualizar listado de hojas de CRD".

3.2.2.4. Gestionar cronograma específico. Sección "Generar cronograma específico".

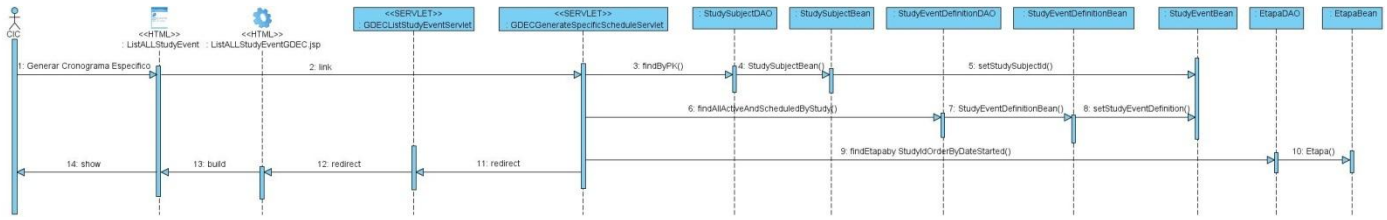


Fig. # 14 Diagrama de secuencia "Gestionar cronograma específico" sección "Generar cronograma específico".

3.2.2.5. Gestionar cronograma específico. Sección "Adicionar momento de seguimiento no programado".

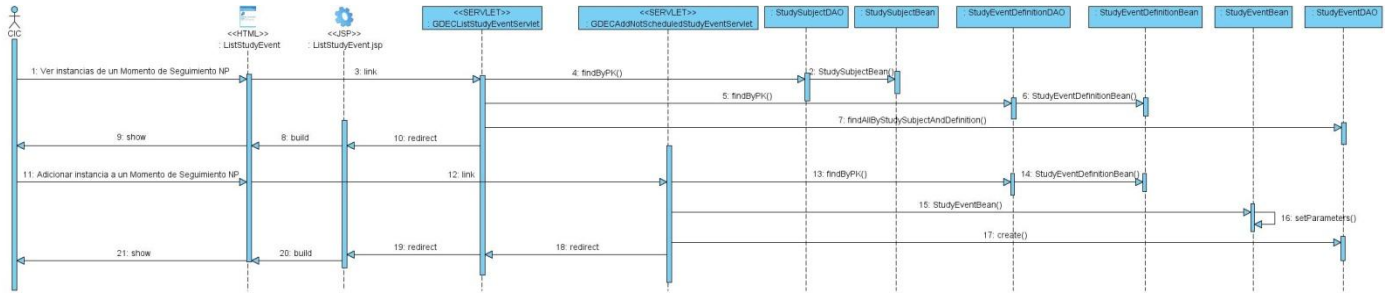


Fig. # 15 Diagrama de secuencia "Gestionar cronograma específico" sección "Adicionar momento de seguimiento no programado".

3.2.2.6. "Gestionar cronograma especifico". Sección "Eliminar momento de seguimiento no programado".

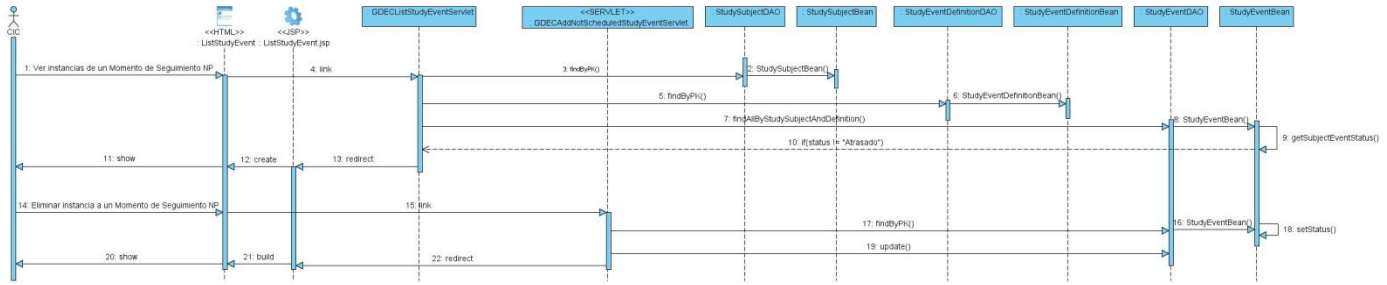


Fig. # 16 Diagrama de secuencia "Gestionar cronograma específico" sección "Eliminar momento de seguimiento no programado"

3.3. Modelo de Datos

Un modelo de datos es un conjunto de conceptos que describe la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. El modelo de datos aporta la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información.

A continuación se muestra el Modelo de Datos para la Gestión de Datos de Ensayos Clínicos en el sistema ***alasClínicas***. El mismo cuenta con 19 entidades agrupadas en 5 conjuntos de acuerdo a sus datos y una referencia a la tabla cronograma:

- ✓ Pacientes

- ✓ Pacientes – Momentos de Seguimiento (Estados)
- ✓ Momentos de Seguimiento
- ✓ Momentos de Seguimiento – Notas (Datos)
- ✓ Notas

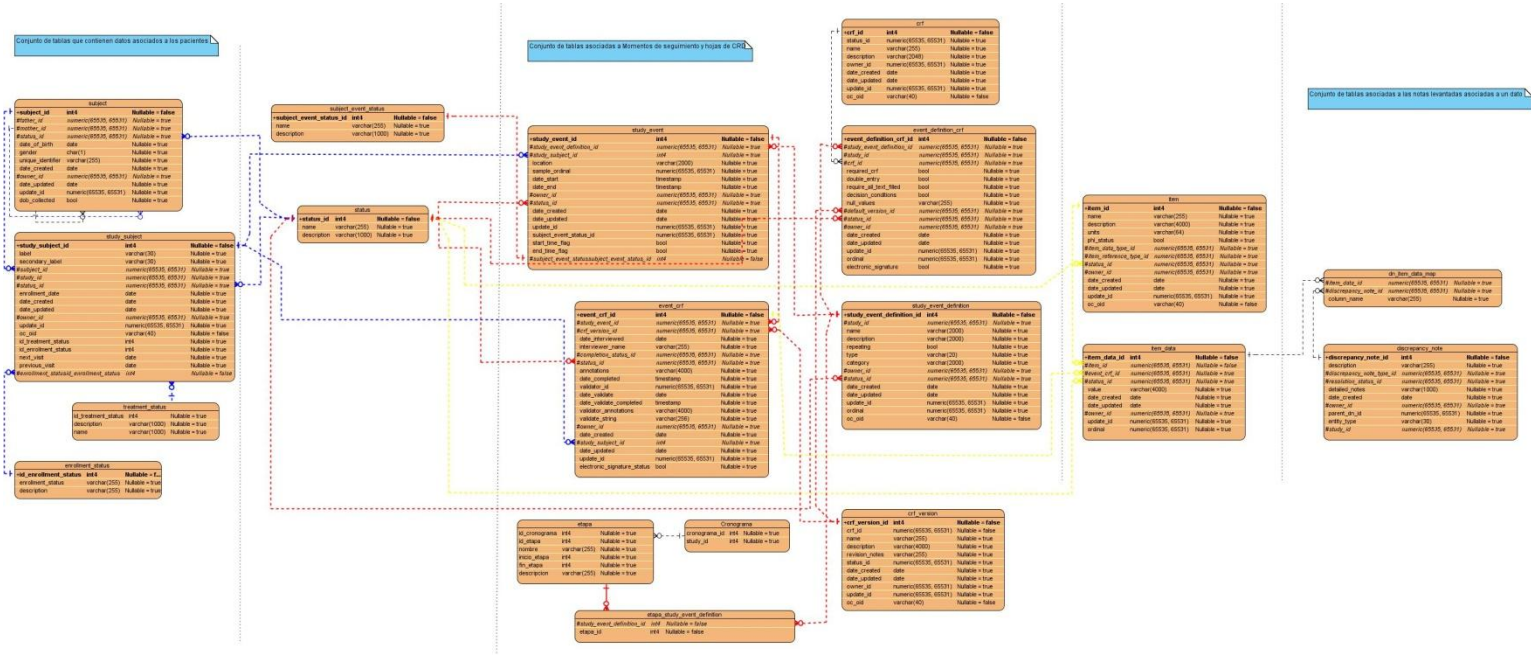


Fig. # 17 Modelo de Datos.

A continuación se muestra para cada una de las entidades que sustentan la GDEC su nombre, llave primaria (PK de sus siglas en inglés primary key), tipo de dato de la llave primaria, número de campos que contiene dicha entidad y una breve descripción.

No.	Entidad	Llave primaria	Tipo de dato	Número de campos	Descripción de Entidad
1	subject	subject_id	integer	12	Entidad que define todos los sujetos asociados al estudio, dígase “padre de sujeto”, “madre de sujeto” y “sujeto” en si mismo.

2	study_subject	study_subject_id	integer	17	Entidad que contiene todos los “sujetos” que se deciden estudiar en un ensayo y pasan a ser “pacientes”.
3	treatment_status	treatment_status_id	integer	3	Entidad que contiene los tipos de “estados” de acuerdo a la “etapa de tratamiento” que puede asumir un “paciente”.
4	enrollment_status	enrollment_status_id	integer	3	Entidad que contiene los tipos de “estados” de acuerdo a su “inclusión” que puede asumir un “paciente”.
5	subject_event_status	subject_event_status_id	integer	3	Entidad que contiene los diferentes estados que pueden asumir los “momentos de seguimiento”.
6	status	status_id	integer	3	Entidad que contiene los tipos de “estados” definidos para “pacientes”, “momentos de seguimientos” y “hojas de crd”.
7	study_event	study_event_id	integer	16	Entidad que contiene los “momentos de seguimiento” asociados a un paciente.
8	event_crf	event_crf_id	integer	20	Entidad que contiene las “hojas de crd” asociadas a un “momento de seguimiento”.
9	crf	crf_id	integer	9	Entidad que contiene las “hojas de crd” existentes el estudio.

10	event_definition_crf	event_definition_crf_id	integer	17	Entidad que asocia las entidades "study_event_definition" con "crf" donde se asignan varios "CRFs" a varias "definiciones de momentos de seguimiento".
11	study_event_definition	study_event_definition_id	integer	14	Entidad que contiene las "definiciones de momentos de seguimientos", es decir, contiene todos los momentos de seguimientos que se le pueden asociar a un paciente.
12	crf_version	crf_version_id	integer	11	Entidad que contiene las "versiones" definidas para una "hoja de crd".
13	item	item_id	integer	13	Entidad que contiene las "variables" existentes en una "hoja de crd".
14	item_data	item_data_id	integer	10	Entidad que contiene los valores asociados a las "variables" de una "hoja de crd".
15	discrpancy_note	discrepancy_note_id	integer	10	Entidad que contiene las "notas de discrepancia" existentes.
16	dn_item_data_map	-		3	Entidad que asocia las "variables" a las "notas de discrepancia" existentes.
17	etapa	etapa_id	integer	6	Entidad que contiene los nombres, días de inicio y fin de las "etapas"

					asociadas a un “cronograma general”.
18	etapa_study_event_definition	etapa_id study_event_definition_id	integer integer	2	Entidad que asocia las “definiciones de momentos de seguimiento” existente a una “etapa” definida en el “cronograma general”.

*Nota: La entidad “cronograma” se encuentra representada también en el Modelo de Datos debido a que de ella depende un grupo de funcionalidades para la “GDEC” aunque pertenece al módulo “Cronograma General”, su uso y descripción quedan evidenciadas en el módulo que la define.

3.4. Modelo de Despliegue

El Modelo de Despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto.

A continuación se muestra el modelo de despliegue para el sistema ***alasClínicas***:

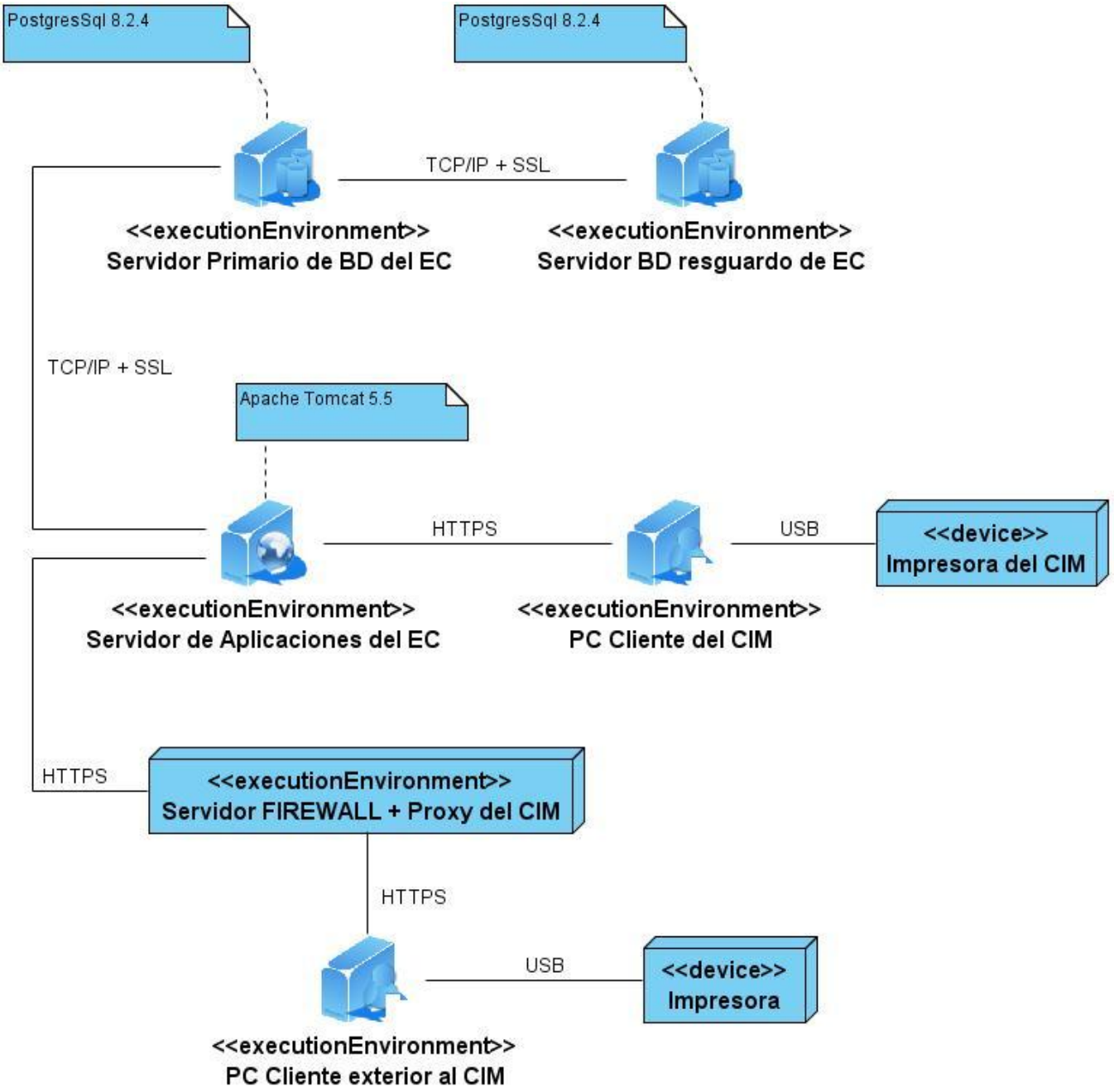


Fig. # 18 Diagrama de Despliegue.

3.5. Conclusiones

Con el desarrollo de este capítulo queda expuesta la arquitectura definida para el sistema así como los patrones de arquitectura y diseño a utilizar. Mediante los diagramas de clases y de interacción quedan

expuestas las clases asociadas a cada caso de uso y la manera en la que estas interactúan entre sí. Se muestra el sistema desde sus cimientos (Modelo de datos) hasta el despliegue del sistema en ejecución (Modelo de despliegue). Con la vista del más alto nivel conceptual ofrecida por la arquitectura y el menor nivel de abstracción y más detallado ofrecido por el diseño, se da guía y paso a la implementación.

CAPÍTULO 4: Implementación del Sistema

El propósito de la implementación es:

- ✓ Implementar clases y objetos en términos de componentes (archivos fuente, binarios, ejecutables y otros).
- ✓ Probar los componentes desarrollados como unidades.
- ✓ Integrar los resultados producidos por los desarrolladores individuales en un sistema ejecutable.

El sistema se desarrolla a través de la implementación de componentes. RUP describe cómo reutilizar los componentes existentes o implementar nuevos componentes con responsabilidades bien definidas, alcanzando un sistema fácil de mantener e incrementando las posibilidades de reutilización[15].

En este capítulo se aborda todo lo referente al flujo de trabajo implementación. Se muestran los diagramas de componentes correspondientes a cada uno de los casos de uso identificados para la Gestión de Datos de Ensayos Clínicos (GDEC), destacando en todos, mediante componentes, las dependencias existentes y la representación física de la estructura interna del sistema. Además se muestran ejemplos del código contenido en alguno de estos componentes y varias interfaces definidas para la GDEC.

