Universidad de las Ciencias Informáticas
Facultad 6

Título: “alasClínicas:
Desarrollo de funcionalidades administrativas y de reporte de trazas para la adaptación del sistema OpenClínica a la ejecución de Ensayos Clínicos cubanos”

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

AUTORES: Lino Yohandy Ballagas Rodríguez
Nosbel Rivera González

TUTORES: Ing. Lucía Rodríguez García
Ing. Richard Díaz Pompa

Ciudad de la Habana
Julio 2009

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 _____.

Lino Yohandy Ballagas Rodríguez

Firma del Autor

Nosbel Rivera González

Firma del Autor

Lucía Rodríguez García

Firma del Tutor

Richard Díaz Pompa

Firma del Tutor

DEDICATORIA

Especialmente a Niurka y Lino, mis padres queridos que han sido los guías y la luz que ha iluminado mi camino desde el primer momento que abrí los ojos. Gracias por creer en mí y darme todo el apoyo que un hijo se merece. Sin ustedes no me hubiera sido posible ser el hombre que soy.

Lino

Con todo mi cariño para mis padres, mi hermana y mi querida esposa, que con su amor, respeto y apoyo he podido alcanzar todos mis logros.

Gracias por estar a mi lado durante todos estos años.

Nosbel

"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

AGRADECIMIENTOS

A la obra maravillosa de la Revolución por haberme dado esta oportunidad.
A mis padres por sus esfuerzos y sacrificios, por su apoyo incondicional y por toda la educación que me han dado. A Yoanna por estar a mi lado en esta última etapa de mi vida como universitario y ser tan paciente y comprensiva. A todos mis compañeros de estudios, a los que están y a los que ya nos volverán. A Yaneisy por haberme acompañado y darme todo su apoyo durante la mayor parte de la carrera. A mis amigos de la adolescencia Aylín, Cordero y Abdel. A mis mejores amigos Yoslainys, Yulieska, Nosbel, Luis Gabriel, Arsenio, Raul, Dioleisys, Yamilé, Linet, Dayana, Liusmila, Félix Mario, Michel, Norlen y a todos aquellos que de una forma u otra me han ayudado en todos estos años. A mis compañeros de proyecto, especialmente a aquellos que más me ayudaron. A Lucia, Richard, Ballester, Aidacelys por su apoyo y preocupación. A mis mejores profesores: Taymara, Yanelis por ser más que una profe, Alexei por enseñarme más que física, a Daulemys y a todos aquellos que pusieron su granito de arena en el desarrollo del presente trabajo.

Lino

A mis padres, hermana y esposa por darme su apoyo incondicional. A la revolución por darme la oportunidad de ser un hombre mejor. A toda mi familia. A todos mis compañeros de estudios y a todos aquellos que de una forma u otra han contribuido con esta obra.

Nosbel

DATOS DE CONTACTO

Tutor: Ing. Lucía Rodríguez.

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor Recién graduado en Adiestramiento.

Categoría Científica: no

Años de experiencia en el tema:

Años de graduado: 2007

Tutor: Ing. Richard Díaz Pompa

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor Recién graduado en Adiestramiento.

Categoría Científica: no

Años de experiencia en el tema:

Años de graduado: 2008

OPINIÓN DEL TUTOR

Resumen

El Centro de Inmunología Molecular lleva a cabo un conjunto de estudios denominados Ensayos Clínicos en los que se evalúan nuevos fármacos o tratamientos médicos, permitiendo a los especialistas determinar si un nuevo tratamiento, medicamento o dispositivo contribuirá a prevenir, detectar o tratar una enfermedad a través de un protocolo de investigación estrictamente controlado: los Cuadernos de Recogida de Datos; que constituyen formularios diseñados para anotar los datos recogidos durante el Ensayo. Este es personalizado para cada proyecto de investigación clínica concreto, donde actualmente no existe una estandarización. Por la necesidad de automatizar estos estudios y estandarizar los procesos de recogida de datos surge así en la Universidad de las Ciencias Informáticas el proyecto "Ensayos Clínicos": con el objetivo de desarrollar un producto capaz de conducir los ensayos clínicos adecuándose a la forma de hacer en Cuba. Para dar respuesta a las necesidades planteadas, el objetivo fundamental es desarrollar un conjunto de funcionalidades para dar solución a problemas administrativos y de reporte de trazas existentes en una aplicación estudiada y evaluada, que constituye una herramienta que se encarga de la conducción de ensayos clínicos según algunos estándares internacionales y cuyo nombre es OpenClínica. Estas funcionalidades tratarán de lograr una mayor compatibilidad entre esta aplicación y los estándares de conducción de ensayos clínicos cubanos.

Palabras Claves: Ensayos Clínicos, Cuadernos de Recogida de Datos.

INDICE

Índice de Figuras

Introducción

Temido por la población, estudiado por los científicos, el cáncer es un problema de salud mundial que nos agrede año tras año con su carga de angustias y esperanzas. Según la Organización Mundial de la Salud el cáncer es un término genérico para un grupo de más de 100 enfermedades que pueden afectar a cualquier parte del organismo. Otros términos utilizados son neoplasias y tumores malignos. Una de las características que define el cáncer es la generación rápida de células anormales que crecen más allá de sus límites normales y pueden invadir zonas adyacentes del organismo o diseminarse a otros órganos en un proceso que da lugar a la formación de las llamadas metástasis.

Cada año aparecen en el planeta unos 10 millones de casos nuevos y mueren por esta causa 7 millones de personas. Se espera que para el 2020 aumenten a 15 millones los casos nuevos, el 60% de ellos en países en desarrollo, que cuentan sólo con el 5% de los recursos destinados a combatir este mal. [1]
Según datos estadísticos de la Organización Mundial de la Salud (OMS):

- El cáncer es la primera causa de mortalidad a nivel mundial; se le atribuyen 7,9 millones de defunciones ocurridas en 2007 (aproximadamente un 13% del total).
- La mayor parte de la mortalidad anual por cáncer obedece a cáncer de pulmón, estómago, hígado, colon y mama.
- La frecuencia de los diversos tipos de cáncer varía según el sexo.
- Aproximadamente el 30% de las defunciones por cáncer son prevenibles.
- El consumo de tabaco es el principal factor singular de riesgo de cáncer.
- El cáncer comienza con una modificación en una sola célula. Ese cambio puede haber sido iniciado por agentes externos o por factores genéticos heredados.[2]

En Cuba hay más de 100 mil personas con algún tipo de enfermedad maligna y aparecen cada año unos 28 mil nuevos casos. Y en cuanto a la mortalidad, se sitúa en la segunda causa en la población en general, pero es la primera causa de muerte antes de los 74 años, o sea la causa que más afecta la esperanza de vida del cubano al nacer. [3]
Este fenómeno está presente en nuestra sociedad en un momento donde predomina la convicción de que el papel de la medicina no debe limitarse solo a alargar el tiempo de vida sino que, debe contribuir a mejorarlo, aportando mayor calidad al tiempo vivido. Para lograr esta premisa, en Cuba se han creado después del triunfo de la revolución varios centros biotecnológicos y científicos investigativos

asociados a la producción, dentro de los cuales se encuentra el Centro de Inmunología Molecular (CIM), creado el 5 de diciembre de 1994. [4]

Este centro desde sus inicios se ha dedicado al desarrollo de biomoléculas y otros fármacos para el tratamiento de diferentes enfermedades relacionadas con el sistema inmune, principalmente el cáncer. En el CIM después de realizar los análisis pertinentes para probar el funcionamiento de los fármacos, se aplican en pacientes que padecen diferentes enfermedades. Al proceso de aplicación de los productos en fase de prueba a los humanos, junto a la recuperación de todos los datos de la evolución del medicamento en ellos, es a lo que se llama Ensayo Clínico (EC), estos presentan un protocolo, documento que establece la razón de ser del estudio, sus objetivos, diseño, métodos y el análisis previsto de sus resultados, así como las condiciones bajo las que se realizará y desarrollará el estudio.

Por estas razones llevan asociados una gran documentación la cual debe mantenerse almacenada durante 15 años posteriores a la realización del mismo; para poder ser consultados por las agencias reguladoras internacionales. Esto implica un enorme cúmulo de información y por ello se torna engorroso el proceso para el manejo de la misma por parte de los especialistas.

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

En el mundo existe una tendencia a realizar los ensayos clínicos de manera electrónica por transmisión remota de datos, ya que con esto se optimiza tiempo y se puede trabajar con una mayor cantidad de pacientes. Por las ventajas que ha reportado esta nueva forma de realización de ensayos clínicos, el CIM conjuntamente con la Universidad de Ciencias Informática, comenzó el proyecto “Ensayos Clínicos”. Como resultado, se espera la obtención de un producto funcional que permita gestionar los procesos de Ensayos Clínicos de forma tal que se pueda realizar la transmisión remota de datos dentro de la Red Nacional de Hospitales hacia el Centro de Inmunología Molecular. Además, la actualización constante de las bases de datos y el procesamiento de la información clínica que de éstas se deriven. [5]

Con este objetivo la UCI realizó una investigación donde se encontró varias soluciones empresariales diseñadas para gestionar los ensayos clínicos, entre las cuales estaba el Sistema OpenClínica una solución software libre que después de pasar por un exhaustivo análisis primero en la Universidad y luego en el CIM, se decidió reutilizar y mejorar. Esto se debió a que el sistema a pesar que cumplía la expectativas del polo científico, no se estandarizaba con la forma de conducir los ensayos clínico en Cuba, por lo que se decidió generar una serie de funcionalidades generales que le dieran al sistema propiedad de adecuarse a las necesidades del país y así surgen un conjunto de requisitos funcionales asociados a funcionalidades de administración y de reporte, a los cuales se pretende dar solución con la presente investigación.