4.1. Modelo de Implementación

La implementación comienza a partir del resultado obtenido en el flujo de Análisis y Diseño y la mayor parte de este flujo tiene lugar en la fase de Construcción. El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre éstos.

4.2. Diagrama de Componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Representa un fragmento de un sistema software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas.

Los diagramas de componentes muestran los componentes del software y los artilugios por los que está compuesto, como los archivos de código fuente, las librerías o las tablas de una base de datos. Este tipo de diagrama muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables. Los componentes software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución.

A continuación se muestran los diagramas de componentes realizados para dar paso a la implementación del sistema. Los componentes se pueden agrupar en 3 paquetes principales, siendo estos “Vista”, “Control” y “Modelo”, en correspondencia con el patrón de arquitectura utilizado Modelo-Vista-Controlador. Además se encuentra otros paquetes de los cuales dependen estos tres como Base de Datos y el paquete de Propiedades DAO

4.2.1. Vista General.

En el presente diagrama de componentes se encuentran resaltados de color amarillo los tres paquetes principales Vista, Control y Modelo. Este representa, mediante el “Caso de Uso X” genérico, una visión general de los diagramas de componentes de cada caso de uso.

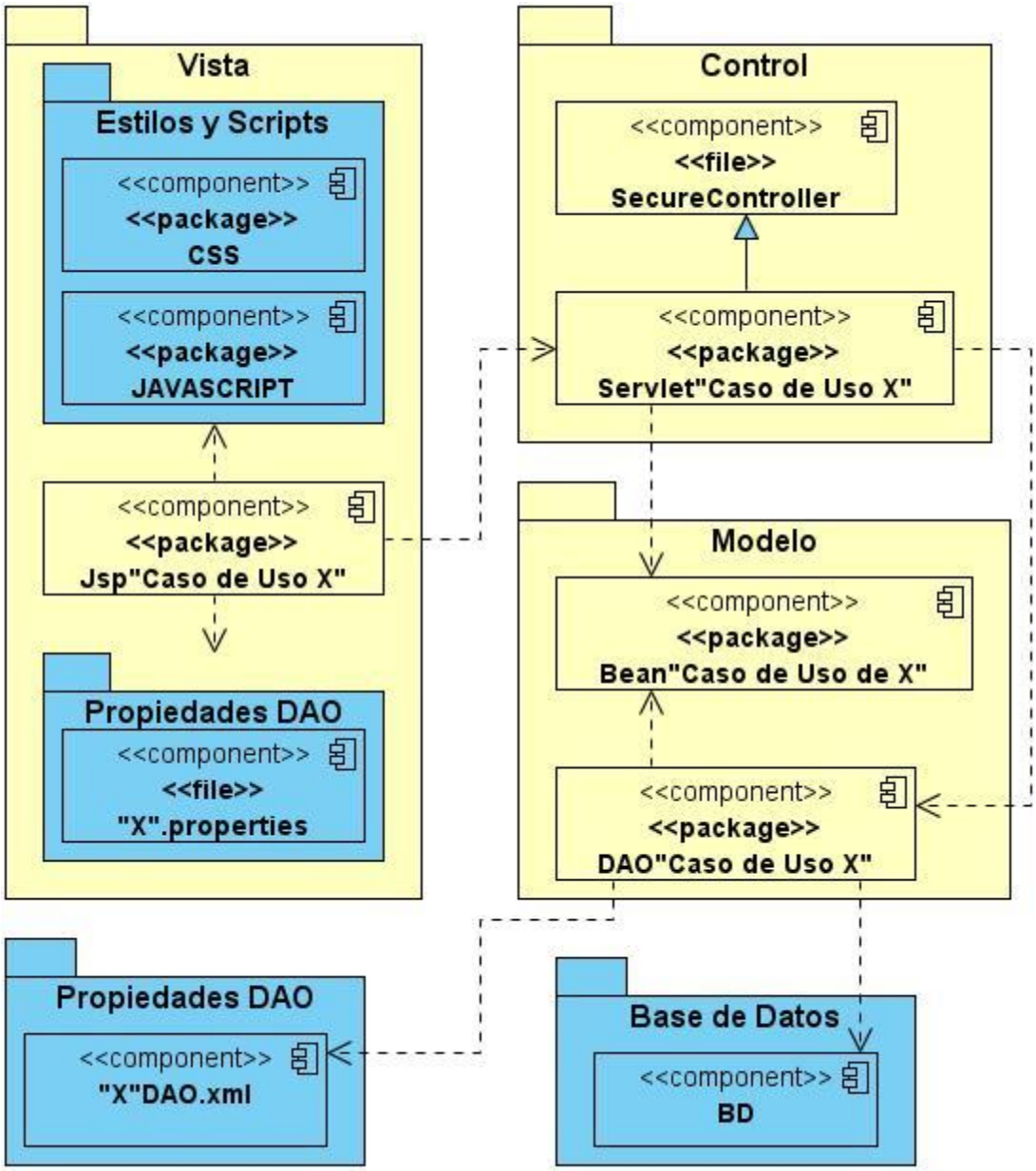


Fig. # 19 Diagrama de Componentes.

En el paquete Vista se incluyen 3 componentes:

- ✓ El componente Jsp"Caso de Uso X" contiene las páginas *.jsp encargadas de crear la vista para el caso de uso X y además las páginas servidoras de las estas que dependen.
- ✓ El componente Scripts contiene los ficheros *.js encargados de validar un conjunto de datos enmarcados en un caso de uso o conformar la vista.

- ✓ El componente CSSs contiene los ficheros *.css contenedores de los estilos que se muestran en la vista de un caso de uso.

En el paquete Control se incluye:

- ✓ El componente Servlet “Caso de Uso X” contiene todas las clases servlet encargadas del control de datos en el caso de uso X
- ✓ El componente SecureController: el cual es un servlet del cual heredan todos los demás servlet en el sistema. En el se definen un conjunto de pasos que son comunes para atender cualquier petición de la vista y se da la posibilidad que cada servlet que herede de él añada otros pasos que considere necesario a través de un método abstracto. Este componente no aparecerá en los demás diagramas debido a que para todos es igual al diagrama general.

En el paquete Modelo se incluyen 2 componentes:

- ✓ El componente Bean “Caso de Uso X” contiene todas las clases Bean encargadas de la contención de datos en el caso de uso X.
- ✓ El componente DAO “Caso de Uso X” contiene todas las clases DAO encargadas de la comunicación con la Base de Datos para el caso de uso X.

En el paquete Propiedades DAO se incluye 1 componente:

- ✓ El componente “X”DAO.xml contiene todos los ficheros *.xml contenedores de las consultas para obtener la información de la base de datos de el caso de uso X, estas consultas se pueden traducir mediante el uso de Digester.

A continuación se muestra el diagrama de componentes para cada caso de uso del sistema, todos de acuerdo a la estructura previamente presentada, mostrando las particularidades de cada caso de uso.

4.2.2. Caso de Uso "Visualizar listado de pacientes".

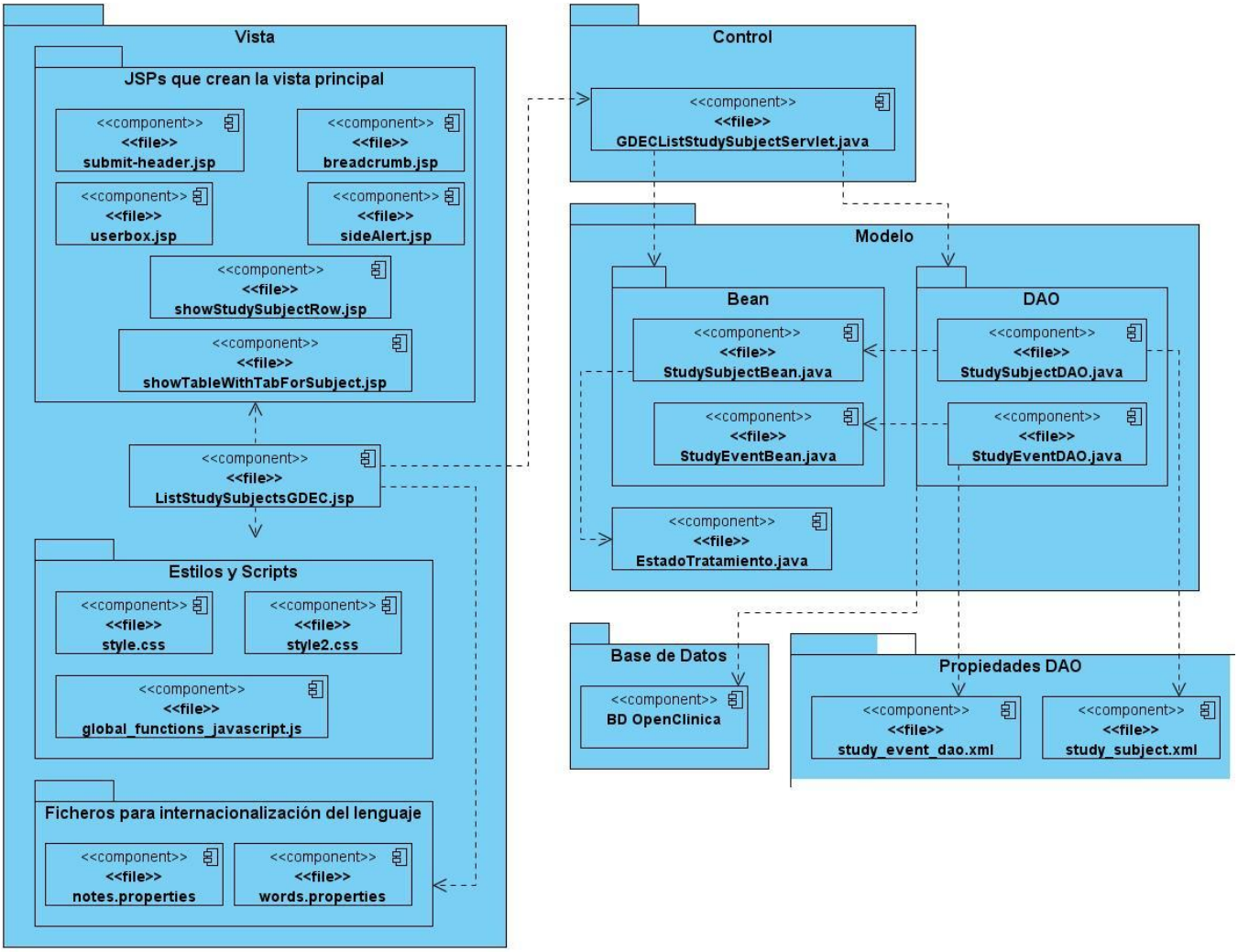


Fig. # 20 Diagrama de componentes "Visualizar listado de pacientes".

4.2.3. Caso de Uso “Visualizar listado de momentos de seguimiento”.

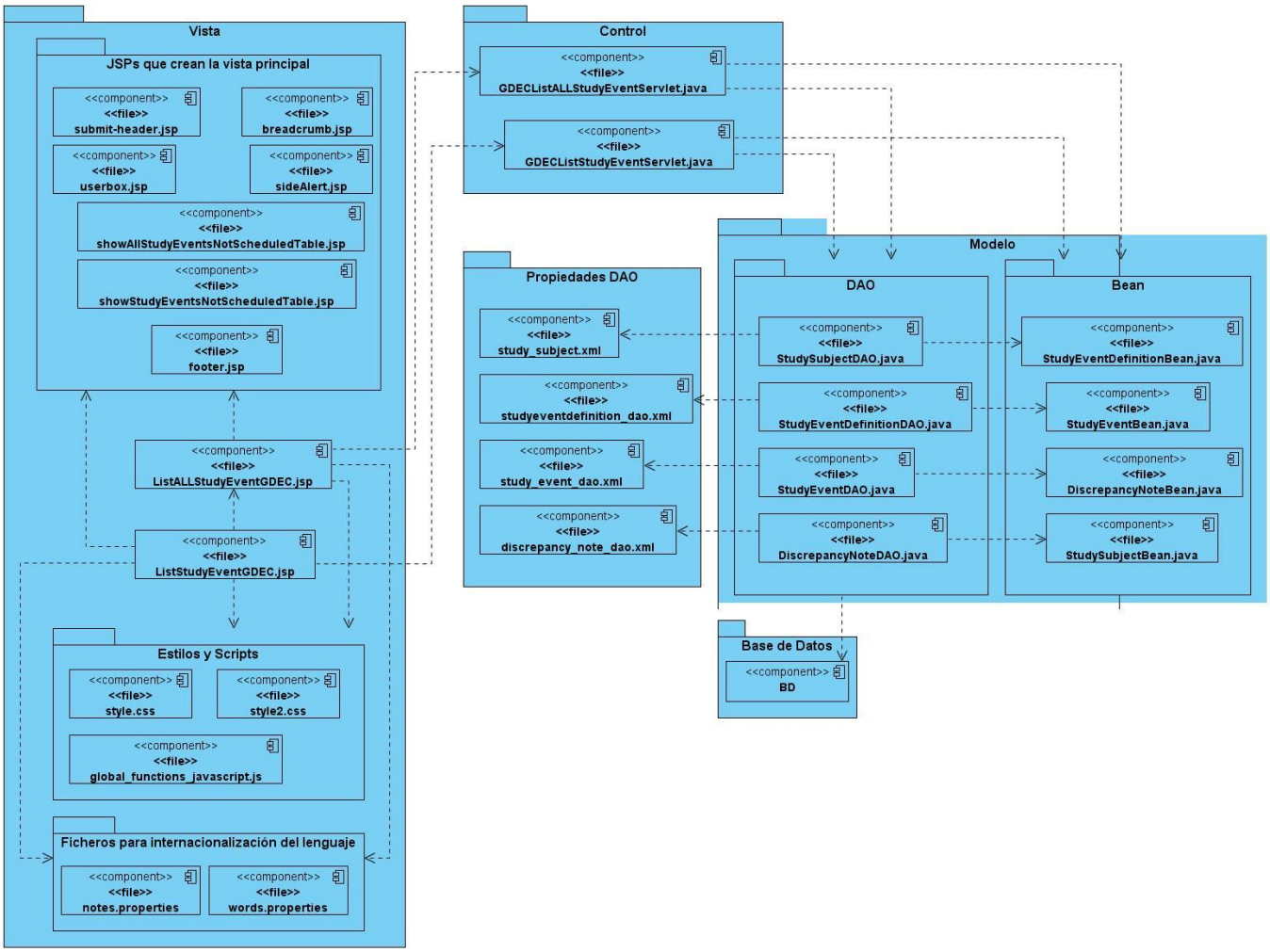


Fig. # 21 Diagrama de componentes “Visualizar listado de momentos de seguimiento”.

4.2.4. Caso de Uso "Visualizar listado de hojas de CRD".

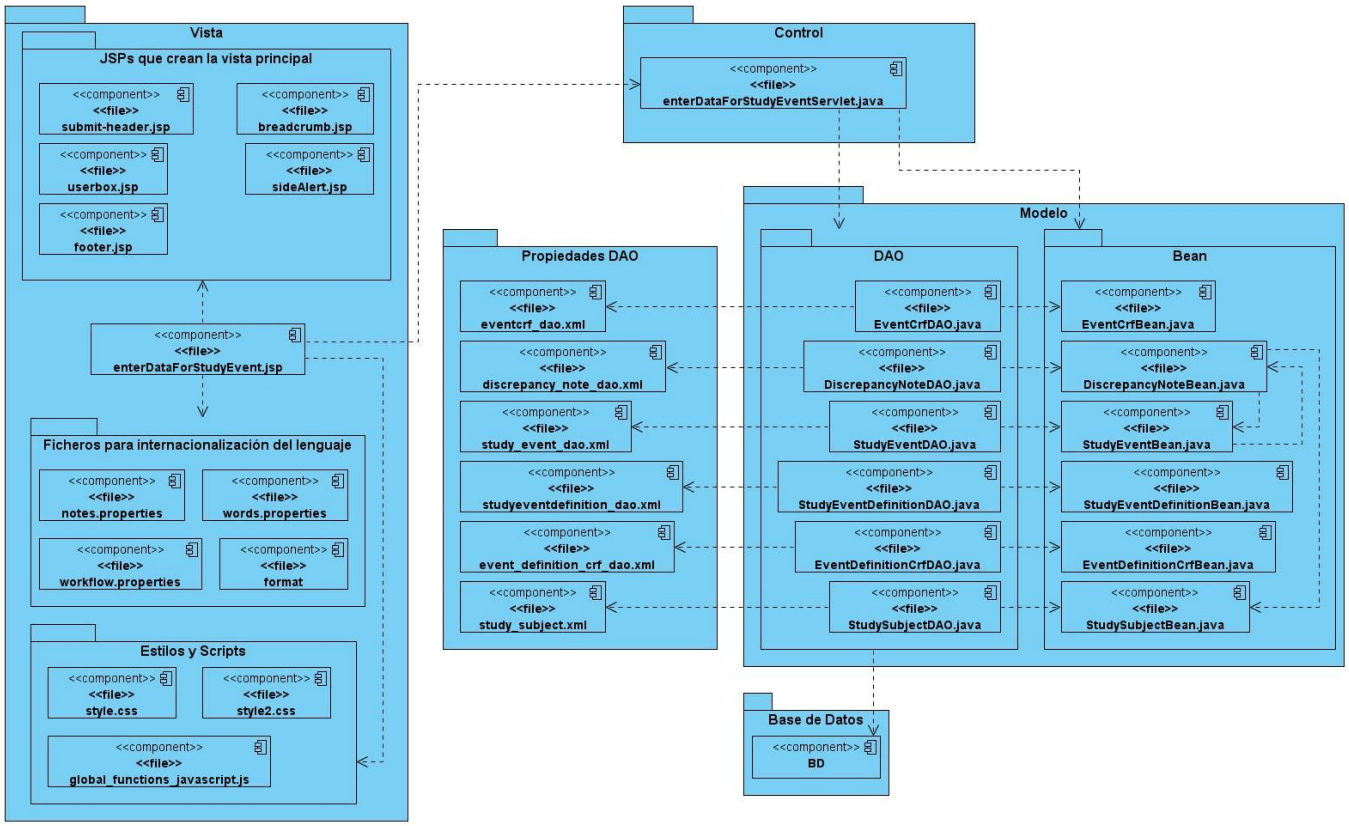


Fig. # 22 Diagrama de componentes "Visualizar listado de hojas de CRD".

4.2.5. Caso de Uso "Gestionar cronograma específico".

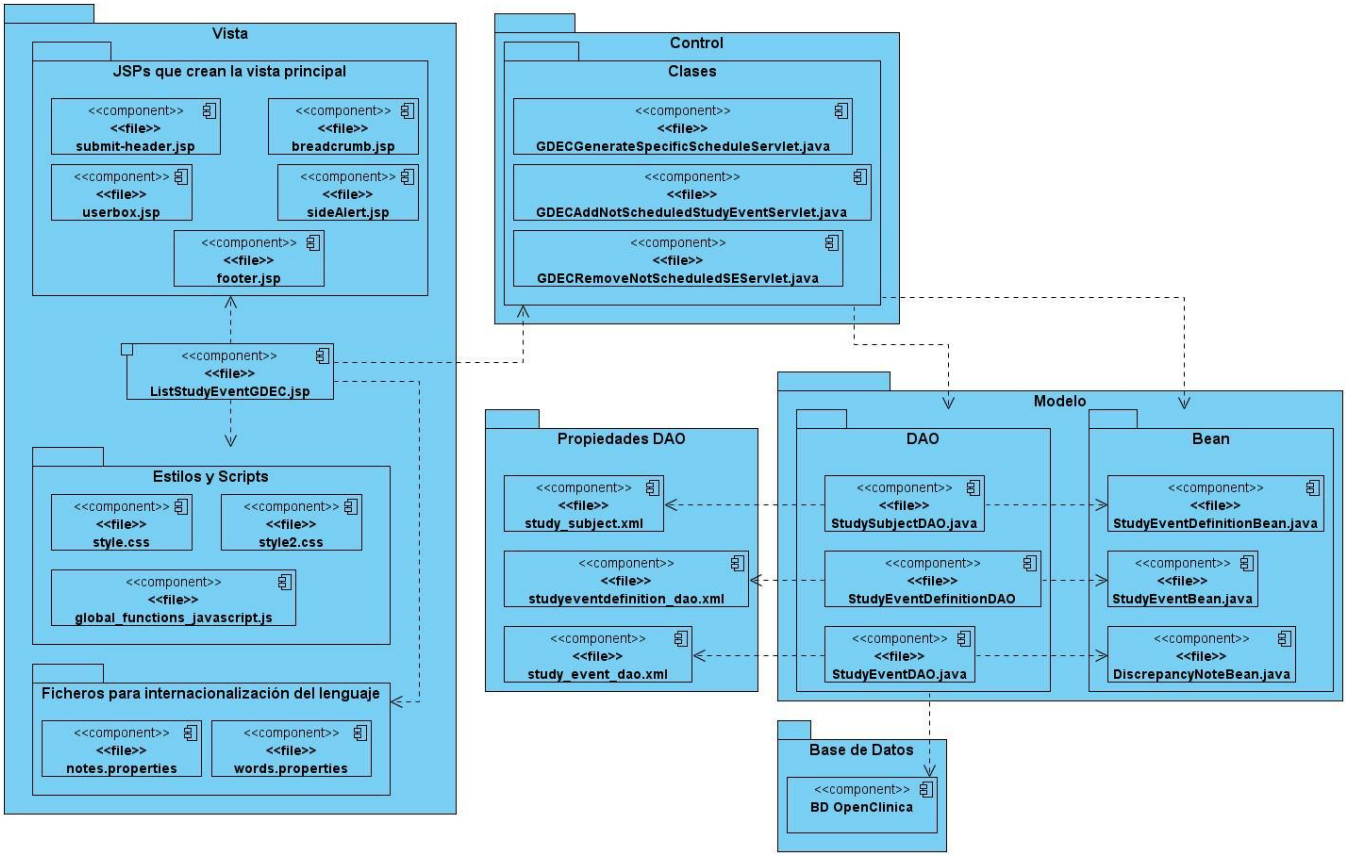


Fig. # 23 Diagrama de Componentes "Gestionar cronograma específico".

4.3. Ejemplos de Código

4.3.1. Función PopulateDisplaySpecificSchedule

El siguiente código pertenece a la función "PopulateDisplaySpecificSchedule(ArrayList sedb, ArrayList seb, ArrayList etapas, StudySubjectBean studySub)" de la clase ViewStudySubjectServlet del caso de uso "Gestionar Datos de Paciente", sección "Mostrar datos de Paciente".

```
public void PopulateDisplaySpecificSchedule(ArrayList sedb, ArrayList seb,
                    ArrayList etapas, StudySubjectBean studySub) throws ParseException
{
    /*
     * Sección 1
     *    Bloque de Código
     *  Sección 2
     *    Bloque de Código
     *  Sección 3
     *    Bloque de Código
     * *Sección 4
     *    Bloque de Código
     */
}
```

Fig. # 24 Función PopulateDisplaySpecificSchedule.

La función recibe por parámetros una lista de todas las definiciones de momentos de seguimiento asociados a un estudio (sedb), una lista de todos los momentos de seguimiento asociados a un paciente (seb), una lista con las etapas del cronograma específico definidas en el cronograma general de un estudio (etapas) y el paciente en cuestión (studySub). Esta función tiene como objetivo crear un grupo de variables con los valores que se mostrarán en el cronograma específico del paciente.

Este constituye un ejemplo del empleo del patrón modelo-vista-controlador ya que en esta función se puede apreciar como el controlador, que es la clase que contiene esta función, gestiona los datos del modelo y se los envía la vista, y como esta es la encargada de la lógica de presentación, es la responsable de conformar la vista que finalmente se le mostrará al usuario, en este caso el cronograma específico.

Primeramente se aprecia un ciclo, el cual es el responsable de conformar el arreglo de fechas en las cuales el paciente tiene programado algún momento de seguimiento (Fig. # 26).

Después se muestra otro ciclo en el que se cuentan cuantas fechas pertenecen a cada una de las etapas para que después en la vista se pueda conocer cuantas columnas debe expandirse la columna de cada etapa en la tabla que conforma el cronograma especifico (Fig. # 27).

Luego se pueden apreciar tres ciclos anidados con el objetivo de conformar una matriz, en la cual se tendrá en cada posición [i; j] un arreglo con los momentos de seguimientos que coinciden con la definición de momento de seguimiento en i y la fecha de inicio del momento en j (Fig. # 28).

Finalmente se tiene un grupo se sentencias a través de las cuales se le pasan todas las variables creadas (la matriz que representa en cronograma específico, el arreglo de fechas, el arreglo de etapas y el arreglo de definiciones de momentos de seguimiento) a la variable request y así de esta manera enviárselas a la vista para que conforme visualmente el cronograma especifico en forma de tabla (Fig. # 29).

```
/*
 * Sección 1
 *     Bloque de Código
 */

SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
ArrayList specificSch = new ArrayList();
ArrayList fechas = new ArrayList();

for (int i = 0; i < seb.size(); i++)
{
    String fecha = formato.format(((StudyEventBean)seb.get(i)).getDateStarted());
    boolean existe = false;
    for (int j = 0; j < fechas.size(); j++)
    {
        if(fechas.get(j).equals(fecha))
        {
            existe = true;
            break;
        }
    }
    if(!existe)
        fechas.add(fecha);
}
```

Fig. # 25 Función PopulateDisplaySpecificSchedule. Bloque de código de la Sección 1.

```
/*
 * Sección 2
 *    Bloque de Código
 */

for (int i = 0; i < etapas.size(); i++) {

   String inicioString = formato.format(studySub.getScheduledDate());
   Date inicioDate = new Date(formato.parse(inicioString).getTime()
       + (((ecEtapasBean)etapas.get(i)).getInicio_etapa() - 1) * 1000 * 60 * 60 * 24);
   inicioString = formato.format(inicioDate);

   String finString = formato.format(studySub.getScheduledDate());
   Date finDate = new Date(formato.parse(finString).getTime()
       + (((ecEtapasBean)etapas.get(i)).getFin_etapa() - 1) * 1000 * 60 * 60 * 24);
   finString = formato.format(finDate);

   int cantColumna = 0;
   for (int j = 0; j < fechas.size(); j++) {
         if(inicioString.compareTo((String)fechas.get(j)) <= 0
                 && finString.compareTo((String)fechas.get(j)) >= 0)
             cantColumna++;
   }
   request.setAttribute("etapa" + i, cantColumna);
}
```

Fig. # 26 Función PopulateDisplaySpecificSchedule. Bloque de código de la Sección 2.

```
/*
 * Sección 3
 *    Bloque de Código
 */

for (int i = 0; i < sedb.size(); i++)
{
    ArrayList specificSchRow =  new ArrayList();
    for (int j = 0; j < fechas.size(); j++)
    {
        specificSchRow.add(new ArrayList());
        for (int k = 0; k < seb.size(); k++)
        {
            String fecha = formato.format(((StudyEventBean)seb.get(k)).getDateStarted());
            if(fecha.equals(fechas.get(j)))
                if(((StudyEventBean)seb.get(k)).getStudyEventDefinitionId() ==
                    ((StudyEventDefinitionBean)sedb.get(i)).getId())
                {
                    ((StudyEventBean)seb.get(k)).setStudyEventDefinition((StudyEventDefinitionBean)sedb.get(i)
                    ((ArrayList)specificSchRow.get(j)).add(seb.get(k));
                }
        }

    }
    specificSch.add(specificSchRow);
}
```

Fig. # 27 Función PopulateDisplaySpecificSchedule. Bloque de código de la Sección 3.

```
/*
 * Sección 4
 *    Bloque de Código
 */

request.setAttribute("sedbs", sedb);
request.setAttribute("specificSch", specificSch);
request.setAttribute("fechas", fechas);
request.setAttribute("etapas", etapas);
```

Fig. # 28 Función PopulateDisplaySpecificSchedule. Bloque de código de la Sección 4.

4.3.2. Página Servidora ListStudyEventsGDEC.jsp.

En esta página servidora se evidencia el uso del patrón Vista Compuesta en cada una de las instrucciones “import” de manera que la vista final queda conformada una vez que se incluyan cada una de las JSP especificadas en las etiquetas <c:import> que cumplan con la condición especificada en la etiqueta <c:if> o <c:when> (Fig. # 30).

```
<h1><span class="title_submit">
<fmt:message key="view_all_study_event_for" bundle="${resText}"/> <c:out value="${studySub.name}"/>
<a href="javascript:openDocWindow('help/2_1_viewAllSubjects_Help.html')">
<img src="images/bt_Help_Submit.gif" border="0" alt="<fmt:message key="help" bundle="${resText}"/>
" title="<fmt:message key="help" bundle="${resText}"/>"></a></span></h1>

<h3><span class="title_submit"><fmt:message key="view_all_scheduled_study_event" bundle="${resText}"/>
</span></h3>
<c:import url="showStudyEventsTable.jsp">
    <c:param name="rowURL" value="showStudyEventRow.jsp" />
</c:import>

<h3><span class="title_submit"><fmt:message key="view_all_not_scheduled_study_event" bundle="${resText}"/>
</span></h3>

<c:choose>
    <c:when test="${all}">
        <c:import url="showAllStudyEventsNotScheduledTable.jsp" />
    </c:when>
    <c:otherwise>
        <c:import url="showStudyEventsNotScheduledTable.jsp" />
    </c:otherwise>
</c:choose>

<c:if test="${!studySub.hasSpecificScheduled}">
    <c:import url="generateSpecificSchedule.jsp" />
</c:if>

<c:import url="../include/workflow.jsp">
   <c:param name="module" value="submit"/>
</c:import>

<jsp:include page="../include/footer.jsp"/>
```

Fig. # 29 Página servidora ListStudyEventsGDEC.jsp (2).

4.3.3. Clase Controladora GDECCreateSiteNoteServlet.

Un ejemplo de cómo quedan constituidas las clases controladoras lo es GDECCreateSiteNoteServlet, la cual hereda de SecureController como todos los sevrlet del sistema, para utilizar las ventajas del patrón Método Plantilla (Templete Method), en el cual, en la clase padre SecureController se definen una serie de pasos a la hora de procesar una petición que son comunes para todos los servet y en las clases hijas se sobrescribe un método para especificar los pasos específicos de cada servlet que son añadidos al conjunto de pasos a realizar para atender una petición. Este método es "processRequest()", el cual es el encargado de gestionar los datos del modelo y enviárselos a la vista o de recibir los datos de la vista y enviarlos al modelo para ser almacenados.

A continuación se muestra este método en la clase GDECCreateSiteNoteServlet dividido en dos secciones. En la primera se cran un conjunto de objetos DAOs los cuales son utilizados para acceder a los datos en la base de datos o para instanciar los objetos que serán mostrados en la vista, en este

caso los objetos Bean. De esta forma se evidencia el patrón DAO y el Creador. También se reciben un grupo de parámetros de la vista y luego se gestiona de un grupo de datos en el modelo en dependencia de los datos recibidos de la vista (Fig. # 31).

Y en la segunda sección está compuesta por una bifurcación en la cual se ejecuta un bloque en dependencia de una condición, si la petición es por la acción de un formulario o no. En caso de que haya sido invocado por la acción de un formulario se procede a crear la nota de sitio con los datos introducidos por el usuario y guardarla en la variable de sección, en caso contrario se procede a transferir a la vista los datos a ser mostrados al usuario para crear la nota de sitio (Fig. # 32).

```
@Override
protected void processRequest() throws Exception {

    FormProcessor fp = new FormProcessor(request);

    int ssid = fp.getInt("ssid", true);
    int ecid = fp.getInt("ecid", true);
    int iid = fp.getInt("iid", true);
    String field = fp.getString("field", true);

    StudySubjectDAO ssdao = new StudySubjectDAO(sm.getDataSource());
    StudyEventDefinitionDAO seddao = new StudyEventDefinitionDAO(sm.getDataSource());
    StudyEventDAO sedao = new StudyEventDAO(sm.getDataSource());
    EventCRFDAO ecdao = new EventCRFDAO(sm.getDataSource());
    CRFVersionDAO cvdao = new CRFVersionDAO(sm.getDataSource());
    CRFDAO cdao = new CRFDAO(sm.getDataSource());
    ItemDAO idao = new ItemDAO(sm.getDataSource());
    StudySubjectBean studySub = (StudySubjectBean)ssdao.findByPK(ssid);
    EventCRFBean eventCrf = (EventCRFBean)ecdao.findByPK(ecid);
    CRFVersionBean crfVersion = (CRFVersionBean)cvdao.findByPK(eventCrf.getCRFVersionId());
    CRFBean crf = (CRFBean)cdao.findByPK(crfVersion.getCrfId());
    StudyEventBean studyEvent = (StudyEventBean)sedao.findByPK(eventCrf.getStudyEventId());
    StudyEventDefinitionBean eventDef =
        (StudyEventDefinitionBean)seddao.findByPK(studyEvent.getStudyEventDefinitionId());
    ItemBean item = (ItemBean)idao.findByPK(iid);
```

Fig. # 30 Clase controladora GDECCreateSiteNoteServlet método principal (inicio).

```
if(!fp.isSubmitted()) {
    request.setAttribute("studySub", studySub);
    request.setAttribute("studyEvent", eventDef.getName());
    request.setAttribute("fechaSE", studyEvent.getDateStarted());
    request.setAttribute("eventCRF", crf.getName());
    request.setAttribute("item", item);
    request.setAttribute("fecha", Calendar.getInstance().getTime());
    request.setAttribute("ecid", ecid);
    request.setAttribute("field", field);
    forwardPage(Page.Create_Site_Note); }
else {
    String descripcion = fp.getString("descripcion", true);
    String detalles = fp.getString("detalles", true);
    DiscrepancyNoteDAO dndao = new DiscrepancyNoteDAO(sm.getDataSource());
    DiscrepancyNoteBean note = new DiscrepancyNoteBean();
    note.setDescription(descripcion);
    note.setDetailedNotes(detalles);
    note.setOwner(ub);
    note.setCreatedDate(new Date());
    note.setResolutionStatusId(6);
    note.setDiscrepancyNoteTypeId(10);
    note.setStudyId(currentStudy.getId());

    FormDiscrepancyNotes noteTree =
        (FormDiscrepancyNotes) session.getAttribute(FORM_DISCREPANCY_NOTES_NAME);

    if (noteTree == null)
        noteTree = new FormDiscrepancyNotes();

    noteTree.addNote(field, note);
    noteTree.addIdNote(note.getEntityId(), field);
    session.setAttribute(FORM_DISCREPANCY_NOTES_NAME, noteTree);
    forwardPage(Page.ADD_DISCREPANCY_NOTE_DONE); }
```

Fig. # 31 Clase controladora GDECCreateSiteNoteServlet método principal (final).