Dada la situación problemática expuesta anteriormente, se define el siguiente problema:

¿Cómo adaptar funcionalidades administrativas y de gestión de reportes de trazas, del Sistema OpenClínica al proceso de ejecución de Ensayos Clínicos cubanos?

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

Procesos de conducción de Ensayos Clínicos cubanos en el sistema OpenClínica.

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

Procesos relacionados con funciones administrativas y de reportes de trazas en el sistema OpenClínica.

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

Desarrollar funcionalidades para la gestión administrativa y el reporte de trazas en el sistema OpenClínica.

Para cumplir este objetivo general se trazaron los siguientes objetivos específicos:

- Analizar el sistema OpenClínica e investigaciones anteriores para la definición de requisitos funcionales.
- Definir las funcionalidades a desarrollar.
- Diseñar e implementar funcionalidades a desarrollar.

Se desarrollarán las siguientes tareas para dar cumplimiento a los objetivos trazados:

- Estudio del sistema OC
- Modelación del dominio
- Levantamiento de requisitos a adicionar al sistema OC
- 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 diagrama de clases persistentes
- Realización de los diagramas de componentes
- Implementación de las funcionalidades para el sistema

Estructura del documento:

Capítulo 1: Fundamentación teórica: en este capítulo se hace un análisis del estado del arte del objeto de estudio, se investiga acerca de los sistemas informáticos vinculados al campo de acción, se fundamentan las herramientas, metodología y tecnologías utilizadas para el desarrollo de la aplicación informática.

Capítulo 2: Características del Sistema: en este capítulo se define el modelo a seguir para el desarrollo de la aplicación. Se definen los requisitos funcionales como no funcionales. Se documentan los patrones de caso de uso y se definen los mismos, son sus respectivas descripciones y los artefactos asociados.

Capitulo 3: Diseño del Sistema: en este capítulo se describen estilos arquitectónicos y los patrones de diseño evidenciando su utilización. Recoge los artefactos correspondientes al flujo de diseño dentro de los que se encuentran: diagramas de clases del diseño, diagramas de secuencia del diseño, modelo de despliegue, modelo de datos con las respectivas clases persistentes así como una breve descripción de cada uno de los diagramas realizados y las clases más relevantes.

Capitulo 4: Implementación del Sistema: en este capítulo se modelan los diagramas de componentes correspondientes a cada uno de los casos de usos que se desarrollan, implementaciones relevantes que constituyen aporte a la investigación y algunas pantallas que representen la aplicación.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se abordan definiciones y conceptos importantes que están presentes durante toda la investigación y se hace un análisis sobre el estado del arte de los procesos de gestión de diseño de los Cuadernos de Recogida de Datos a nivel mundial. Se fundamenta el uso de las distintas herramientas, tecnologías y metodologías utilizadas haciendo una comparación con otras muy difundidas en el contexto internacional.

1.1 Ensayos Clínicos

Los ensayos clínicos son una parte fundamental en el proceso de desarrollo, aprobación e introducción en el mercado de nuevos fármacos y tratamientos contra el cáncer. 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.

Cada uno de estos protocolos tiene unos determinados requerimientos o criterios de inclusión que deben cumplir todos los pacientes que vayan a participar. En algunos casos se trata de un tipo de cáncer concreto, o una fase de la enfermedad determinada; en otros, existen criterios excluyentes que impiden la participación de ciertos grupos de edad o subgrupos de pacientes con características determinadas.
Los ensayos son desarrollados por fases:

Fase I: en esta etapa se estudia la seguridad de un nuevo fármaco, cómo se debe administrar, con qué frecuencia y cuál es la dosis máxima tolerada, es decir, cuál es el umbral a partir del cual el medicamento se convierte en una sustancia peligrosa. Generalmente este tipo de ensayos están restringidos a pacientes con tumores en fases avanzadas. En otras especialidades, diferentes de la oncología, los ensayos en fase I se llevan a cabo con voluntarios sanos.
Fase II: estos ensayos añaden cierta información a los anteriores con respecto a la actividad del fármaco y sus efectos en un determinado tipo de tumor. Según la Sociedad Americana de Cáncer, si el 20% de los participantes responde a la nueva terapia, se seguirá avanzando en la investigación.

Fase III: en estos casos se compara un nuevo tratamiento con la terapia estándar que viene administrándose de forma habitual en la práctica clínica. Los pacientes se asignan aleatoriamente a uno de los dos grupos, de manera que son tratados bien con el nuevo o con el viejo fármaco.

Fase IV: estos ensayos están diseñados con la intención de conocer más datos sobre la dosis o administración de un determinado fármaco, los efectos secundarios que puede provocar etc. Estos ensayos también pueden comparar dos tratamientos diferentes que están aprobados para usos diferentes, de manera que los científicos puedan determinar cuál de los dos es más eficaz en el tratamiento de un determinado tipo de tumor. Tanto esta etapa como la anterior deben llevarse a cabo obligatoriamente con medicamentos registrados.

En el caso de las dos primeras fases, una vez que el ensayo ha finalizado se revisan todos los datos obtenidos y se toma una decisión respecto a seguir su evolución o interrumpir el proceso investigador. En fases posteriores, los resultados de los ensayos suelen darse a conocer en revistas médicas o congresos especializados.
Para su aprobación por las autoridades sanitarias y su posterior comercialización, un fármaco debe pasar obligatoriamente por las tres primeras fases. [6]

1.2 Cuadernos de recogida de datos

El Cuaderno de Recogida de Datos (CRD) ó Case Report Form (CRF) por sus siglas en inglés, es un formulario, con espacios definidos, diseñado para anotar las variables recogidas durante un ensayo clínico. El documento, siempre personalizado para un proyecto de investigación clínica concreto, permite registrar todos los datos que se solicitan en el protocolo y, según las normas de buena práctica clínica, debe ser reflejo de la documentación clínica original (historia clínica, informes de laboratorio, exploraciones complementarias), individual para cada paciente o voluntario y debe facilitar la reproducción de los datos generados, su tratamiento estadístico, la preparación y corrección del informe final de un ensayo clínico y, en definitiva, la sustentación científica de un nuevo medicamento o producto sanitario. [7]

La elaboración de documentos confusos y poco manejables limita, por no decir imposibilita, llegar a conclusiones válidas, a partir de datos que puedan haber necesitado meses o años de trabajo, en centenares o miles de pacientes. Por todo ello, su diseño y elaboración es un proceso que requiere una gran inversión temporal y de mucho cuidado.

Actualmente estos cuadernos se confeccionan en soporte de papel lo cual dificulta y retrasa el proceso del ensayo clínico, por tal motivo se hace necesaria la confección o el diseño de los cuadernos de recogida de datos en soporte electrónico los cuales se rigen por los mismos protocolos, pero además poseen notables ventajas:

- La comprobación de los datos puede ser implementada durante la entrada de los mismos, evitando así posibles errores.
- Eliminación de incongruencias en los datos.
- Estandarización de la información recogida.
- Recopilación organizada y específica de gran cantidad de información.
- La entrada de los datos puede realizarse de forma remota dado los avances y tecnologías actuales.
- Recoge la información correcta que da respuestas a las preguntas del estudio y que es consecuente con el protocolo.
- Organiza los etiquetas de los formularios y los campos de los mismos logrando que la entrada de los datos sea intuitiva
- Evita la duplicación de los datos: información repetitiva como los datos demográficos del paciente, número del estudio, fecha etc.… pueden ser replicados a partir del primer formulario hacia todos los demás. Por esta razón nuestra investigación va en encaminada al diseño y la implementación de una herramienta encargada de gestionar los procesos de diseño de los CRD y de esta forma lograr la buscada estandarización y centralización de los datos.

1.3 Soluciones existentes vinculadas a los Ensayos Clínicos

En la actualidad han surgido diversas soluciones empresariales diseñadas para gestionar los ensayos clínicos de una forma más fácil y rápida.

1.3.1 Entrypoint Plus

Es un sistema de captura de datos electrónicos altamente probado diseñado para crear, desarrollar y administrar ensayos clínicos. Caracterizado por una avanzada, escalable y multi-hilos arquitectura cliente-servidor sobre protocolo TCP/IP, con conectores ODBC para interconectarse con bases de datos SQL, condición necesaria para ensayos clínicos con sitios remotos. Es una aplicación de escritorio, es decir no es implementado para trabajar sobre la Web.

Entrypoint Plus tiene incluido un conjunto de plantillas de CRDs electrónicos con campos para datos demográficos, historia médica, reacciones adversas y exámenes de laboratorios. Soporta la entrada remota de datos evitando de esta forma el papel e introduciendo los datos directamente en la base de datos. Además posee opciones sofisticadas de seguridad y permite la exportación de la información a varios formatos que incluye Microsoft Excel (CSV), Access y XML. Es fácil de usar lo que significa emplear menos tiempo en el desempeño de los ensayos y en la construcción de los cuadernos de recogía de datos así como también la disminución de los costos de desarrollo. Los clientes que usan Entrypoint pueden rápidamente crear una amplia variedad de aplicaciones de entrada de datos adaptadas a las necesidades de los mismos y permitiendo la validación de los campos durante la construcción.
Las plantillas que incluye están organizadas de forma tal que cada una de ellas concentra un tipo específico de datos. Algunas pueden ser usadas sin ningún tipo de modificaciones para ensayos típicos. Otras pueden necesitar pocos ajustes para hacerlas compatibles con el estudio del cliente.