4.3.4. Uso de Digester.

Digester es una librería de clases que se utiliza para procesar archivos XML a través de la búsqueda de reglas. Existen dos métodos básicos para parsear documentos XML. Uno es utilizar el método DOM (Document Object Model). El segundo método es utilizar SAX y parsear el documento XML a través de eventos. Un parser SAX genera eventos cada vez que encuentra una etiqueta (tag) XML. El método DOM es más fácil de implementar, aunque es más lento ya que utiliza más recursos que SAX. Digester simplifica el parseo SAX brindando una interfaz de alto nivel a los eventos que genera SAX. Esta interfaz, oculta toda la complejidad que trae la navegación de documentos XML permitiendo al programador concentrarse en el procesamiento de los datos en vez de preocuparse por el parseo del documento. Digester introduce tres conceptos importantes: patrones de búsqueda de elementos,

reglas de procesamiento y pila de objetos. El uso de esta librería se muestra en la Fig. # 33, las consultas a la base de datos se encuentran en ficheros *.xml, para el caso de uso que se muestra a continuación (Gestionar paciente) en el fichero study_event_dao.xml, el uso de Digester asegura y oculta la forma en la que el sistema interpreta estos ficheros para que las clases DAO interactúen con la base de datos.

```
public ArrayList findAllForSpecificSchedule(int subjectId, Date fecha)
{
    ArrayList answer = new ArrayList();

    this.setTypesExpectedWithType();

    HashMap variables = new HashMap();
    variables.put(new Integer(1), new Integer(subjectId));
    variables.put(new Integer(2), new Date(fecha.getTime()));

    ArrayList alist = this.select(digester.getQuery("findAllForSpecificSchedule"), variables);

    Iterator it = alist.iterator();
    while (it.hasNext()) {
        StudyEventBean seb = (StudyEventBean) this.getEntityFromHashMapWithType((HashMap) it.next());
        answer.add(seb);
    }

    return answer;
}
```

Fig. # 32 Uso de la librería Digester.

4.4. Ejemplos de Interfaces

A continuación se muestra un grupo de interfaces nuevas que han sido incorporadas al sistema OpenClinica en el proceso de desarrollo del sistema alasClínicas. Estas interfaces permiten el proceso de Gestión de Datos de Ensayos Clínicos como se hace en Cuba, partiendo de un listado de pacientes (ver Fig. # 39) y culminando con el llenado de las hojas de CRD a partir de seleccionar una en un listado de hojas de CRD (ver Fig. # 42). Se incluyen además una vista con los datos de un paciente y el cronograma especifico del mismo (ver Fig. # 40) y otra con un listado de momentos de seguimiento asociados a un paciente (ver Fig. # 41) que incluye la sección “generar cronograma específico” ya que aún no ha sido generado en el caso del paciente que se muestra como ejemplo. En cada una de estas interfaces quedan evidenciados los requisitos funcionales y no funcionales identificados previamente.

4.4.1. Listado de pacientes.

En esta interfaz se visualizan todos los pacientes que pertenecen a un sitio determinado y se da la posibilidad de gestionar cada uno se ellos (ver sus datos, interrumpirlos si es posible y además insertar más pacientes al estudio). Anteriormente en el sistema OpenClinica en esta interfaz se permitía además programar los momentos de seguimiento de cada paciente uno por uno, pero como esto no cumple con los requisitos del CIM y fue modificado para el sistema alasClínicas, en la siguiente interfaz se explica con mas detalles.

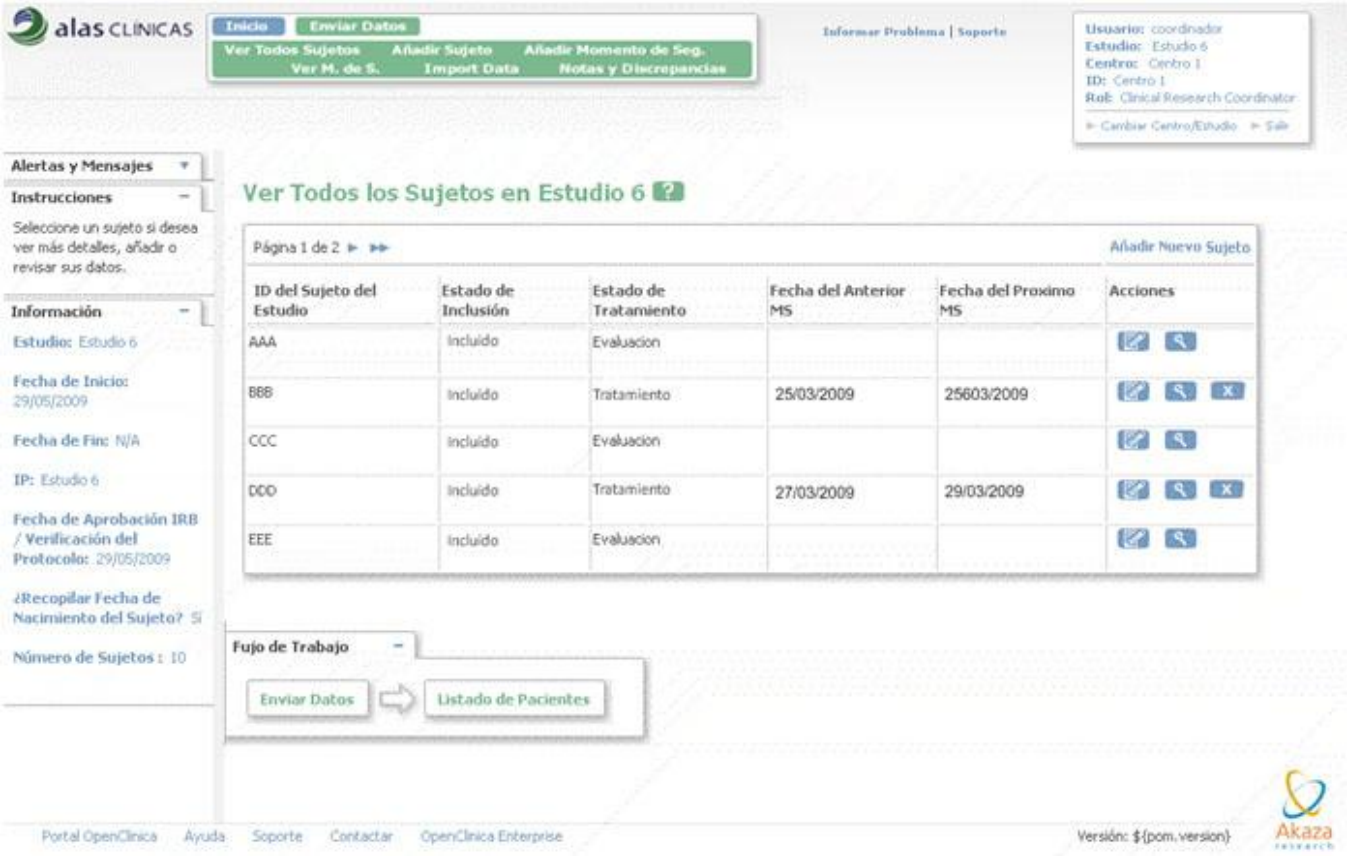


Fig. # 33 Interfaz "Listado de pacientes".

4.4.2. Listado de momentos de seguimiento.

En esta interfaz se visualizan todos los momentos de seguimiento de un paciente en los que es posible agregar datos a sus hojas de CRD, estos aparecen en dos listados, los programados y los no programados. Además se da la posibilidad de gestionar el cronograma especifico de la forma como se

realiza la GDEC en Cuba y no como lo permitía el sistema OpenClinica anteriormente, en la cual se podía programar cada momento de seguimiento uno por uno con fecha de inicio y fin que el CIC quisiera, ahora en el sistema alasClínicas el CIC indica una fecha y a partir de esta se programan automáticamente todos los momentos de seguimiento programados según la planificación de estos en el cronograma general (en el cronograma general se especifica el día que debe de iniciar cada uno de estos momentos de seguimiento y cuantos días va a durar, con este día, el tiempo de duración y la fecha que indica el CIC es que el sistema determina para que fecha debe de programar el inicio y el fin de cada uno de los momentos) y se permite además añadir y eliminar, siempre que sea posible, momentos de seguimientos no programados.

Fig. # 34 Interfaz "Listado de momentos de seguimiento".

4.4.3. Listado de hojas de CRD.

En esta interfaz se muestran datos que son de interés a la hora de llevar a cabo la conducción de EC en Cuba, se da la posibilidad además de realizar un grupo de acciones sobre la hojas de CRD como introducirle datos o solamente ver sus datos, y otras que no forman parte de la GDEC como monitorear la cada una de las hojas o firmarlas que en esta figura no se muestran porque el CIC no realiza estas actividades.

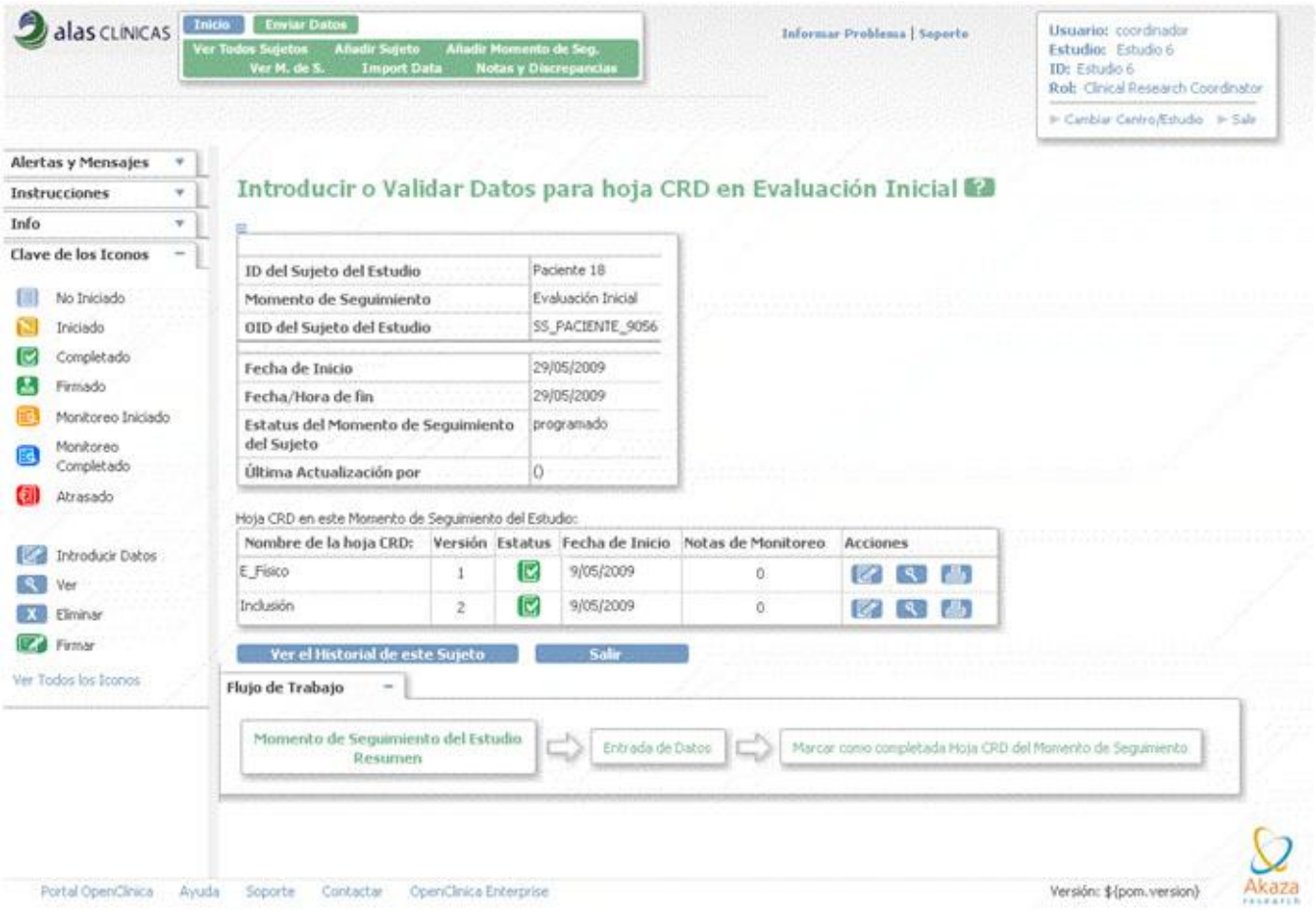


Fig. # 35 Interfaz "Listado de hojas de CRD".

4.4.4. Datos de paciente.

En esta interfaz se muestran todos los datos de un paciente y a demás el cronograma especifico del mismo, en el cual se puede ver de todos los momentos de seguimientos definidos para un paciente la

fecha en que inician, el estado que tienen, y a que etapa pertenecen. En caso que el paciente no tenga generado el cronograma específico y ya se le pueda generar, en esta interfaz también el CIC lo puede generar.

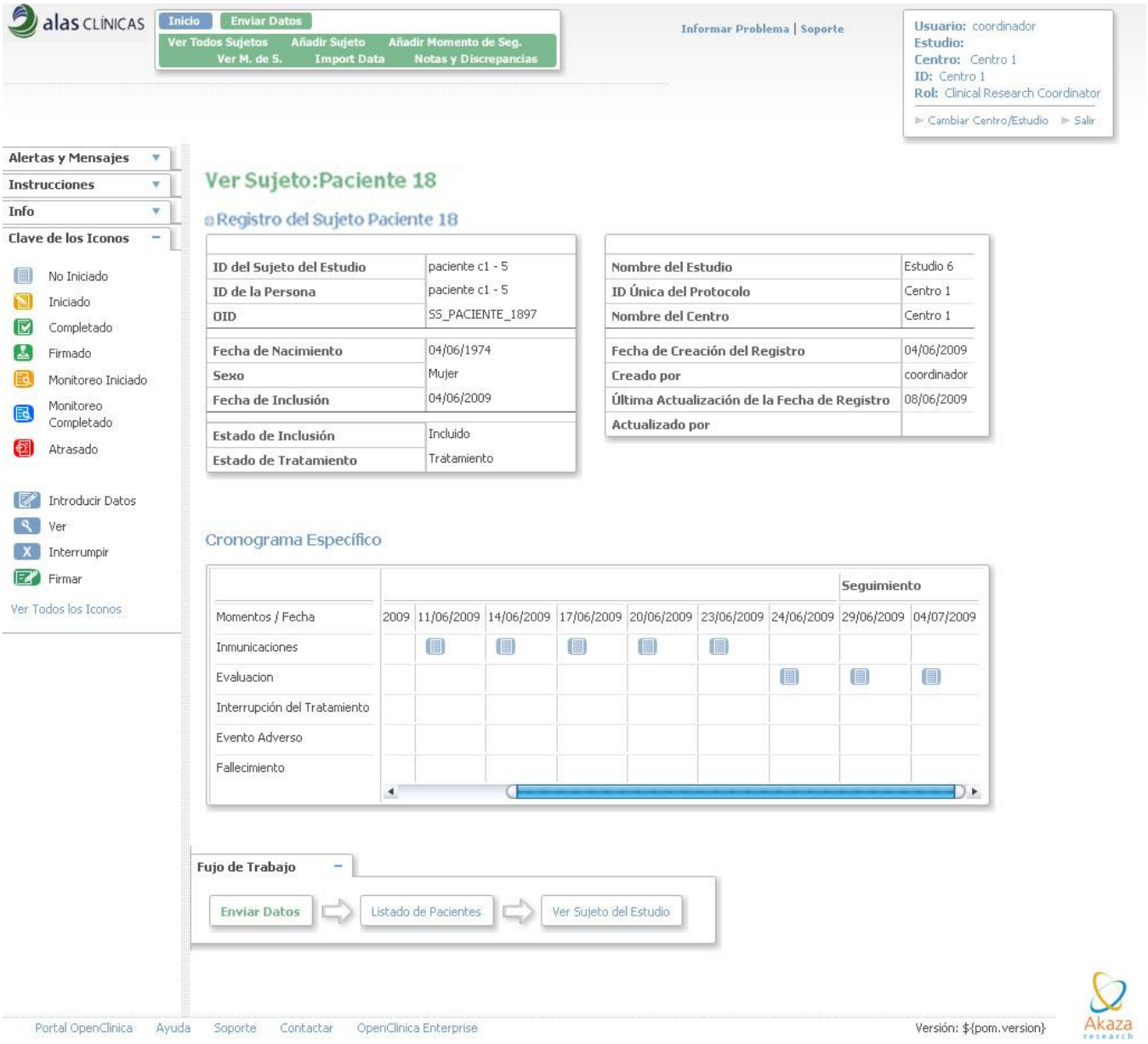


Fig. # 36 Interfaz "Datos de paciente".