Es un software propietario diseñado por una compañía norteamericana llamada “Phoenix Software International” situada en Los Ángeles, en el estado California de los Estados Unidos. No es multiplataforma pues está pensado solo para Windows. A pesar de cumplir con requisitos como la creación de CRDs de forma electrónica, no es factible para el país debido a la actual situación económica en la que se encuentra.

1.3.2 Macro

Macro, es una de las soluciones y servicios que ofrece InferMed para el manejo y colección de datos que se diseñan para mejorar los resultados y sacar fármacos utilizados en los ensayos clínicos, que desde julio de 1999, esta tecnología se utiliza en más de 400 sitios en 36 diversos países a través del mundo. Los datos clínicos de los ensayos que se han recogido usando MACRO ascienden a los 80.000 pacientes.

Pretende llevar a cabo una red europea unificada para la entrada alejada de datos en ensayos clínicos para investigaciones. Proporciona medios eficientes para la comunicación entre un ensayo y los clínicos del hospital que tratan directamente a los pacientes. Realza la velocidad y la calidad de los ensayos clínicos y se espera que este sistema desarrollado en el campo del cáncer se extienda a otras ramas de la medicina como los estudios del SIDA. Brinda soporte para todos los ensayos, durante

todas las fases, desde un solo sitio, hasta grandes ensayos internacionales. InferMed trabaja en conjunto con las principales compañías farmacéuticas, para implementar MACRO como una solución global para la recolección electrónica de datos.

Esta herramienta permite el diseño del cuaderno de recogida de datos (CRD) de acuerdo al estudio clínico que vaya a realizarse y la validación de los datos que estarán incluidos en dicho estudio. Es uno de los sistemas más completos para los procesos de diseño, pero no permite la estandarización de los CRD, cualquier usuario puede diseñar un CRD según criterios propios, nombrando las variables que recogerá como desee sin tener una plantilla como guía. Para su adquisición es necesario un contrato con InferMed

Ltd, pues poseer las licencias de configuración del servidor, del uso de cada uno de los módulos de Macro, por cada estación de trabajo, el mantenimiento, entrenamiento, instalación incurriría en un gasto total aproximado de €38,900 con una distribución de los gastos de la siguiente manera:

- €28,000 por el paquete base de la aplicación, que está compuesto por:
- 1 licencia de configuración del servidor
- Licencias de Diseño del Estudio (Study Design). Estas licencias son de usuarios concurrentes.
- 8 Licencias de Manejo de Datos (Data Management). Cada una tendrá un valor de €500. Esta licencia puede ser usada para la entrada de datos on-line, entrada remota desde Windows o para el monitoreo del estudio.
- Las licencias son por estación de trabajo. Cada licencia permite a los usuarios de cada estación, participar en cualquier cantidad de ensayos.
- El honorario anual de mantenimiento se puede pagar por adelantado, incurre en un gasto de €5,600 o el 20% del total del costo de la licencia.
- €4,800 por ofrecer un servicio de entrenamiento e instalación.

En los gastos anteriormente mencionados no se incluyen las licencias para el uso de las herramientas que utiliza este producto entre ellas SQL Server como gestor de base de datos y otras para su uso de manera legal.[8]

1.3.3 OpenClínica

OpenClínica es disponible libremente, es una plataforma de software código abierto, basado en la web, para la gestión de estudios de investigación clínica. Tiene características para configuración de protocolo,

diseño de Cuadernos de Recogida de Datos (CRDs), captura, almacenamiento y gestión de datos electrónicos (CDE), recuperación y gestión de datos clínicos. OpenClínica es escalable, modular y basado en estándares. Presenta diferentes niveles de acceso a lectura, creación, y modificación de los datos.

Los módulos primarios de la aplicación incluyen:

- **Gestión de estudio:** Facilita la configuración y gestión de los protocolos de ensayo clínico, los sitios, los CRD, los usuarios y las definiciones de eventos de estudio. Usted puede definir los elementos de datos, los CRD, y eventos de protocolo sin ningún tipo de programación.
- **Enviar datos:** Proporciona una interfaz fácil de usar basada en la web para la inscripción de pacientes, el envío electrónico de datos y validación de datos.
- **Extraer datos:** Permite la definición, el filtrado, y la extracción de datos de estudio.
- **Administrar el Sistema:** Permite sistema global de supervisión, auditoría, configuración, gestión de cuenta de usuario, y la presentación de informes por los administradores.

Algunas características de OpenClínica incluyen:

- Organización de la investigación clínica por el protocolo de estudio y el sitio, cada uno con su propio conjunto de usuarios autorizados, pacientes, definiciones de eventos de estudio, y CRDs. Apoyo para el intercambio de recursos entre los estudios en un lugar seguro y transparente.
- Generación dinámica de CRD basados en la Web para la captura electrónica de datos definidos por el usuario a través de parámetros clínicos y lógica de validación especificados en las plantillas de Excel.
- Gestión de los datos longitudinales de complejas y recurrentes visitas de pacientes.
- Herramientas de importación y exportación de datos para la migración de bases de datos clínicos.
- Amplias interfaces para la consulta y recuperación de datos a través de los pacientes, el tiempo y los parámetros clínicos, con exportación de datos en los formatos comunes de archivo como el delimitado por tabuladores, SPSS, XML y CDISC ODM.
- Permite el cumplimiento con las normativas como la norma 21 CFR Parte 11. Las características incluyen diferentes funciones y privilegios de usuario, contraseña y la autenticación de usuario de

seguridad, firma electrónica, encriptación SSL, identificación de Información de Salud Protegida (PHI), y auditoría global para registrar y vigilar el acceso de datos y cambios.

- Una robusta y escalable infraestructura tecnológica desarrollada en Java utilizando J2EE framework con bases de datos relacionales incluyendo PostgreSQL (fuente abierta) y Oracle 10G, para apoyar las necesidades de la empresa de investigación clínica.
- La aplicación puede ser desplegada sobre Linux ó Windows. Fue desarrollada para Apache Jakarta Tomcat 5.x, este contendor web soporta las especificidades de la comunidad Java para Servlet 2.x y además Java Server Pages 2.

Luego de una intensa investigación se encontraron una serie de características (incluidas algunas de las anteriormente citadas), que dado su trascendencia le brindaban al sistema OpenClínica las cualidades necesarias para cumplir con las expectativas planteadas en dicha investigación, a diferencia de las otras soluciones empresariales estudiadas, este sistema puede aplicarse a las condiciones cubanas para la conducción de ensayos clínicos. A continuación se detallan algunas de estas características:

OpenClínica es software libre: puede redistribuirse y/o modificarse bajo los términos de la GNU Lesser General Public License publicada por la Fundación de Software Libre, desde la versión 2.1 de la Licencia, hasta (a su elección) cualquier versión posterior. Este software se distribuye con la esperanza de que sea útil, pero sin ninguna garantía, incluso sin la garantía implícita de comerciabilidad o idoneidad para un propósito particular.

La GNU Lesser General Public License es una licencia de Software Libre y al igual que cualquier otra licencia Software Libre, concede las siguientes cuatro libertades:

1. La libertad de ejecutar el programa para cualquier propósito.
2. La libertad de estudiar cómo funciona el programa y adaptarlo a necesidades específicas.
3. La libertad de distribuir copias, con lo que puede ayudar a su vecino.
4. La libertad de mejorar el programa y poner sus mejoras a disposición del público, para beneficio de toda la comunidad.

Se podrán ejercer las libertades especificados aquí siempre que se cumpla con las condiciones de esta licencia. La LGPL se destina a las bibliotecas de software, en lugar de programas ejecutables.

Las principales condiciones son:

- Se debe de forma adecuada y bien visible publicar en cada copia que se distribuya un anuncio de copyright adecuado y mantener intactos todos los anuncios que se refieran a esta Licencia y a la ausencia de garantía, y proporcionar a cualquier otro receptor del programa una copia de la GNU Lesser General Public License junto con el Programa. Cualquier traducción de la GNU Lesser General Public License debe ir acompañada por la GNU Lesser General Public License.
- Si se modifica la copia o copias de la biblioteca o de cualquier porción de ella, se puede distribuir la biblioteca resultante siempre y cuando se haga bajo la GNU Lesser General Public License. Sin embargo, los programas que se enlacen a la biblioteca podrán ser objeto de licencias en los términos de la elección de cada cual, siempre y cuando la biblioteca se pueda cambiar.

OpenClínica es Multiplataforma: Este término es usado para referirse a que el programa puede funcionar o ejecutarse en diversas plataformas, que son una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo como por ejemplo Windows o GNU/Linux.

Además de esto la aplicación es multilingüe: lo que significa que se puede acceder a ella en dependencia de su lengua natal, (hasta el momento solo soporta Español, Inglés y Francés).

A pesar de todo lo antes mencionado, OpenClínica presentó una serie de inconsistencias que derivaron en varias funcionalidades que el sistema no respondía o respondía parcialmente, debido a todo esto se toma la decisión de modificar el código fuente de la aplicación y así surgen un conjunto de requisitos asociados a funcionalidades administrativas y de reporte de trazas, a los cuales se pretende dar solución con la presente investigación.