4.5. Conclusiones

Con el desarrollo en este capítulo de los diagramas de componentes para cada caso de uso y la representación de fragmentos de código, utilizados en la implementación del sistema alasClínicas, se evidencia la disposición física del sistema y se muestra su estructura interna.

Conclusiones Generales

Con el desarrollo del presente trabajo se puede arribar a las siguientes conclusiones:

- ✓ Se hizo un análisis de los principales aspectos y conceptos del proceso de conducción de ensayos clínicos cubanos.
- ✓ Se hizo un análisis de los principales aspectos y conceptos del sistema para la conducción de ensayos clínicos: OpenClinica.
- ✓ Se realizó la captura de los requisitos funcionales y no funcionales, agrupándose los primeros en casos de uso.
- ✓ Se dio cumplimiento a los objetivos trazados logrando identificar las funcionalidades a agregar o modificar, realizando el diseño de las clases que se agregaron o se modificaron y la implementación de las mismas al sistema ***alasClínicas***.

Con la culminación de este trabajo se espera un sistema capaz de garantizar de forma segura y rápida la Gestión de Datos de Ensayos Clínicos, facilitando este trabajo al Centro de Inmunología Molecular y de esta forma contribuir al desarrollo de nuevos fármacos en Cuba para combatir disímiles enfermedades.

Recomendaciones

1. Asegurar sobre el sistema el cumplimiento de un proceso de mantenimiento, soporte y actualización periódica para garantizar la fiabilidad, seguridad y agilidad del sistema, contribuyendo a mejorar de forma sustancial la calidad del mismo y de esta obtener un funcionamiento óptimo del sistema y de la información por él gestionada.

2. Modificar el módulo que se encarga de la validación de los datos en las hojas de CRD y que este sea el encargado de asegurar la consistencia de los datos.

3. Capacitar al personal encargado de operar el sistema en el CIM sobre aspectos particulares de su funcionamiento.

Referencias Bibliográficas

1. **REUMATOLOGÍA, I. F. D.** *Ensayos Clínicos*, En Línea. [2008]. Disponible en: http://www.institutferran.org/ensayos_cl%C3%ADnicos.htm

2. **MOLECULAR, C. D. I.** *Portal del CIM* En línea. [2009]. Disponible en: http://www.cim.sld.cu/

3. **OPENCLINICA, S. O. D.** *About OpenClinica: Features*, En línea. [2009]. Disponible en: http://www.openclinica.org/section.php?sid=1

4. **SYSTEM, G. O**. *La Definición de Software Libre*, En Línea. [2009]. Disponible en: http://www.gnu.org/philosophy/free-sw.es.html

5. **LIBRE., F. D. S.** *Licencia Pública General GNU*, 1991. [2009]. Disponible en: http://www.sindominio.net/gugs/licencias/gpl-2-es_MX.html

6. *Sitio Oficial de OpenClinica*. En línea. [2009]. Disponible en: http://www.openclinica.org/

7. **GUTIÉRREZ, G. A.** *J2EE. Una plataforma para el cómputo empresarial*, 2003. [2008]. Disponible en: http://www.enterate.unam.mx/Articulos/2003/junio/j2ee.htm

8. **NÚÑEZ, J. M. B.** *Investigación de la plataforma J2EE y su aplicación práctica*. Departamento de Ciencias de la Computación. Facultad de Ciencias Físicas y Matemáticas. Santiago de Chile, Universidad de Chile, 2003. 116. p.

9. **BRIANO, F.** *Control de versiones con Subversion*, 2008. 33p.

10. **EDUCATION, P.** *Modelo del Dominio*, En Línea. 10p.

11. **PAKOZ, Z.** *Procesos De La Ingeniería De Requerimientos*, En línea. [2009]. Disponible en: http://www.mitecnologico.com/Main/ProcesosDeLaIngenieriaDeRequerimientos

12. **GURU, S.** *Patrones de Casos de Uso*, 2005. 60p.

13. **NARANJO, M.** *Fundamentos de Definición de Arquitectura de Software* Bogotá, 2005. 2009.

14. **LUGO, R.** *Patrón "Data Access Object"* 2007. [2009]. Disponible en: http://www.proactiva-calidad.com/java/patrones/DAO.html

15. **SOLUTIONS, T.** *Metodología de desarrollo*, En línea. 16p.

Bibliografía Consultada

1. **ABIÁN, M. Á.** *El Archipiélago Eclipse 2da Parte*, 2003. [Disponible en: http://www.javahispano.org/contenidos.item.action?id=1081&menuId=ARTICLES
2. **CLÍNICOS, C. N. C. D. E.** *Registro de Ensayos Clínicos Cubanos*, En línea. [2009]. Disponible en: http://registroclinico.sld.cu/
3. **DÍAZ, A. L. y GARCÍA, L. R.** *Sistema de Manejo de Datos de Ensayos Clínicos.*: Facultad 6. La Habana, Universidad de las Ciencias Informáticas, 2007. 109. p.
4. **EDUCATION, P.** *Modelo del Dominio*, en línea. 10p.
5. **FIGUEROA, P.** *Conceptos de un diagrama de implementación Diagrama de componentes* En linea. [2009]. Disponible en: http://www.cs.ualberta.ca/~pfiguero/soo/uml/implementacion01.html
6. **GARCERANT, I.** *Modelo de Dominio* 2008. [2009]. Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/
7. **JACOBSON, I.; G. BOOCH*, et al***. *El Proceso Unificado de Desarrollo de Software*, 2000.
8. **LARMAN, C.** *UML y Patrones: Introducción al Análisis y Diseño Orientado a Objetos*, 1999.
9. **LUGO, R.** *Patrón "Data Access Object"* 2007. [2009]. Disponible en: http://www.proactiva-calidad.com/java/patrones/DAO.html
10. **MOLECULAR, C. D. I.** *Portal del CIM* En línea. [2009]. Disponible en: http://www.cim.sld.cu/
11. **NARANJO, M.** *Fundamentos de Definición de Arquitectura de Software* Bogotá, 2005. 2009.
12. **ORTIZ, A. M.** *Bases de datos: Modelos de datos*, 2000. [Disponible en: http://elies.rediris.es/elies9/4-2.htm
13. *Patrones de Diseño en Aplicaciones Web con J2EE*. 2009. [Disponible en: http://www.articuloweb.com/pdf.php?art_id=192
14. *pgAdmin 1.6 online documentation*. En línea. [2009]. Disponible en: http://www.pgadmin.org/docs/1.6/pg/release-8-2-4.html
15. **POSTGRESQL, T. F**. *Acerca de PostgreSQL*. Disponible en: http://www.postgresql.org/about/
16. **SÁNCHEZ, S. G.; R. M. SALCEDO*, et al.*** *Patrón Iterador*, En línea.
17. **SYSTEMS, S.** *El Modelo Físico*, 2007. [2009]. Disponible en: http://sparxsystems.com.ar/resources/tutorial/physical_models.html
18. **TRIKARTY y BIOTECH, H.** *Situación actual y oportunidades de negocio en el sector biotecnológico en América Latina*. *Genoma España*, 2005. 41p.

Glosario de Términos

A

ACID: es un acrónimo de Atomicidad (Atomicity), Consistencia (Consistency), Aislamiento (Isolation) y Durabilidad (Durability).

API: Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Representa una interfaz de comunicación entre componentes software.

B

Bean: Componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

Biofármacos*:* Producto medicinal que consiste de una proteína y/o un ácido nucleico. Los biofármacos pueden considerarse medicamentos críticos, pues se utilizan para mejorar y salvar la vida de pacientes con enfermedades crónicas discapacitantes que en algunos casos pueden resultar mortales. Los biofármacos se usan para tratar varios tipos de cáncer, enfermedades auto inmunes como la artritis reumatoide, la esclerosis múltiple y enfermedades degenerativas como el Alzheimer.

C

CASE: Ingeniería de Software Asistida por Ordenador (por sus siglas en inglés Computer Aided Software Engineering)

CIC: Coordinador de Investigación Clínica. Persona implicada en la Gestión de los Datos de un Ensayo Clínico.

CIM: Centro de Inmunología Molecular.

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

Cronograma Específico: Tabla que contiene una programación de todos los momentos de seguimiento, definidos para un paciente, para una fecha determinada a partir de una fecha indicada por el CIC y la planificación de cada momento del estudio en el cronograma general. En este los días del cronograma general se traducen a fechas específicas por paciente.

Cronograma General: Planificación de todos los momentos de seguimiento de un estudio, que debe de ser cumplida por todos los pacientes del mismo. En este se especifica el día para el que se programó cada momento de seguimiento.

CRUD: Crear (Create), Buscar (Read), Modificar (Update) y Eliminar (Delete).

CU: Caso de uso

CUS: Caso de uso del sistema.

D

Definición de Momento de Seguimiento: Son aquellas que se crean en el cronograma general con un conjunto de características entre las que se encuentran días, plazo de llenado, tipo y hojas de CRD asociadas a la misma, especifican como debe de ser un momento de seguimiento.

DOM: Es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

E

EC: Ensayo Clínico.

Estudio: Ensayo Clínico.

F

G

GDEC: Gestión de Datos de Ensayos Clínicos. Conjunto de procesos de un Ensayo Clínico guiado por el CIC para gestionar información asociada a Pacientes, Hojas de CRD y Momentos de Seguimiento de un estudio.

H

Hoja de CRD: Modelo o Formulario que agrupa un conjunto de variables sobre datos de interés del paciente para el estudio.

I

IDE: Entorno de Desarrollo Integrado por sus siglas en inglés (Integrated Development Environment). Entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Inclusión: Tipo de estado que puede asumir un paciente de acuerdo a su inclusión en el Estudio.

J

K

L

M

Momento de Seguimiento: Conjunto de hojas de CRD que se le asocian a un paciente en una fecha planificada y con un plazo de tiempo para el llenado de todas.

Monitor: Persona implicada en el proceso de monitoreo de un Ensayo Clínico.

Monitoreo: Proceso que se encarga de revisar los datos asociados a una hoja de CRD con el fin de identificar posibles errores desde el punto de vista clínico y pedir explicaciones por estos errores a través de las notas de monitoreo.

N

No Programado: (Referente a “momento de seguimiento”) Tipo de momento de seguimiento que se le asocia a un pacientes si ocurre algún problema durante el estudio del mismo. Su realización no es fija en los pacientes de un estudio.

O

P

Paciente: Persona asociada a un estudio.

Parsear: (Referente a “Parser”) Proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical. Acción de Parseo.

Parseo: (Referente a “Parser”): Proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical.

Parser: Analizador sintáctico. Una de las partes de un compilador que transforma su entrada en un árbol de derivación.

Plazo de tiempo: Rango de días en los que debe de iniciar la entrada de datos de momento de seguimiento.

Plugin: Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica.

Programado: (Referente a “momento de seguimiento”) Tipo de momento de seguimiento cuya fecha y plazo de llenado dependen del una día y plazo concebidos en su definición en el cronograma general. A diferencia de los no programados la ocurrencia de este es fija para todos los pacientes de un estudio.

Q

R

Request: Objeto para pasar variables por la URL a través de los formularios y obtener informaciones prácticas sobre el servidor o el cliente.

S

Sujeto: Persona.

SSL: Protocolo de Capa de Conexión Segura. Proporciona comunicaciones seguras por una red, comúnmente Internet.

T

Tratamiento: Tipo de estado que puede asumir un paciente de acuerdo a la etapa del estudio en que se encuentre. Varía a medida que al paciente le es llenado un momento de seguimiento de una etapa posterior a la que se encuentra actualmente.

U

UML: Lenguaje de modelado.

URL: Localizador de Recurso Uniforme (en inglés Uniform Resource Locator), la dirección global de documentos y de otros recursos en la Web.

V

Variable: (Referente a “Hoja de CRD”) Cada uno de los datos que se recogen en una hoja de CRD como por ejemplo: el nombre del paciente, la edad del paciente, la provincia etc.

W

X

Y

Z

Anexos

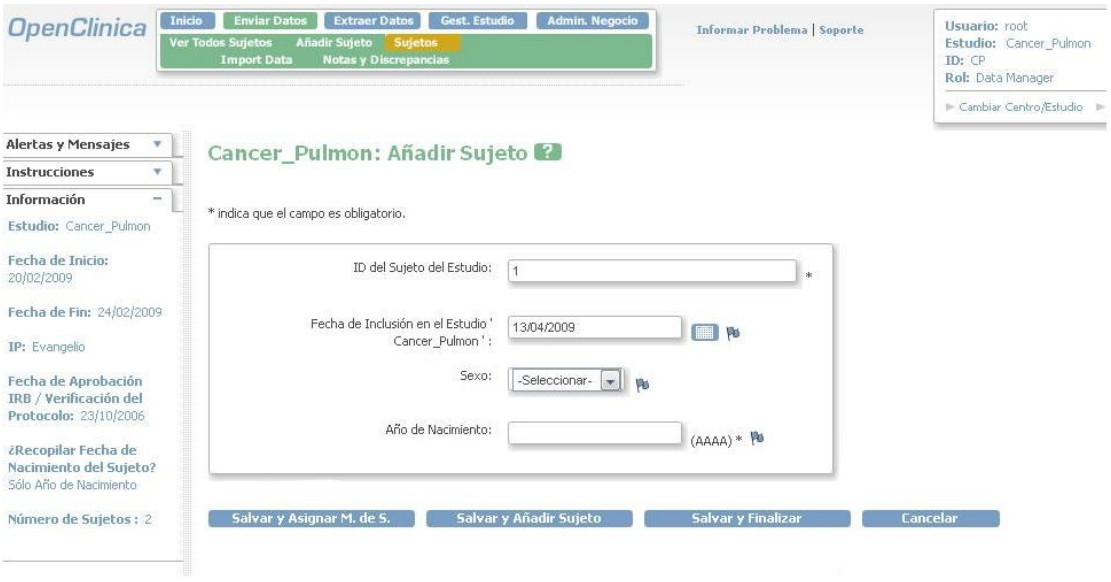
Anexo 1. Descripción de casos de uso.

CU Gestionar Paciente.

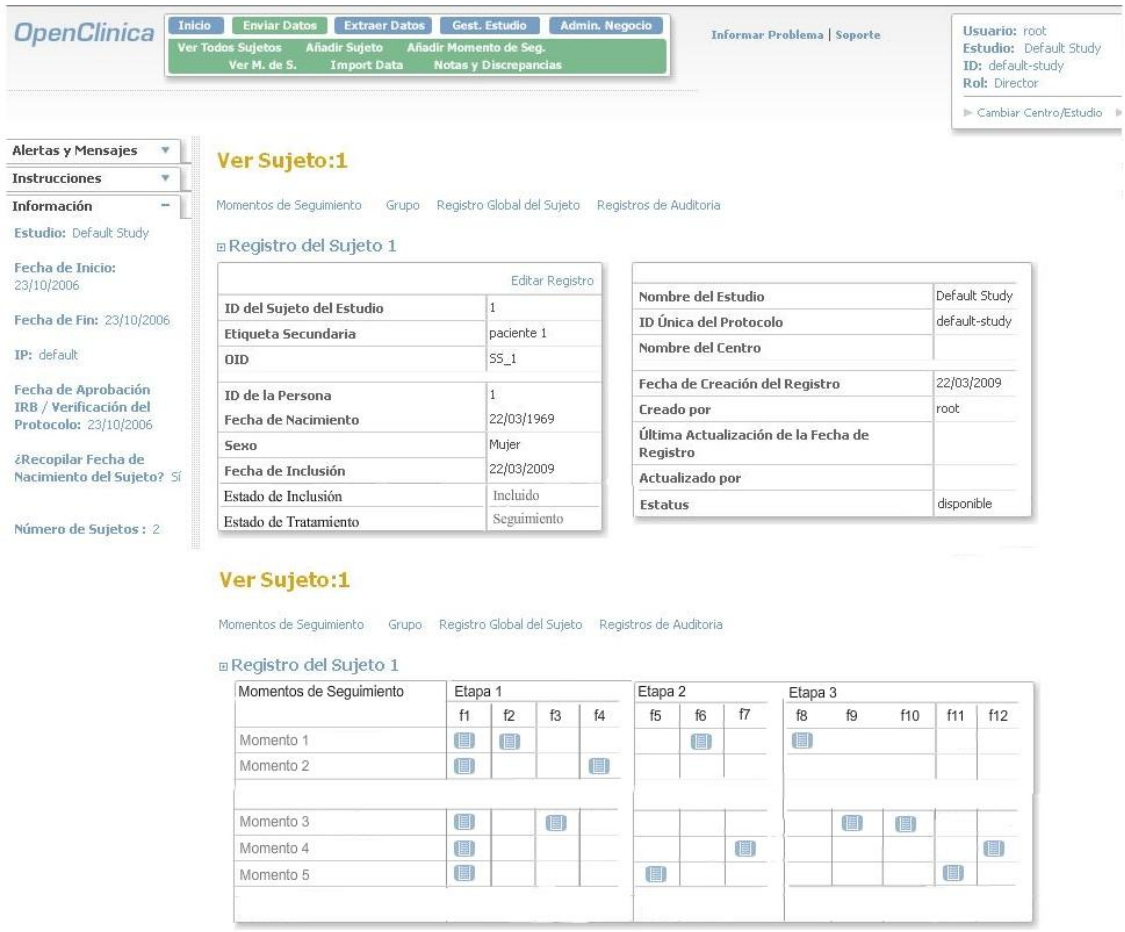
Este CU ya existe en el sistema OpenClinica pero será modificado porque no cumple con muchas de las políticas de la conducción de EC en Cuba, por ejemplo: a la hora de agregar Pacientes al estudio se recogen datos innecesarios. Además no todos los datos que se muestran en las interfaces asociadas a este son de importancia para el CIM por lo que estos serán cambiados por aquellos que son verdaderamente significativos para la conducción de EC en nuestro país.