1.4 Herramientas, tecnologías y metodologías

Para el desarrollo de la solución informática fue necesario el estudio de de diferentes herramientas, tecnologías y metodologías para seleccionar la que más se adecuaba a las necesidades y características para el proceso de implementación de la aplicación.

1.4.1 Metodologías de desarrollo de software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. [9]

Dentro de las metodologías más importantes podemos citar a: Proceso Unificado de Desarrollo ó RUP por sus siglas en inglés, XP y MSF.

1.4.2 Proceso unificado de desarrollo (RUP)

Un proceso de desarrollo del software es un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo el proceso unificado es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software. RUP define las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. El proceso de desarrollo se divide en cuatro fases.

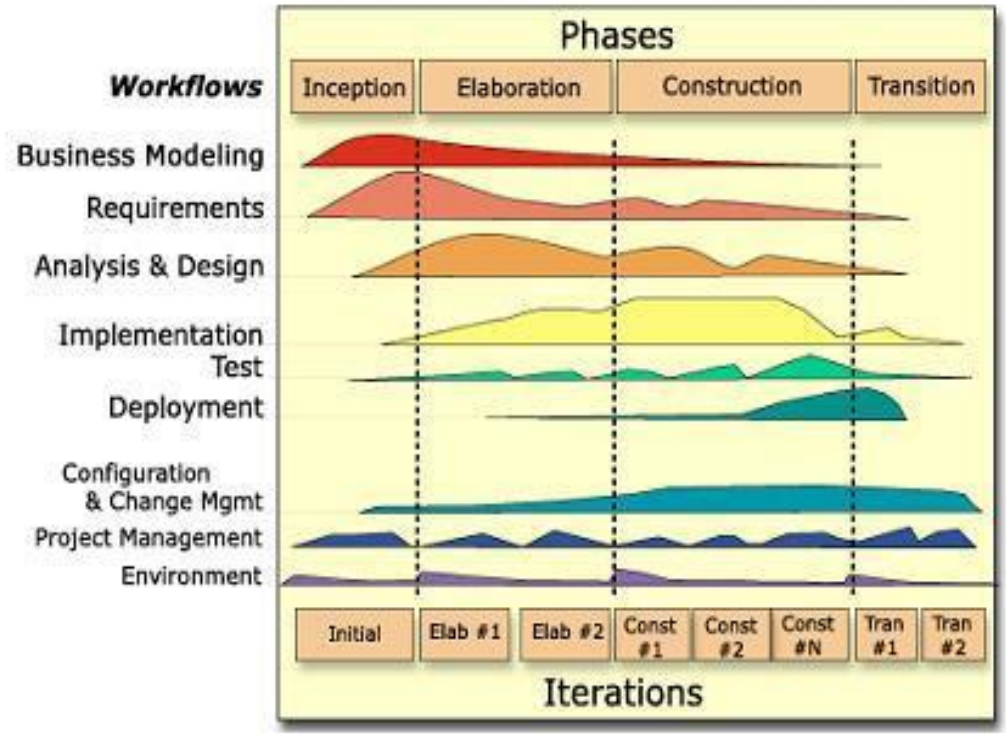

Figura 1 : Fases y flujos de trabajo

Flujos de trabajo:

- **Modelamiento del negocio:** Describe los procesos de negocio, mediante la identificación de participantes y actividades que requieren automatización.
- **Requerimientos***:* Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos).
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.).
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente***:* Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Define las siguientes fases:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto con la descripción de sus alcances con la identificación de los casos de uso del sistema.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release del producto que han pasado las pruebas.
- **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

El ciclo de vida de RUP se caracteriza por:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo.
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo, por lo que describe los elementos del modelo más importantes para su construcción. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.
- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, unos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. [10]

1.4.3 Java 2 Edición Empresarial

La plataforma J2EE de Sun es una de las dos grandes alternativas para construir aplicaciones de empresa y e-commerce que existen actualmente, siendo la otra la plataforma .NET de Microsoft. Las llamadas aplicaciones de empresa son aplicaciones del lado del servidor, aplicaciones grandes y complejas que proporcionan servicios a través de internet o redes privadas. Factores que contribuyen a esa complejidad son la diversidad de necesidades de información, puesto que cada usuario requiere distintas informaciones en distintos formatos, la complejidad de los procesos de negocio, y la diversidad de aplicaciones que se utilizan en cada empresa.

Lo primero que hay que decir de la plataforma J2EE es que es una especificación. Proporciona un modelo de programación que consiste en un conjunto de APIs y que dirige la manera de construir aplicaciones J2EE. Por otra parte J2EE especifica cómo debe ser la infraestructura de aplicación sobre la cual puedan correr las aplicaciones J2EE. Esta infraestructura de aplicación es proporcionada por los contenedores de las implementaciones de J2EE.

Un contenedor es el software que en tiempo de ejecución proporciona a las aplicaciones J2EE acceso al conjunto de APIs de J2EE y que gestiona los componentes de aplicación desarrollados según las especificaciones de las APIs. Esta gestión de los componentes por parte de los contenedores es, junto con la independencia de plataforma, una de las ventajas más destacables de J2EE.

La idea principal de J2EE es la de proveer un estándar simple y unificado para aplicaciones distribuidas a través de un modelo de aplicaciones basado en componentes. Para la construcción de un servicio web bajo la arquitectura J2EE, deberemos utilizar Java como lenguaje de programación. Y el contenedor que suministrará toda la infraestructura necesaria para convertir nuestra aplicación en un servicio web, también estará escrito en Java. [11]

1.4.3.1 Ventajas de J2EE sobre otras plataformas

Independencia de plataforma: Las aplicaciones J2EE son portables. Si cambiamos el sistema operativo de nuestros servidores y/o el programa servidor de aplicaciones, podremos seguir utilizando sin modificaciones las aplicaciones J2EE desarrolladas.

Objetos gestionados por los contenedores: El hecho de que los contenedores gestionen los objetos permite al desarrollador abstraerse de cantidad de aspectos que requieren tiempo y dedicación. Como consecuencia los desarrollos son más rápidos, con menos coste, y se obtiene un código mucho menos propenso a fallos y de más fácil mantenimiento. Además las aplicaciones J2EE son declarativas, lo que quiere decir que se puede modificar su comportamiento sin modificar el código, lo cual contribuye también a un mantenimiento mucho más rápido, sencillo, y de menor coste.

Reusabilidad: Al desarrollar el prototipo ALPA, se utilizaron componentes que habían sido desarrollados anteriormente para otras aplicaciones J2EE. Las aplicaciones J2EE están constituidas por la unión de componentes, como si se tratase de las piezas de un rompecabezas. Así se consiguen desarrollos más rápidos y de menor coste, y se proporciona mayor robustez a las aplicaciones.

Modularidad: La modularidad facilita enormemente el mantenimiento de las aplicaciones, ya que si se desean realizar modificaciones en algún módulo, éstas no deberían afectar al resto. Vistas estas ventajas, se puede entender mejor lo que queríamos decir con eso de que la idea principal tras J2EEes la de proveer un estándar simple y unificado para aplicaciones distribuidas a través de un modelo de aplicaciones basado en componentes. [11]

1.4.3.2 APIs de J2EE

JDBC 2.0

Enterprise Java Beans (EJB) 2.0

Java Servlets 2.3

Java Server Pages (JSP) 1.2
Java Message Service (JMS) 1.0
Java Transaction API (JTA) 1.0
Java Mail 1.2
Java Beans Activation Framework (JAF) 1.0
Java API for XML Parsing (JAXP) 1.1
The Java Connector Architecture (JCA) 1.0
Java Authentication and Authorization Service (JASS) 1.0 [11]

1.4.4 Java como lenguaje de programación

Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

- **Simple:** Java es un lenguaje de programación "simple" que puede ser programado sin extenso entrenamiento, mientras siga siendo adaptado a prácticas actuales de software. Los conceptos fundamentales de la Tecnología de Java son comprendidos rápidamente: los programadores pueden ser productivos desde el principio.
- **Orientado a Objetos:** la programación en Java está diseñada para ser orientada a objetos desde la raíz.
- **Multihilo:** La capacidad multihilo de la tecnología Java proporciona el significado de construir aplicaciones con muchos hilos (procesos) concurrentes de actividad. Multithreading da como resultado un alto grado de interactividad para el usuario final.
- **Dinámico:** El lenguaje y el sistema "run-time" son dinámicos en sus etapas de conexión. Las clases son conectadas solamente si son necesarias. Nuevos módulos de código pueden ser conectados en una demanda desde una variedad de fuentes, incluso de fuentes a lo largo de la red.
- **Arquitectura Neutral:** Porque Java ha sido diseñado para soportar aplicaciones que han sido desarrolladas en diferentes entornos de red heterogéneos. Estas aplicaciones tienen la capacidad de ejecutarse en una amplia variedad de arquitecturas de hardware. El compilador

de Java, genera bytecodes, una arquitectura neutral y formato intermediario diseñado para transportar código de una forma eficiente a múltiples plataformas de hardware y software.

- **Portable:** los programas desarrollados en Java son los mismos en cada plataforma no hay incompatibilidades de tipo de datos entre arquitecturas de hardware y software. La arquitectura neutral y la plataforma portable de la tecnología de Java es conocida como la Java Virtual Machine. Es la especificación de una máquina abstracta para la cual el compilador de Java puede generar código. La Máquina Virtual está basada principalmente en la interfaz POSIX—una definición estándar industrial de un sistema portable de interfaz.
- **Alto Rendimiento:** Rendimiento es siempre una consideración. La Plataforma Java logra un rendimiento superior adoptando un plan por el cual el interprete puede correr a toda velocidad sin necesidad de revisar el entorno “run-time”. El recolector automático de basura corre como un proceso o hilo de baja prioridad, asegurando una alta prioridad de que la memoria esté disponible cuando sea requerida.
- **Robusto:** La programación en lenguaje Java está diseñada para crear software fiable. Provee un extenso chequeo en el tiempo de compilación, seguido por un segundo nivel donde se verifica su correcto funcionamiento (run-time). El Modelo del Administrador de Memoria es extremadamente simple: los objetos son creados con un “new” (nuevo) operador.
- **Seguro:** La tecnología Java está diseñada para operar en entornos distribuidos, lo cual significa que la seguridad es de extrema importancia. Con características de seguridad diseñadas dentro del lenguaje y el “run-time system”, Java permite construir aplicaciones que no pueden ser invadidas desde fuera. En un entorno de red, las aplicaciones escritas en lenguaje de programación Java son seguras de intrusiones no autorizadas, virus o sistemas de invasión de archivos.[12]

La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Uno de los fundadores de Sun rescató la idea para utilizarla en el ámbito de Internet y convirtieron a Java en un lenguaje potente, seguro y universal gracias a que lo puede utilizar todo el mundo y es gratuito. Una de los primeros triunfos de Java fue que se integró en el navegador Netscape y permitía ejecutar programas dentro de una página web, hasta entonces impensable con el HTML.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas.

Para lo que nos interesa a nosotros, con Java podemos programar páginas web dinámicas, con accesos a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que deseemos hacer con acceso a través web se puede hacer utilizando Java.

1.4.5 Gestor de base de datos PostgreSQL

PostgreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que posean alrededor de 500.000 peticiones por día. Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo.

Características**:**

- **DBMS Objeto-Relacional**
  PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.
- **Altamente Extensible**
  PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.
- **Soporte SQL Comprensivo**
  PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins).
- **Integridad Referencial**
  PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- **Lenguajes Procedurales**
  PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.
- **Cliente/Servidor**
  PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL. [13]

Una de las características que más sorprende e impacta sobre PostgreSQL es el número de peticiones que se le pueden hacer a la base de datos sin corromper el sistema, esto es fundamental para gestionar ensayos clínicos desde todas las localidades del país donde se implante la solución informática. También soporta el gran volumen de información que se requiere recoger de todos los pacientes participantes en los ensayos.

1.4.6 Lenguaje de Modelamiento Unificado. UML: Unified Modeling Language

Es un lenguaje basado en gráficos que permite visualizar y documentar cada una de las partes que comprende el desarrollo del software que se quiere implementar. Muchas veces UML es referenciado como una herramienta ó como un método de desarrollo, sin embargo es un lenguaje, no indica que debe hacerse ó como debe hacerse, simplemente ayuda al entendimiento de lo que se debe hacer, significa un estándar para la descripción gráfica de todo el sistema ayudando al entendimiento para los desarrolladores.

1.4.7 Herramienta de Modelado Visual: Visual Paradigm para UML

Visual Paradigm para UML es una herramienta UML 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, además es multiplataforma y a su vez es software libre. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal.
En investigaciones anteriores la herramienta CASE de modelado utilizada, debido a sus características y facilidades de uso era Rational Rose, pero presentaba la limitante de ser un software propietario y la universidad no poseer licencia para utilizarlo. Por lo que se decidió emplear Visual Paradigm. [14]

1.4.8 Servidor web: Apache Jakarta Tomcat

El Jakarta Tomcat es un servidor de código abierto, es un contenedor de aplicaciones Web basadas en Java que fue creado para ejecutar Servlets y Java Server Page de aplicaciones web. Existe en el marco del subproyecto Apache-Jakarta, donde es apoyado y reforzado por un grupo de voluntarios de la comunidad de código abierto de Java.

El servidor Tomcat se ha convertido en la implementación de referencia para las especificaciones de Servlet y JSP. Es muy estable y tiene todas las características de un contenedor comercial de aplicaciones web. Tomcat también proporciona funcionalidades adicionales que hace que sea una gran opción para el desarrollo de una solución completa de aplicaciones Web. Algunas de estas características adicionales que ofrece Tomcat son: que es código abierto y libre, además incluye el Administrador de Aplicaciones Web de Tomcat, implementaciones especializadas en dominio, y válvulas de Tomcat.

1.4.9 Entorno de Desarrollo Integrado: Eclipse

Es uno de los más poderosos editores, para lenguajes como JAVA, C, PHP, y otros, y lo mejor de todo es que es OpenSource. Consta de una comunidad de millones de usuarios y miles de desarrolladores, cientos de plugins, soporte multi-lenguaje en una única herramienta, cuenta con coloreado de sintaxis PHP, autocompletado de código e inspección de métodos y atributos.
Esta versión de Eclipse dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant
- Asistentes (*wizards*): para creación de proyectos, clases, pruebas, etc.
- Refactorización

Asimismo, a través de "plugins" libremente disponibles es posible añadir:

- Control de versiones con Subversion.
- Integración con Hibernate.

1.4.10 Herramienta para el Control de Versiones: Subversion

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS (Concurrent Versions System). Este software es libre, se conoce como SVN.
Subversion posee las siguientes características:

- Versionado de directorios. Subversion implementa un sistema de versionado virtual, que almacena los cambios realizados tanto en ficheros como en directorios.
- Verdadero control de versiones. Subversion soporta operaciones adicionales sobre los ficheros y directorios almacenados guardando el historial de cambios, como por ejemplo añadir, borrar, copiar o renombrar (mover), tanto para ficheros como para directorios.
- Commits Atómicos. Subversion garantiza que todas las modificaciones sobre un repositorio se realizan completamente, o por el contrario no se realiza ninguna. Esto garantiza la eliminación de problemas de concurrencia, cuando por algún motivo solamente se envía una parte de los cambios al repositorio. Esta propiedad se debe a que Subversion almacena sus datos en una base de datos, que cumple las propiedades ACID (atomicidad, coherencia, aislamiento y permanencia).
- Control de versiones sobre metadatos. Cada fichero y directorio almacenado en el repositorio, puede tener asociado un conjunto de propiedades (clave: valor). Estas propiedades, son versionadas, al igual que el contenido de los ficheros almacenados.
- Diferentes modos de acceso. Subversion posee una abstracción en el acceso al repositorio, de forma que es fácil implementar un nuevo mecanismo de acceso a través de la red. Un ejemplo es añadir el plugin de Apache, de forma que podemos acceder al repositorio a través del protocolo HTTP. Automáticamente, Subversion hereda todas las funcionalidades que provee Apache para el manejo del repositorio, como son la autenticación o la compresión en el envío de datos. Aparte de esto, Subversion implementa un protocolo propio, que puede ser arrancado como un demonio independiente. Este protocolo se puede cifrar usando un túnel SSH para garantizar seguridad en la comunicación.
- Consistencia en los datos. Subversion maneja las diferencias entre los datos almacenados mediante un algoritmo de diferencias, que funciona de forma idéntica tanto para ficheros de texto, como para ficheros binarios (que almacenan código máquina). Ambos tipos de ficheros son almacenados idénticamente en el repositorio, y no es necesario especificar si los datos son binarios o no. [15]

Esta herramienta es de gran importancia para el equipo de desarrollo debido a la ventaja de la implementación concurrente acelerando así el trabajo y permitiendo un control completo de cada una de las versiones.

Después del análisis realizado teniendo en cuenta las especificidades sobre las que el sistema OpenClínica fue desarrollado se decidió adoptar las siguientes versiones de acuerdo a las herramientas expuestas anteriormente:

- PostgreSQL 8.2.4-1
- Visual Paradigm para UML (VP-UML) 6.4, correspondiente al paquete VP suite 3.4
- Apache Jakarta Tomcat 5.5.27
- Eclipse Platform 3.4.0
- Subversion 1.6
- Plataforma J2EE 1.5.0, que incluye la máquina virtual de java (JRE).

Conclusiones

Con el análisis realizado sobre las distintas herramientas, soluciones existentes, tecnologías y metodologías que fueron utilizadas durante el desarrollo de la solución, se puede entender mejor como fluye el proceso desde las primeras fases, y cuales implementos utilizar para lograr los objetivos de la investigación.

Capítulo 2: Características del Sistema

Introducción

En el presente capítulo se describe la propuesta de solución para la situación problémica. Para ello se detallan los procesos del Dominio mediante Diagramas de clases del Dominio con sus correspondientes descripciones.

Por otra parte, el capítulo brinda una concepción general del sistema propuesto, se exponen las funcionalidades y características que tendrán las modificaciones de las funciones administrativas y de reporte de traza en el sistema OpenClínica, se identifican los actores que interactúan con las mismas y se presentan los Diagramas de Casos de Uso del Sistema con las correspondientes descripciones de los casos de uso.

2.1 Modelo del Dominio

Todo sistema conlleva un grado de complejidad, por lo que para llegar a comprenderlo, es necesario modelarlo. En el caso de esta investigación se decidió utilizar modelo de Dominio debido a que los procesos de modificación de funcionalidades administrativas y de reporte de trazas del sistema OpenClínica no están bien identificados en el negocio definido por el CIM, por lo que solo se brindan una serie de representaciones visuales de los conceptos u objetos del mundo real significativos para esta área de interés.