Caso de Uso:	Gestionar Paciente
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el Coordinador de Investigación Clínica indica una de las siguientes opciones: - Insertar paciente - Mostrar datos de un paciente - Interrumpir un paciente De acuerdo a la opción seleccionada el sistema se encarga de insertar un paciente, mostrar los datos de un paciente o interrumpir un paciente.
Precondiciones:	
Referencias	R4, R16, R17, R18, R19, R20

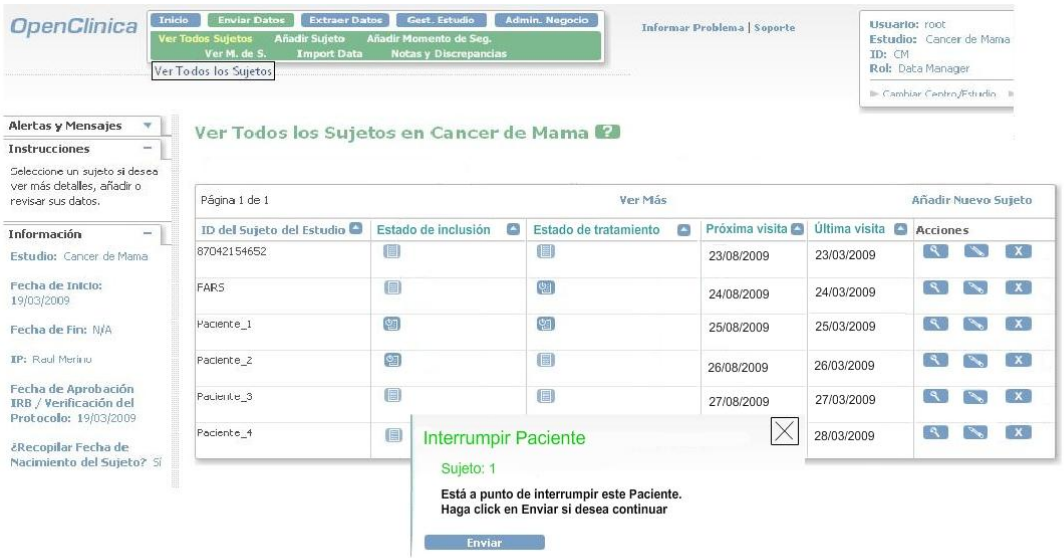
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El CIC indica realizar una de las siguientes opciones en el listado de pacientes: - Insertar paciente. - Mostrar datos de un paciente. - Interrumpir un paciente. Las dos últimas opciones se indican sobre el paciente con que se va a trabajar.	En dependencia de la opción que indique el actor el sistema ejecuta una de las siguientes secciones.
Flujo Normal de Eventos	
Sección "Insertar paciente"	
Acción del Actor	**Respuesta del Sistema**
3 El CIC introduce los datos requeridos e indica guardarlos.	2 El sistema muestra una interfaz con los datos: identificador del paciente en el estudio, fecha de inclusión, sexo y fecha de nacimiento, que deben de ser introducidos por el CIC. 4 El sistema inserta el paciente con los datos introducidos por el CIC y le

	asigna “Evaluación” a su estado de tratamiento. 5 El sistema busca todas las definiciones de momentos de seguimiento correspondientes a la etapa “Evaluación”. 6 El sistema asigna al paciente nuevos momentos de seguimiento correspondientes a las definiciones de momentos de seguimiento de la etapa “Evaluación” y finaliza así el caso de uso.
Prototipo de Interfaz	
	
Flujo Normal de Eventos	
Sección “Mostrar datos de un paciente”	

Acción del Actor	Respuesta del Sistema
	2. El sistema busca los datos asociados al paciente y al estudio al que pertenece dicho paciente. 3. El sistema muestra una interfaz con los datos asociados al paciente y al estudio al que pertenece dicho paciente. 4 El sistema muestra además el cronograma específico del paciente, especificando de cada momento de seguimiento su estado, la fecha para la cual se planificó su llenado y la etapa a la que pertenece, de no estar generado muestra la sección “Generar cronograma específico” si es posible generarlo (Ver caso de uso Gestionar cronograma específico, sección Generar cronograma específico) y finaliza así el caso de uso.
Prototipo de Interfaz	



Flujo Normal de Eventos	
Sección "Interrumpir paciente"	
Acción del Actor	**Respuesta del Sistema**
	2 El sistema muestra al CIC un mensaje de confirmación de interrupción del paciente preguntándole si está seguro de interrumpir ese paciente.
3 El CIC indica que está seguro de	4 El sistema cambia el estado del

interrumpir ese paciente.	tratamiento del paciente a "Interrumpido". 5 El sistema deja de mostrar el plazo de tiempo de llenado correspondiente a cada momento de seguimiento programado asociado a este paciente. 6 El sistema deja de verificar el plazo de tiempo de llenado de cada momento de seguimiento asociado a este paciente evitando que tomen estado "Atrasado" y finaliza así el caso de uso.
Prototipo de Interfaz	
	
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
3a) El CIC indica que no está seguro de interrumpir ese paciente y finaliza así el caso de uso.	
Prototipo de Interfaz	
Poscondiciones	En dependencia de la sección que se ejecutó, queda introducido un nuevo paciente al estudio o se interrumpe el tratamiento de un paciente del estudio.

CU Gestionar datos de hoja.

Caso de Uso:	Gestionar datos de hoja
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el CIC indica una de las siguientes opciones: - Introducir datos a una hoja de CRD. - Mostrar datos de una hoja de CRD. - Modificar datos de una hoja de CRD. De acuerdo a la opción seleccionada el sistema se encarga de introducir a una hoja de CRD sus datos, mostrar los datos contenidos en una hoja de CRD o modificar sus datos.

Precondiciones:	
Referencias	R7, R8, R10, R11, R12
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
5 El CIC indica introducir datos a una hoja de CRD en el listado de hojas de CRD de un momento de seguimiento para realizar una de las siguientes opciones: - Introducir datos. - Modificar datos - Visualizar datos.	El sistema muestra información referente al momento de seguimiento y al paciente en cuestión y en dependencia de la presencia o no de los datos, y de la intención del actor, el sistema ejecuta una de las siguientes secciones.

Flujo Normal de Eventos	
Sección "Mostrar datos"	
Acción del Actor	**Respuesta del Sistema**
	2 El sistema busca los datos introducidos por el CIC anteriormente. 3 El sistema muestra los datos

	introducidos por el CIC anteriormente y finaliza así el caso de uso.

Prototipo de Interfaz

Flujo Normal de Eventos

Sección "Introducir datos"

Acción del Actor	Respuesta del Sistema
3 El CIC introduce los datos requeridos en la hoja de CRD e indica guardar.	2 El sistema muestra una interfaz con los campos a llenar en la hoja de CRD por el CIC. 4 El sistema guarda los datos introducidos por el CIC. 5 El sistema actualiza el estado de la hoja de CRD a "Iniciada" o "Completada" según la decisión del CIC. 6 El sistema comprueba que el momento de seguimiento al cual pertenece la hoja de CRD corresponde a una nueva etapa del estudio. 7 El sistema cambia el estado de tratamiento del paciente a "Tratamiento" o "Seguimiento" según sea la etapa correspondiente. 8 El sistema comprueba el estado de todas las hojas de CRD asociadas al momento de seguimiento actual. 9 El sistema le establece al momento de seguimiento el menor de los estados de las hojas de CRD asociado a él, que sea distinto de "No Iniciado" y finaliza así el caso

	de uso.

Prototipo de Interfaz

Flujos Alternos

Acción del Actor	Respuesta del Sistema
	7a) Si momento de seguimiento al cual

	pertenece la hoja de CRD no corresponde a una nueva etapa del estudio el sistema no cambia el estado de tratamiento del paciente y continúa en el paso 8.
Flujo Normal de Eventos	
Sección "Modificar datos"	
Acción del Actor	**Respuesta del Sistema**
3 El CIC introduce los datos a modificar en la hoja de CRD e indica guardar.	2 El sistema muestra una interfaz con los campos ya introducidos por el CIC. 4 El sistema actualiza los datos introducidos por el CIC. 5 El sistema actualiza el estado de la hoja de CRD a "Iniciada" o "Completada" según la decisión del CIC y finaliza así en caso de uso.
Poscondiciones	En dependencia de la sección que se ejecute, se le introducen datos a una hoja de CRD o se le modifican los datos que ya tenía, se le establece un nuevo estado a la hoja de CRD, al momento de seguimiento y al tratamiento del paciente.

CU Gestionar notas de sitio.

Este CU se le agregará al sistema para cumplir con las necesidades de la conducción de EC en Cuba.

Caso de Uso:	Gestionar Notas de Sitio
Actores:	CIC (inicia)
Resumen:	El caso de uso inicia cuando el Coordinador de Investigación Clínica indica: - Crear nota de sitio - Mostrar nota de sitio
Precondiciones:	
Referencias	R13, R14
Prioridad	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El CIC indica realizar una de las siguientes opciones sobre un dato en el llenado de una hoja de CRD: - Crear nota del sitio. - Ver nota del sitio.	En dependencia de cuál sea la opción indicada por el CIC el sistema ejecuta una de las siguientes secciones.
Flujo Normal de Eventos	
Sección "Crear nota de sitio"	

Acción del Actor	Respuesta del Sistema
3 El CIC inserta los datos e indica enviar la nota de sitio.	2 El sistema muestra una interfaz con los campos a insertar en la nota de sitio: “Descripción” y “Detalles de la Nota”. 4 El sistema guarda los datos referente a la nota y le establece el tipo de nota “Anotación”.
Prototipo de Interfaz	
	
Flujo Normal de Eventos	
Sección “Mostrar nota de sitio”	
Acción del Actor	**Respuesta del Sistema**
	2 El sistema busca todos los datos referentes a esa Nota del Sitio. 3 El sistema muestra todos los datos

	de la Nota del Sitio al CIC.
Poscondiciones	En dependencia de cuál sea la sección que se ejecute, queda creada una nueva nota de sitio asociada a una hoja de CRD de tipo.

Anexo 2. Diagramas de clases del diseño.

Caso de uso: Gestionar pacientes.

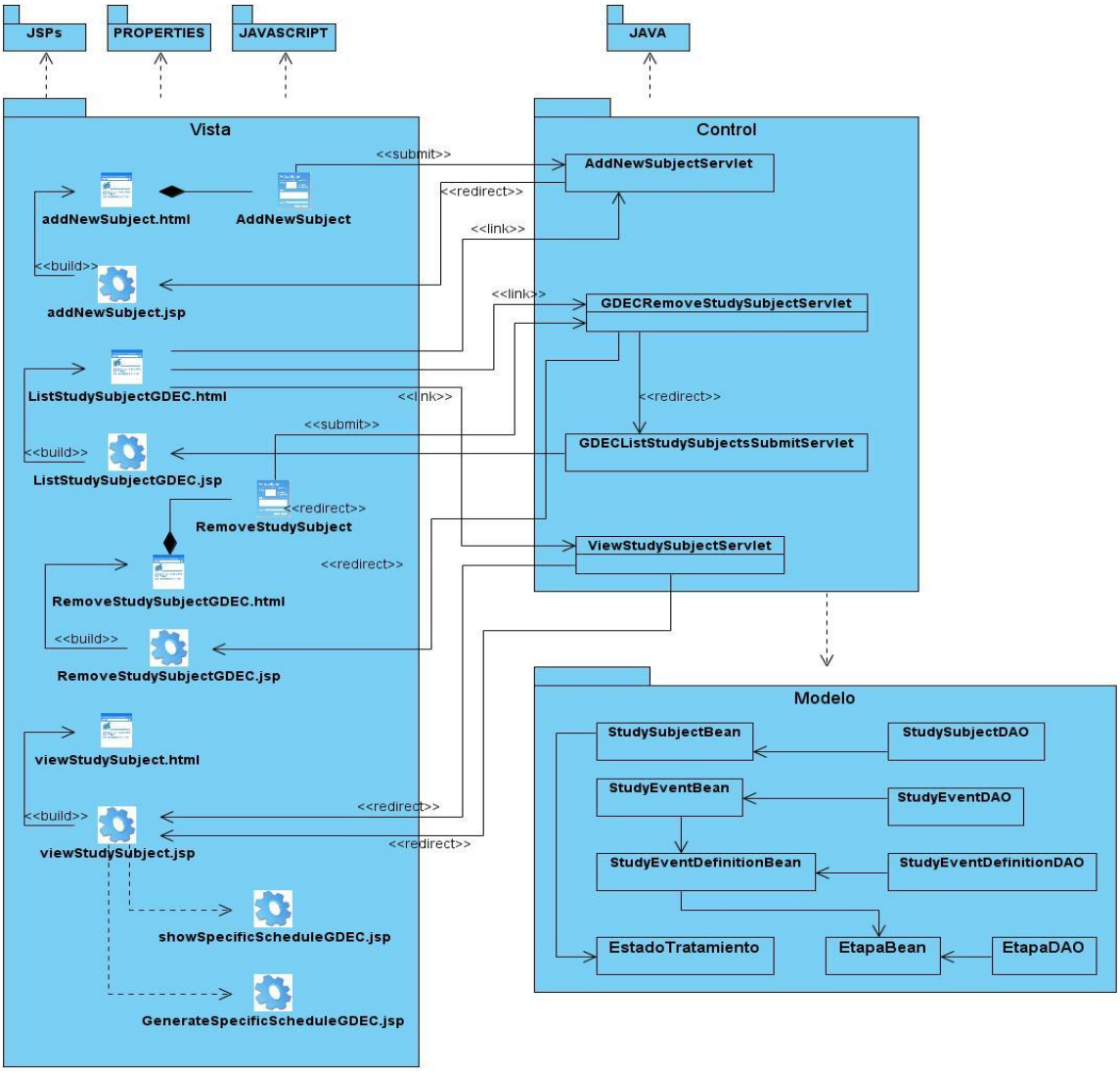


Fig. # 37 Diagrama de clases de diseño "Gestionar paciente".