2.1.1 Diagrama de Clases del Dominio

El modelo de dominio se representa en UML con un Diagrama de Clases en los que se muestra:

- Conceptos u objetos del dominio del problema: clases conceptuales
- Asociaciones entre las clases conceptuales
- Atributos de la clase conceptuales
- En el Modelo de Dominio no se muestra comportamiento.

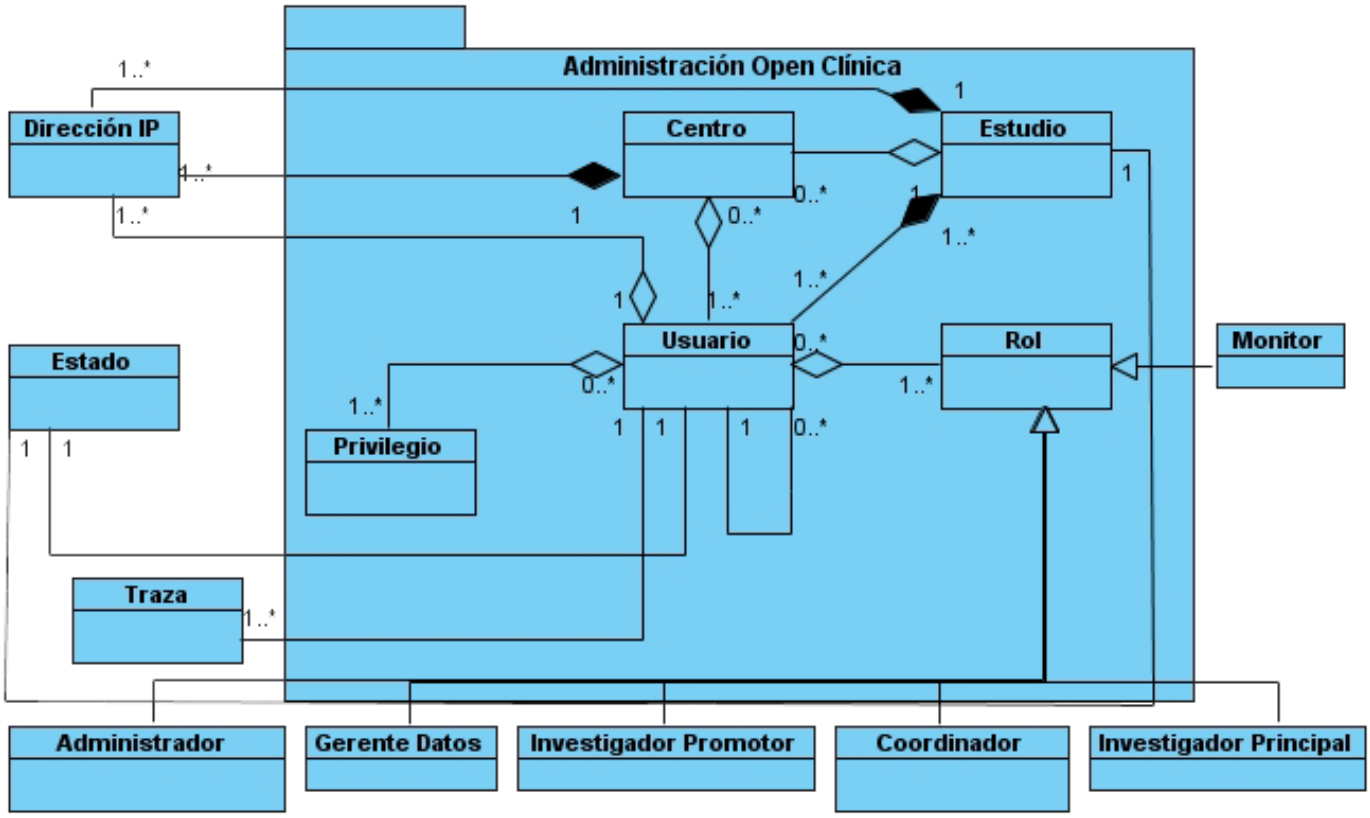

Figura 2 Modelo de Dominio

2.1.2 Descripción de Clases del Modelo de Dominio

Clase Centro

Descripción de la clase Centro

La clase Centro recoge información relacionada a los sitios u hospitales como:

Nombre del centro, ciudad, provincia, país, código postal, teléfono de contacto y correo electrónico.

Clase Estudio

Descripción de la clase Estudio

La clase Estudio recoge información como: nombre del estudio, fecha de creación, investigador principal, descripción, estado del estudio y otras que pueden ser especificadas en el momento que se crea el mismo.

Clase Usuario

Descripción de la clase Usuario

La clase Usuario recoge información relacionada con los usuarios que se crean para tener acceso a la aplicación, esta información puede ser:
Nombres y apellidos, correo electrónico teléfono, afiliación institucional, fecha de creación y estudio al cual está asociado.

Clase Privilegio

Descripción de la clase Privilegio

En la clase Privilegio se especifican los permisos que tendrán los usuarios de acuerdo a sus roles.

Clase Rol

Descripción de la clase Rol

La clase Rol contiene los tipos de usuarios que tienen acceso a la aplicación los cuales pueden ser:

1. Gerente de Datos
2. Administrador Técnico
3. Investigador Promotor
4. Coordinador del sitio
5. Investigador Principal
6. Monitor
7. Auditor

Clase Dirección IP

Descripción de la clase Dirección IP

La clase Dirección IP contiene los IPs o los rangos de IPs que desde los cuales un usuario puede acceder a un centro o sitio además de un identificador y una descripción.

Clase Traza

Descripción de la clase Traza

La clase Traza contiene o recoge información sobre las posibles acciones que se pueden realizar sobre los pacientes, momentos de seguimientos, CRD, variables y centros, esta información recogida puede ser:

1. Fecha de la acción realizada.
2. Usuario que realiza la acción.
3. Tabla a la que se le realiza la acción.

Clase Estado

Descripción de la clase Estado

La clase Estado contiene un conjunto de estados por los que pueden pasar los usuarios los centros o los estudios, como son:

1. Diseño
2. Ejecución
3. Completado
4. Disponible
5. No Disponible
6. Otros

2.2 Requisitos Funcionales

Muestran las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener, los cuales en la fase de construcción deben ser posibles de probar o verificar.

- **Gestionar IP Estudio y Centro**

1 Asignar número de IP y/o rango de IPs a un estudio.
2 Mostrar IPs y/o rango de IPs asignados a un estudio.
3 Eliminar un IP y/o rango de IPs asignado a un estudio.
4 Asignar número de IP y/o rango de IPs a un centro.
5 Mostrar IPs y/o rango de IPs asignados a un centro.
6 Eliminar un IP y/o rango de IPs asignado a un centro.

- **Gestionar IP Usuario**

7 Asignar número de IP y/o rango de IPs a un usuario.
8 Mostrar IP y/o rango de IPs asignados a un usuario.
9 Eliminar un IP y/o rango de IPs asignado a un usuario.

El Administrador establecerá los IP y/o rangos de IP de cada centro y del Centro Rector. Además podrá restringir el rango de IPs o IPs asignados a los roles Coordinador e Investigador Principal.

❖ **Modificar Privilegios Usuario**

10 Establecer permisos por rol para el acceso al sistema.

11 Validar acceso al sistema en dependencia de la dirección IP y el rol del usuario.

Cada usuario del sistema tendrá una cuenta de usuario, una contraseña y uno rol dentro de un estudio o centro.

Solo habrá un usuario con tipo de usuario:" Administrador" y los demás serán de tipo de usuario: "Usuario", es decir solo podrá haber un usuario con permisos de administración independientemente del rol que estos posean.

El usuario con permisos de administración no necesariamente tendrá un rol en el sistema.

Se definen los siguientes roles: Investigador Promotor, Gerente de Datos, Investigador Principal, Coordinador del sitio, Monitor.

El acceso por roles será de la siguiente forma:

- Desde cualquier centro se podrá acceder sólo al centro en cuestión y sólo a los módulos Gestionar datos y Extraer datos con los roles: Coordinador, Investigador Principal y Monitor.
- Un centro tendrá asociado un conjunto de IP. Para la entrada del Investigador Principal y del Coordinador, a un centro, deberá verificarse que los mismos se encuentren en uno de los IP asociados al centro.
- Un usuario en el Centro Rector del estudio podrá tener uno o varios roles.
- Un usuario con el rol Monitor o Investigador Promotor, podrá acceder a todos los centros desde el Centro Rector del estudio.

Cada usuario tendrá acceso a funcionalidades de los módulos en dependencia del rol que tenga.

La validación del acceso de los usuarios al sistema en dependencia de la dirección IP de la máquina cliente desde la cual intenta conectarse a la aplicación se realiza con el objetivo siguiente: si el usuario que intenta registrarse en el sistema tiene como rol Investigador Principal ó Coordinador de la Investigación Clínica, tendrá que acceder desde una de las PC predefinidas en el centro a que pertenece el usuario en cuestión, sino será redireccionado a una página de error. Para esta gestión se toma en cuenta la dirección IP del cliente. Es decir que la dirección IP del usuario tiene que estar corresponderse a la dirección o rango de direcciones del centro o estudio al cual intenta conectarse si y solo si el rol del usuario es alguno de los anteriormente mencionados.