Caso de uso: Gestionar datos de hojas

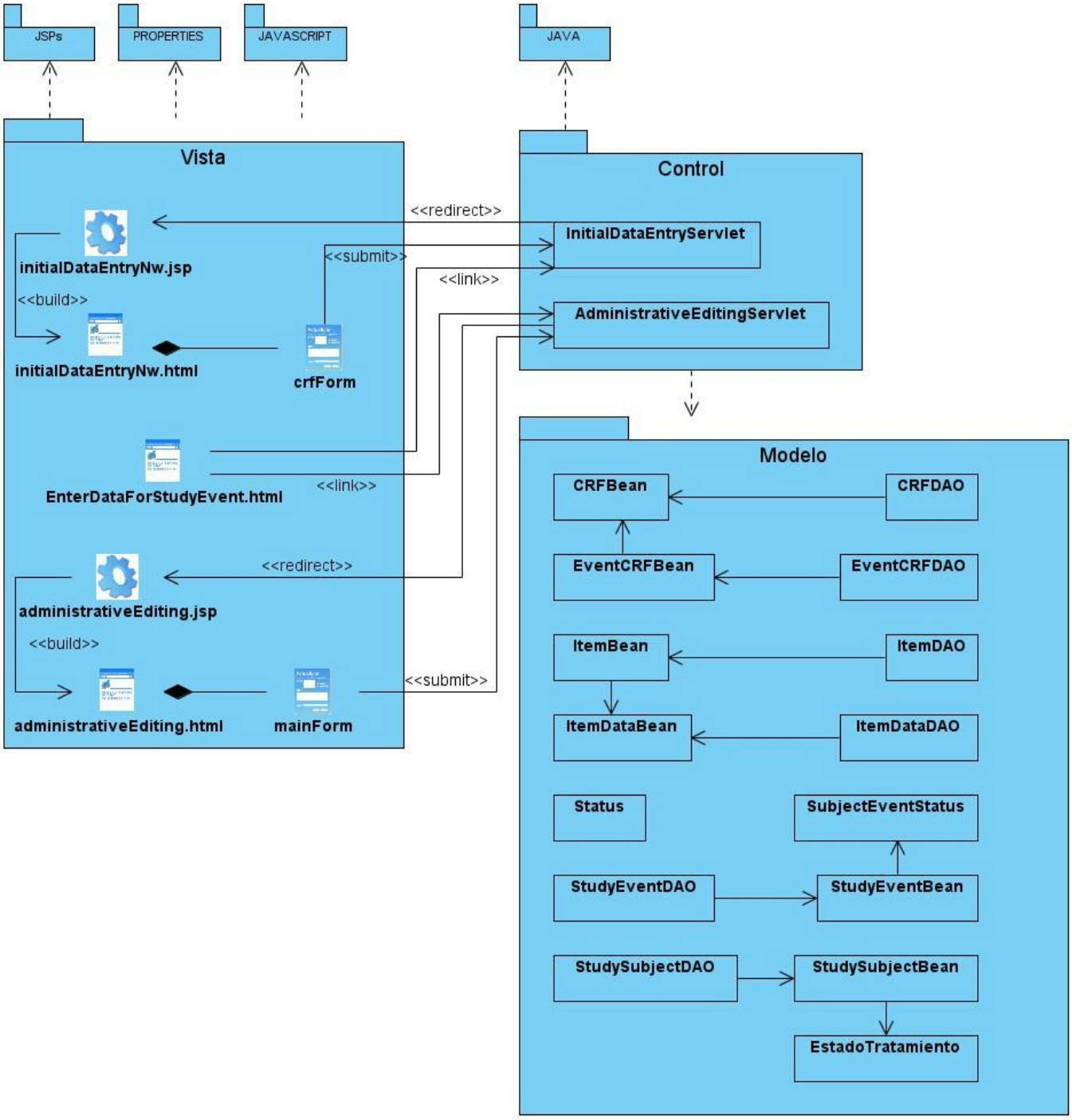


Fig. # 38 Diagrama de clases de diseño "Gestionar datos de Hoja de CRD".

Caso de uso: Gestionar notas de sitio

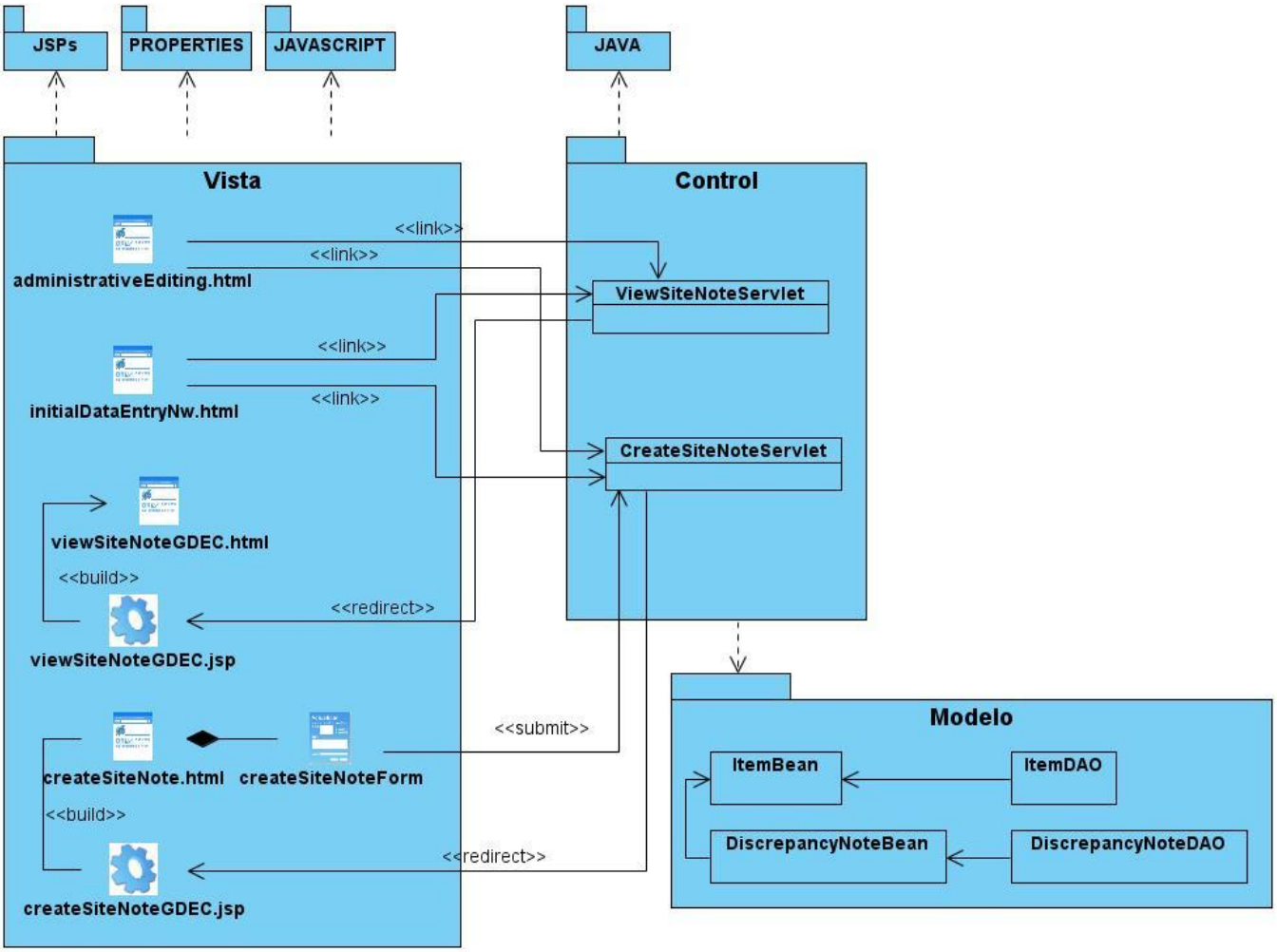


Fig. # 39 Diagrama de clases del diseño "Gestionar notas del sitio.

Anexo 3. Diagramas de secuencia.

Caso de uso: Gestionar paciente.

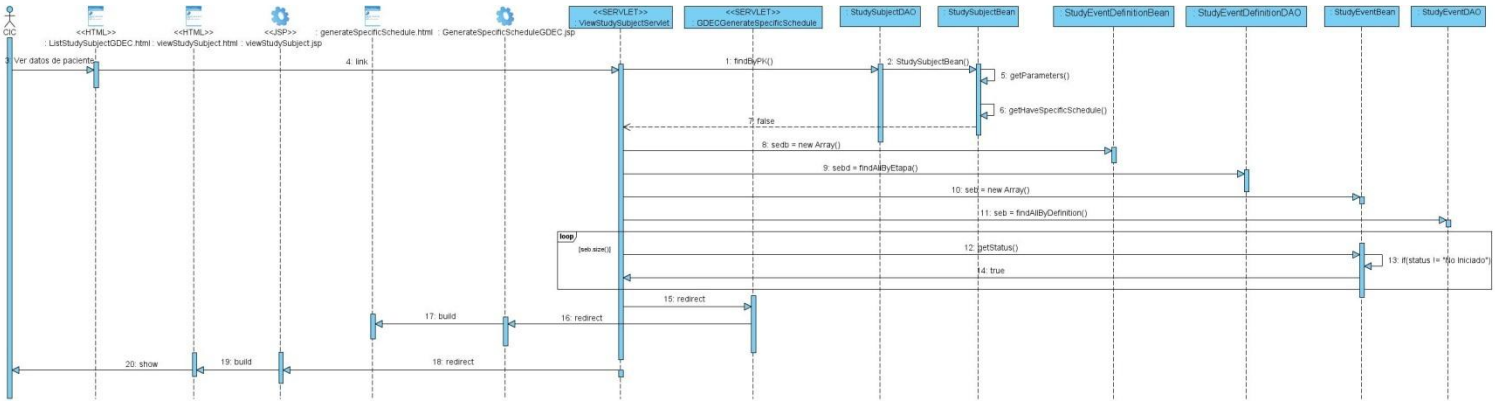


Fig. # 40 Diagrama de secuencia "Gestionar paciente". Sección mostrar datos de paciente sin cronograma específico.

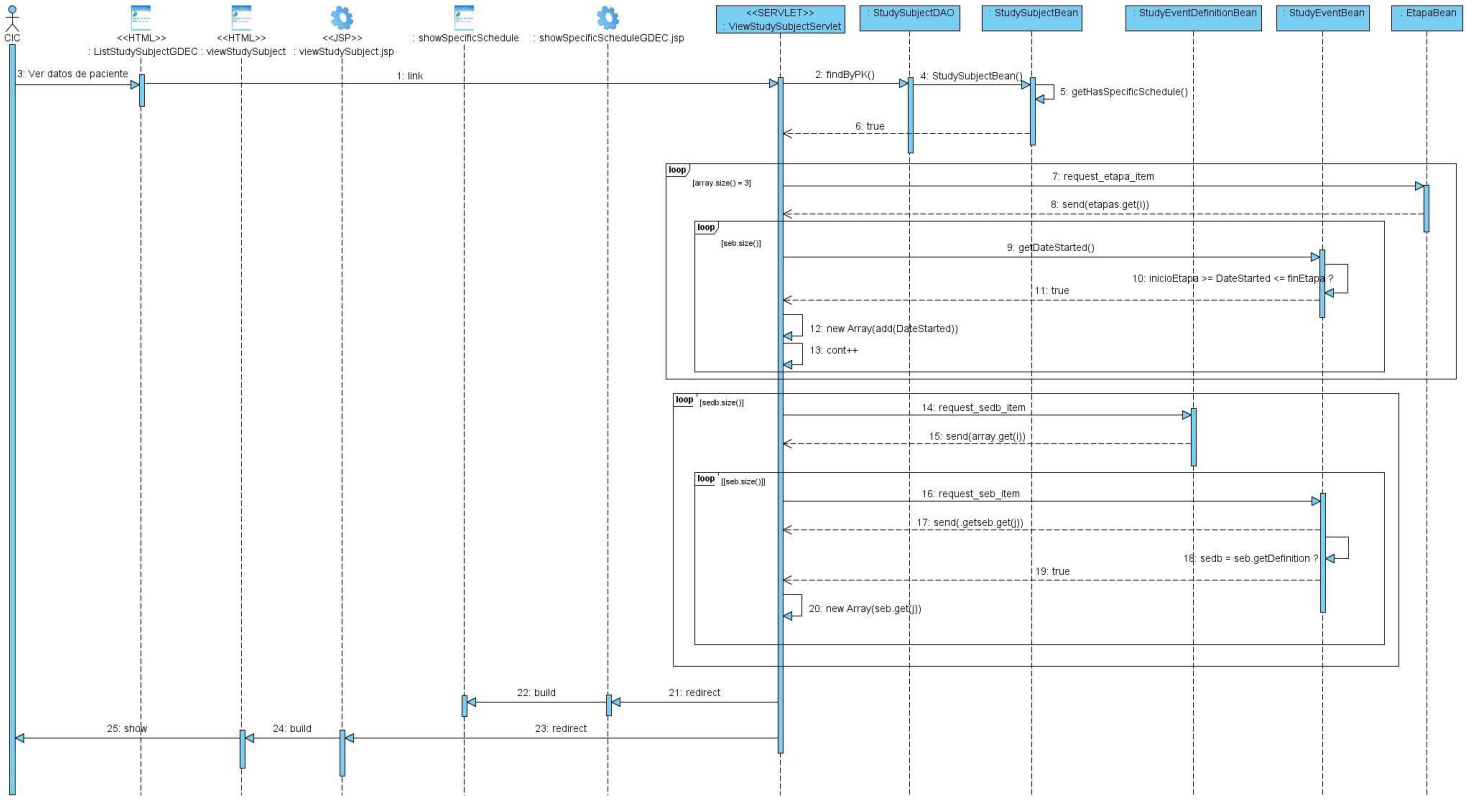


Fig. # 41 Diagrama de secuencia "Gestionar paciente". Sección mostrar datos de paciente con cronograma específico.

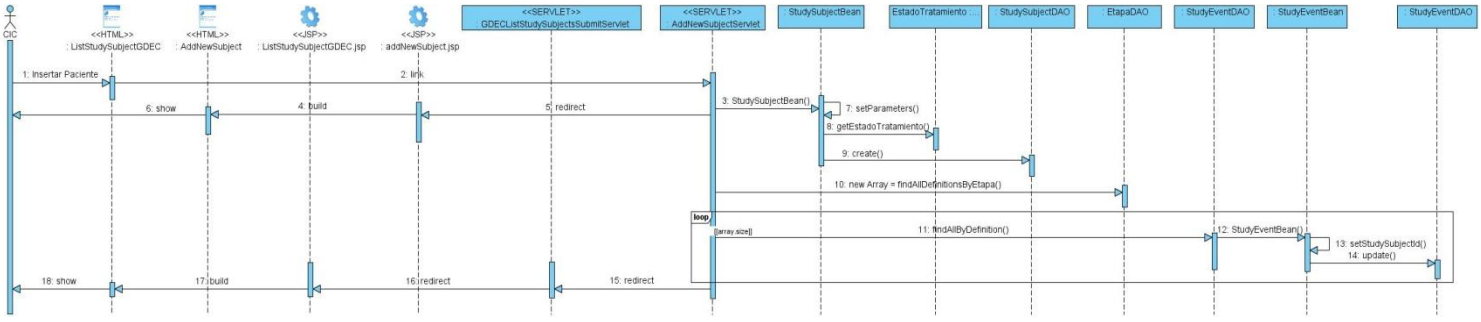


Fig. # 42 Diagrama de secuencia "Gestionar paciente". Sección insertar paciente.

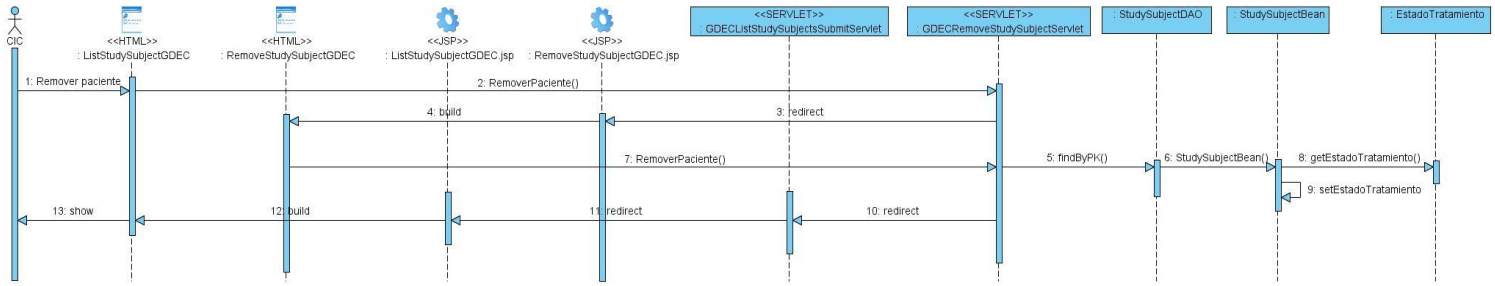


Fig. # 43 Diagrama de secuencia "Gestionar paciente". Sección interrumpir paciente.

Caso de uso: Gestionar datos de hojas de CRD

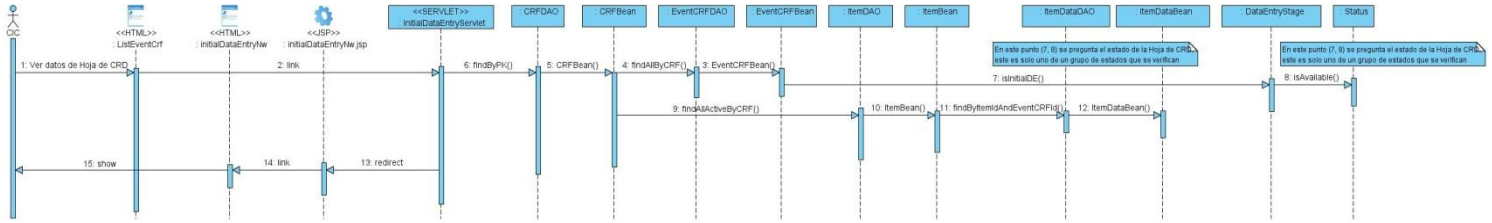


Fig. # 44 Diagrama de secuencia "Gestionar datos de hoja de CRD". Sección mostrar datos.

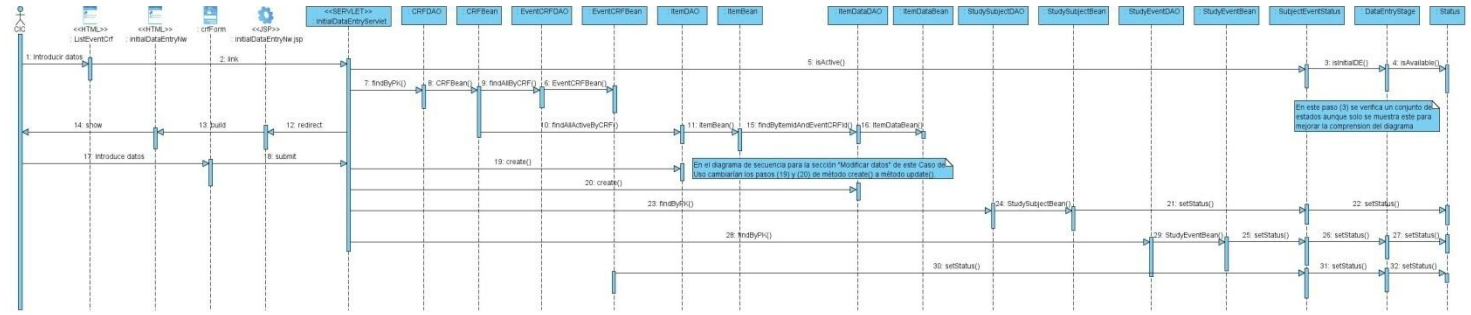


Fig. # 45 Diagrama de secuencia "Gestionar datos de hoja de CRD". Sección introducir datos.

Caso de uso: Gestionar notas de sitio

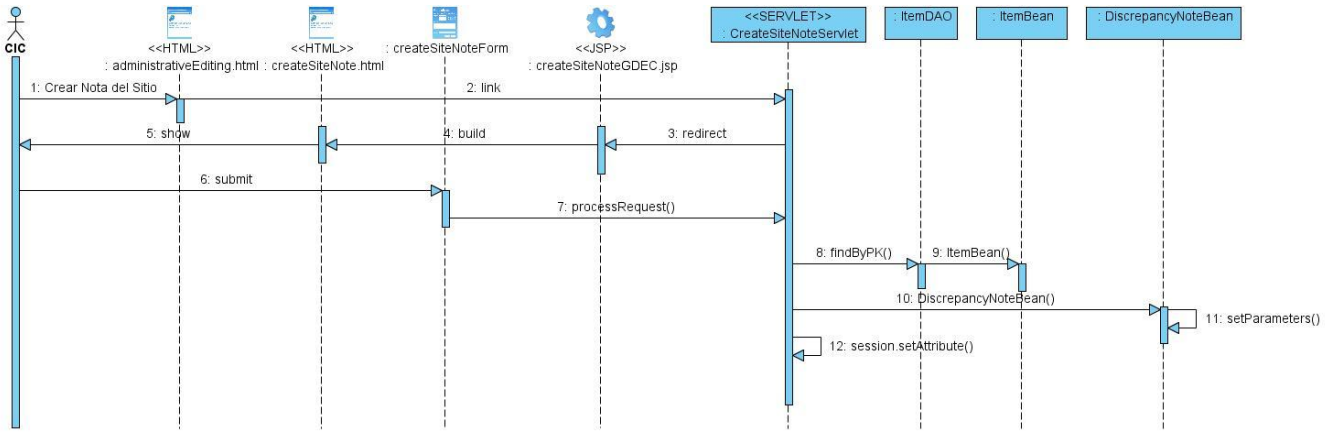


Fig. # 46 Diagrama de secuencia "Gestionar notas de sitio". Sección crear nota.

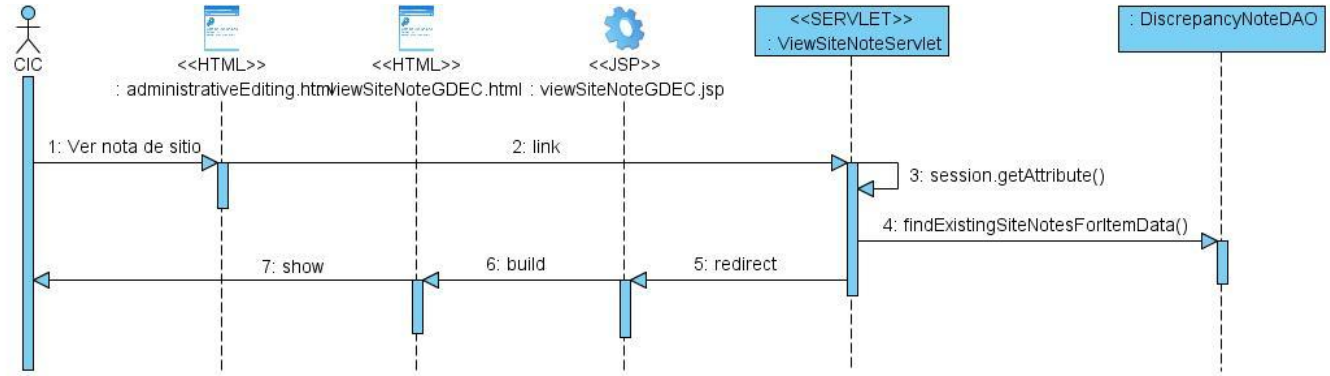


Fig. # 47 Diagrama de secuencia "Gestionar notas de sitio". Sección ver nota.

Anexo 4. Diagramas de componentes.

Caso de uso: Gestionar paciente

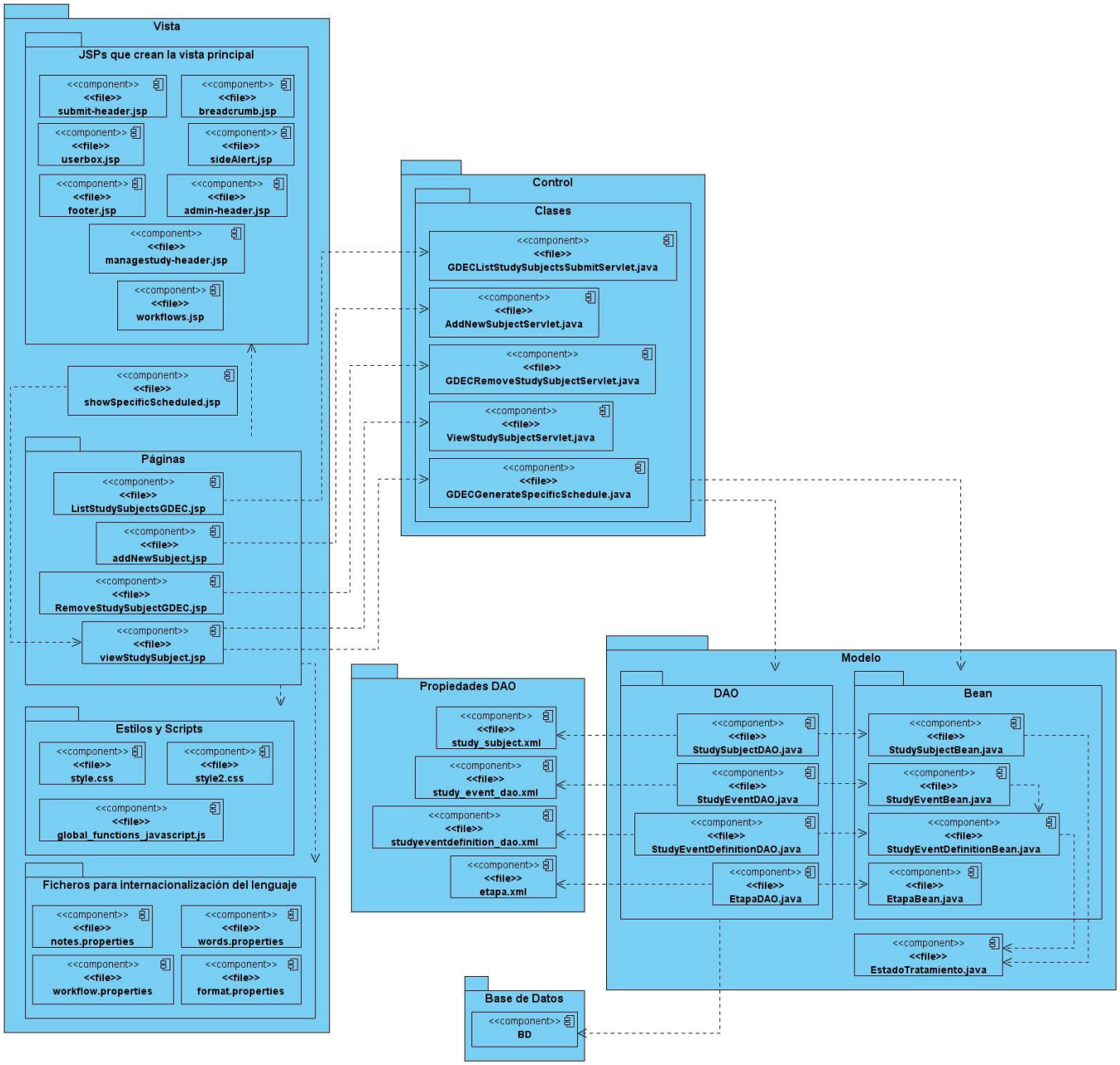


Fig. # 48 Diagrama de componentes "Gestionar paciente".

Caso de uso: Gestionar datos de hoja de CRD

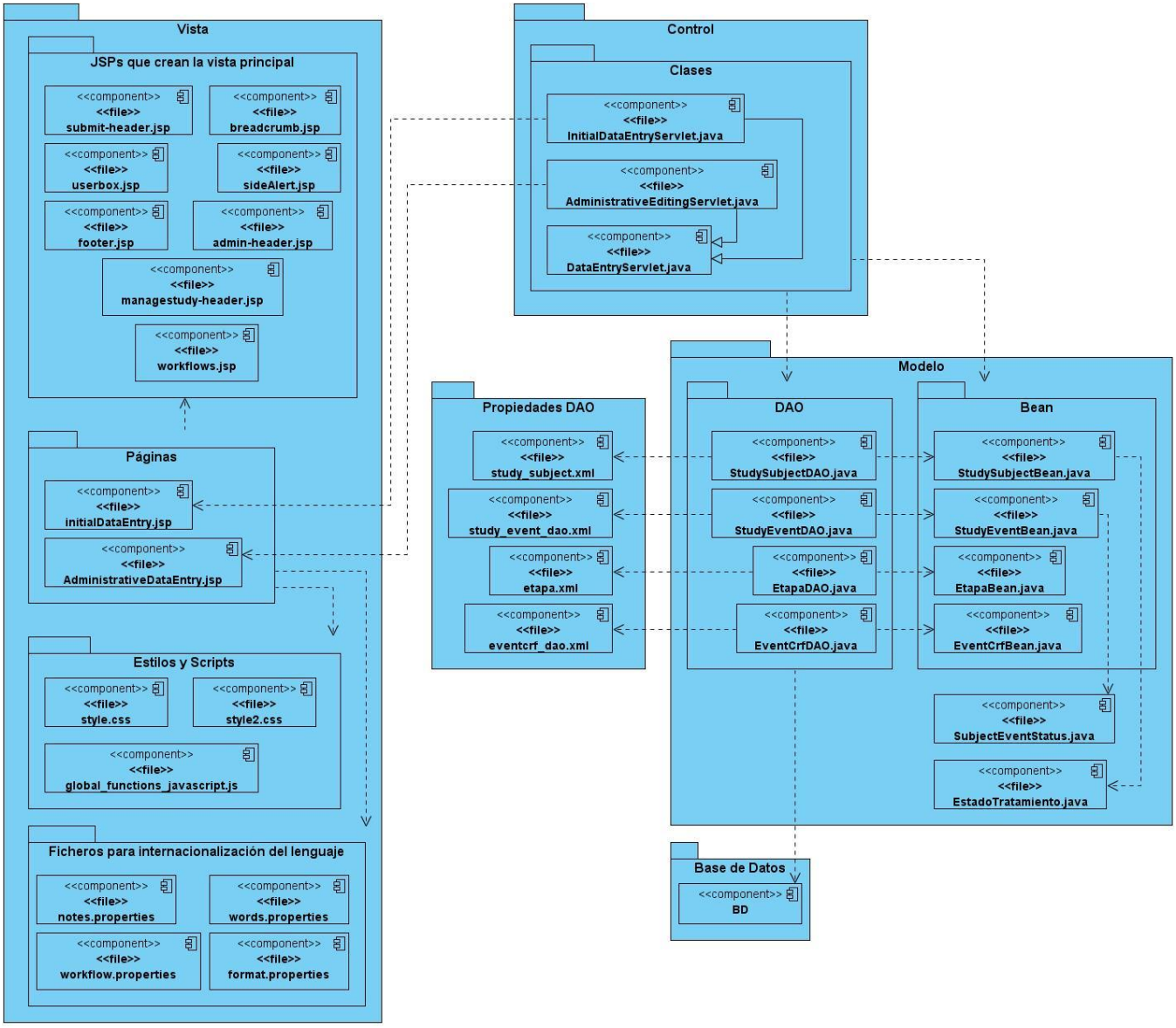


Fig. # 49 Diagrama de componentes "Gestionar datos de hoja de CRD".

Caso de uso: Gestionar nota de sitio

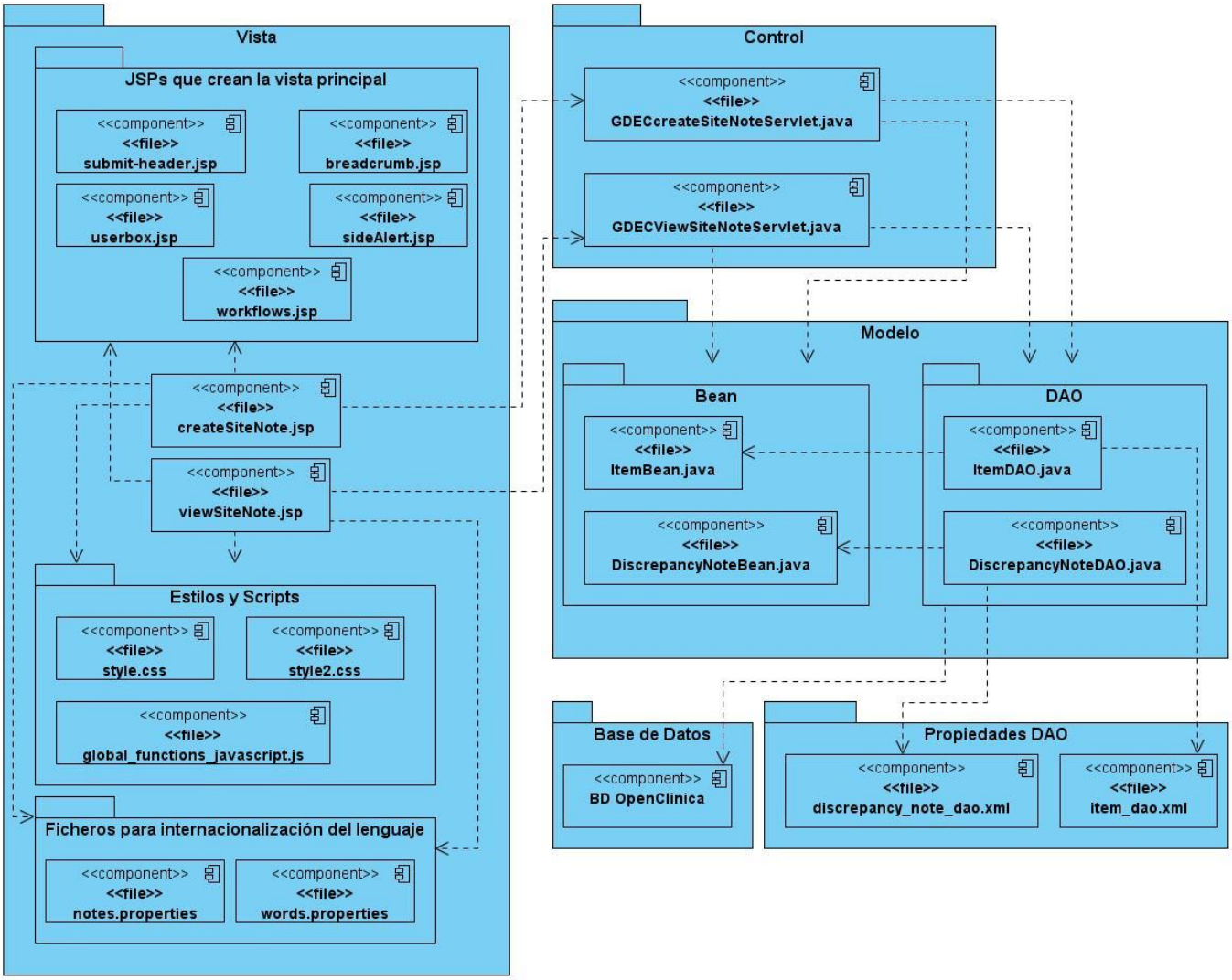


Fig. # 50 Diagrama de componentes "Gestionar notas del sitio".