❖ **Insertar Usuario**

12 Modificar la asignación de tipo de usuario.

13 Modificar los roles existentes en OpenClínica.

❖ **Insertar Estudio**

14 Cuando el estudio es creado este debe pasar a estado de “Diseño”.

❖ **Gestionar Trazas**

15 Mostrar trazas por variable.

16 Mostrar trazas generales en dependencia de las acciones de los usuarios sobre las entidades del negocio: Pacientes, momentos de seguimientos y CRDs.

17 Imprimir reporte de las trazas.

18 Mostrar trazas por variables desde los CRDs solo para los monitores.

El sistema permitirá realizar un reporte de las trazas de auditoría. Estará asociado fundamentalmente a las variables de los modelos y a las acciones en general que realiza determinado usuario sobre los elementos antes mencionados. Las acciones de las variables se pueden visualizar de dos formas distintas:

- Desde un módulo que se dedique a la gestión de las trazas.
- Desde los cuadernos de recogía de datos, permitiendo al usuario a través de un vinculo sobre alguna variable específica la visualización de las trazas de dicha variable.

19 Validar Gestión IP

Toda la gestión de la información relacionada con las direcciones IP del sistema lleva consigo su respectiva validación, garantizando que no existan incongruencias en cuanto a información errónea ni repetida.

2.3 Requisitos no funcionales del sistema

1- RNF de apariencia o interfaz externa

- Las páginas no tendrán muchas imágenes y poseerán pocos colores.

- Las páginas principales tendrán información que servirá de guía al usuario.
- Cada rol tendrá una interfaz diferente con las funciones que le corresponden.
- Se hará uso de simbología mediante íconos para indicar el estado de los elementos utilizados en el diseño. Además, los íconos contendrán funcionalidades específicas.

2- RNF de usabilidad

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

3- RNF de Soporte

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

4- RNF Seguridad

- El acceso a cualquier manipulación del sistema, tanto entrada como análisis de datos debe estar sometido a un proceso de autentificación del usuario donde será especificado el usuario y contraseña.
- Las contraseñas deberán tener más de 7 caracteres de longitud y tener una fortaleza media.
- Los usuarios estarán obligados a cambiar la contraseña cada 60 días como máximo.

- Cada usuario tendrá asignado uno o varios roles en el sistema. Cada rol definido tendrá niveles de acceso al Software.
- Solo podrán acceder a la aplicación, clientes a través de direcciones IP específicas bien controladas.
- Todo cambio o modificación en el sistema debe ser atribuible a un usuario particular según su autenticación.

5- RNF de Software

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

6- RNF de Hardware

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

7- RNF de Restricciones en el Diseño y la Implementación

- El análisis y diseño de la aplicación será basado en la Metodología RUP con el uso del lenguaje de modelado UML.
- Se usará como herramienta CASE Visual Paradigm para el modelado de los artefactos que se generan en cada uno de los flujos de trabajo definidos por RUP.
- Se usará como lenguaje de programación Java.
- Se usará como Gestor de Base de Datos PostgreSQL.
- Podrán ser utilizados varios estándares como HTTP, HTML, XML, SOAP, UDDI.

8- RNF de Extensibilidad

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

9- RNF de Disponibilidad

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

2.4 Patrones de Casos de Uso

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. Dado un contexto y un problema a resolver, estas técnicas han mostrado ser la solución adoptada en la comunidad del desarrollo de software. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática. En la presente investigación la utilización de estos patrones ha permitido que la modelación de los caso de uso definidos se acerque más a la realidad con lo que se logra que los requisitos elaborados obtengan un mayor grado de precisión. De todas estas técnicas o patrones se utilizaron aquellas que permitieron lograr mejores resultados como son:

2.4.1 Concordancia (Commonality)

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

- ***Reuso***

Consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones. De manera que deben existir al menos dos de ellos.

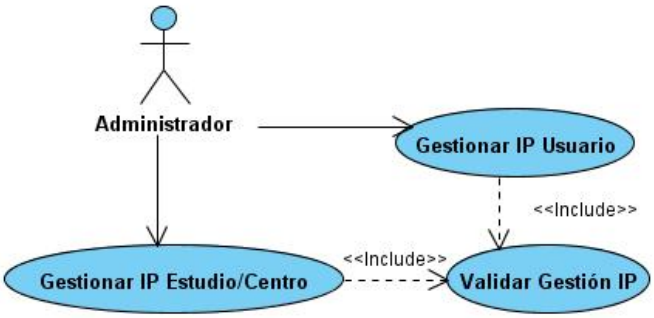

Figura 3 Reuso

2.4.2 CRUD (Creating, Reading, Updating, Deleting)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

- ***Completo***

Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

Figura 4 CRUD Completo

2.4.3 Múltiples actores

- ***Roles comunes***

Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso.

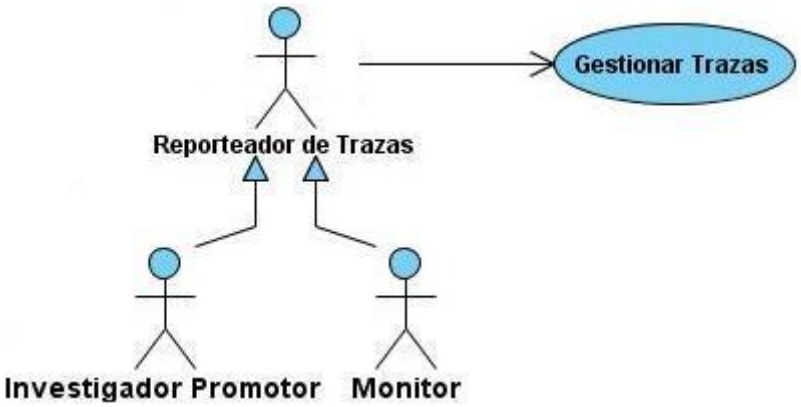

Figura 5 Roles Comunes

2.5 Actores del Sistema

Actor	Descripción
Administrador	El administrador es el encargado de realizar varias funciones disponibles en el módulo Administrar Empresa: • Insertar Usuario • Gestionar IPs y/o rango IPs de centros • Gestionar IPs y/o rango IPs de estudios • Gestionar IPs y/o rango IPs de usuarios
Gerente de datos	El gerente de datos tiene acceso al módulo Gestionar Estudio en el cual es el encargado de Insertar los estudios.
Reporteador de Trazas	Este actor es una generalización de: • Investigador promotor • Monitor Es el encargado de gestionar las trazas del sistema en dependencia de sus necesidades, ya sea mediante las acciones del usuario o por alguna variable específica. Además puede una vez obtenido el reporte imprimir la información resultante.
Investigador Promotor	Constituyen una especialización del actor del sistema Reporteador de Trazas y por tanto lleva a cabo las mismas funcionalidades que este. El monitor además puede visualizar las trazas de las variables pertenecientes a los CRDs e
Monitor	

	imprimirlas, desde vínculos a dichas variables dentro de los cuadernos.

2.6 Diagrama de Actores del Sistema

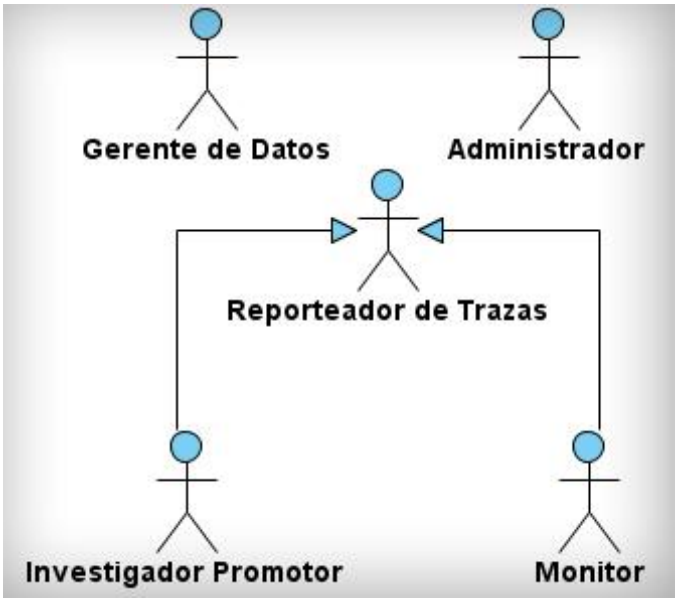

Figura 6 Diagrama de Actores del Sistema

2.7 Diagrama de Casos de Uso del Sistema

Los casos de uso que se presentan en el siguiente diagrama dan solución a los requerimientos funcionales del sistema:

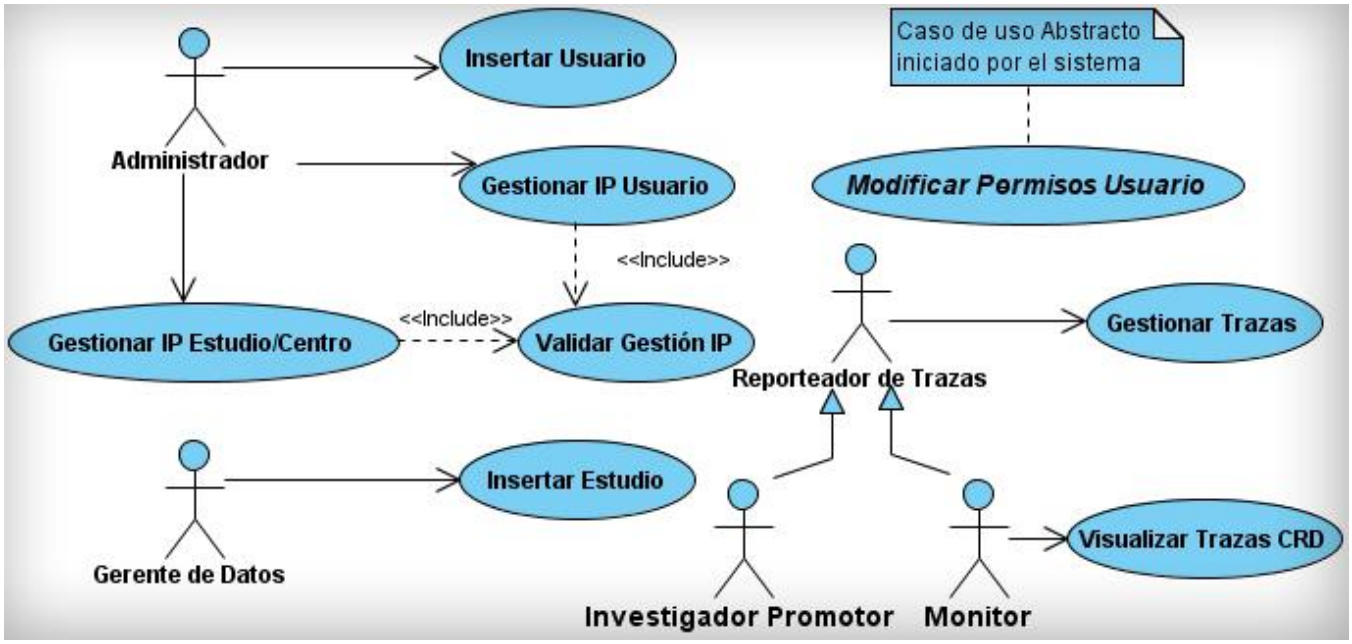

Figura 7 Diagrama Casos de Uso del Sistema

2.8 Descripción de los Casos de Uso del Sistema

2.8.1 Caso de uso Gestionar Trazas

Descripción de Casos de Uso Gestionar Trazas

Caso de Uso:	Gestionar Trazas
Actores:	Reporteador de Trazas (Inicia)
Resumen:	El caso de uso inicia cuando el Reporteador de Trazas decide gestionar las trazas de los usuarios sobre determinadas entidades del sistema. Dentro del módulo Extraer Datos el usuario selecciona la opción Gestionar Trazas, el sistema brinda la posibilidad de realizar la búsqueda pos dos variantes: uno por acciones generales sobre dentro de la aplicación y otra sobre variables pertenecientes a los cuadernos. Después de realizada la búsqueda es posible imprimir el resultado. El caso de uso finaliza cuando el Reporteador de Trazas indica la opción Salir.
Precondiciones:	El usuario registrado debe ser Monitor o Investigador Promotor.

	(Especializaciones del actor del sistema Reporteador de Trazas).
Referencias	RF 15,RF 16, RF 17
Prioridad	5

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Reporteador de Trazas selecciona la opción Gestionar Trazas del módulo Extraer Datos.	1.1 El sistema muestra las variantes por las cuales el usuario puede realizar la búsqueda y brinda la posibilidad de imprimir los resultados: • Criterios Generales. Ir a la sección Criterios Generales. • Variables de CRD. Ir a la sección Variables de CRD. • Imprimir. Ir a la sección Imprimir.
Sección “Criterios Generales”	
Acción del Actor	Respuesta del Sistema
1. El usuario escoge los criterios de búsqueda y ejecuta el filtro.	1.1 El sistema valida los datos en caso de que el usuario introduzca una fecha. Los demás campos son opcionales. 1.2 Realiza la búsqueda y muestra el resultado al usuario en la misma página.
2. El usuario en dependencia de sus necesidades puede: • Imprimir los resultados. Ir a la sección Imprimir. • Realizar nueva búsqueda por otros criterios. Ir al paso 1.1 del flujo normal de eventos. • Escoger la opción Salir. Terminar el caso de uso.	

Flujo alterno de eventos acción 1.1	
Acción del Actor	Respuesta del Sistema
	1.1a Si el usuario introduce un campo fecha con formato incorrecto el sistema muestra un mensaje indicando el error.
1.1b El usuario corrige los datos. Ir al paso 1 del flujo normal de eventos Sección “Criterios Generales”.	
Sección “Variables de CRD”	
Acción del Actor	Respuesta del Sistema
1. El usuario escoge los criterios de búsqueda y ejecuta el filtro.	1.1 El sistema valida los datos necesarios para los campos que son de obligada selección y en caso de que el usuario introduzca alguna fecha. 1.2 El sistema realiza la búsqueda y muestra los resultados en la misma página.
2. El usuario en dependencia de sus necesidades puede: • Imprimir los resultados. Ir a la sección Imprimir. • Realizar nueva búsqueda por otros criterios. Ir al paso 1.1 del flujo normal de eventos. • Escoger la opción Salir. Terminar el caso de uso.	
Flujo alterno de eventos acción 1.1	
Acción del Actor	Respuesta del Sistema
	1.1a Si el usuario no realiza las selecciones obligatorias o introduce una fecha con formato incorrecto el sistema muestra un mensaje indicando

	el error correspondiente.
1.1b El usuario corrige los datos. Ir al paso 1 del flujo normal de eventos Sección "Variables de CRD"	
Sección "Imprimir"	
Acción del Actor	Respuesta del Sistema
1. El usuario indica la opción Imprimir.	1.1 El sistema muestra el resultado de la búsqueda y brinda la posibilidad de imprimir el reporte mostrando un mensaje: "Para imprimir pulse <CTRL+P>."
2. El usuario imprime el resultado.	
Flujos Alternos	
Flujo alterno de eventos acción 2	
Acción del Actor	Respuesta del Sistema
2.a El usuario cierra la ventana. En dependencia de sus necesidades puede escoger: • Opción Salir. El caso de uso termina. • Escoger una de las variantes. Ir al paso 1.1 del flujo normal de eventos	

Prototipo de Interfaz

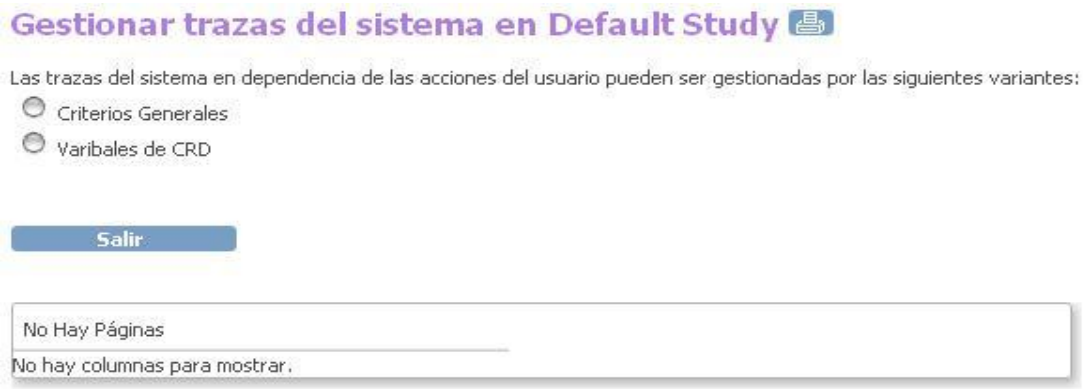

Figura 8 Prototipo de Interfaz Gestionar Trazas

Figura 9 Prototipo de Interfaz Gestionar Trazas por criterios generales

Figura 10 Prototipo de Interfaz Gestionar trazas por variables específicas

Poscondiciones	

2.8.2 Caso de uso Visualizar Trazas CRD

Descripción de Casos de Uso Visualizar Trazas CRD

Caso de Uso:	Visualizar Trazas CRD
Actores:	Monitor (Inicia)
Resumen:	El caso de uso inicia cuando el Monitor decide visualizar las trazas de variables pertenecientes a los cuadernos. En el módulo Enviar Datos selecciona la opción Monitorear Datos, en el listado de pacientes selecciona Ver y nuevamente selecciona la opción Ver correspondiente al nuevo listado que el sistema muestra, esta opción permite ver los datos referentes al cuaderno de datos asociado al momento de seguimiento que ha sido programado para el paciente. El Monitor selecciona el vínculo asociado a la variable y el sistema muestra las trazas de dicha variable, brindando la posibilidad de imprimir el reporte resultante. El caso de uso termina si el Monitor selecciona la opción Salir.
Precondiciones:	El usuario registrado debe ser un Monitor
Referencias	RF 18
Prioridad	5

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Monitor selecciona la opción Monitorear Datos del módulo Enviar Datos.	1.1 El sistema muestra un listado con los pacientes, los momentos de seguimientos existentes en el sistema y las acciones disponibles.
2. El Monitor selecciona la acción Ver correspondiente a un paciente.	2.1 El sistema muestra los datos del paciente, los momentos asignados al mismo y los cuadernos que pertenecen a dichos momentos. En todos los casos muestra el estado de cada una de las entidades mencionadas.

3. El Monitor selecciona la opción Ver correspondiente a un cuaderno especifico.	3.1 El sistema muestra los datos del cuaderno. Cada una de las variables que mostradas poseen un vinculo que permite visualizar las trazas de las mismas.
4. El Monitor selecciona una variable específica.	4.1 El sistema muestra una nueva ventana con las acciones que han realizado los usuarios sobre la variable seleccionada, brindando la posibilidad de imprimir el reporte presionando la combinación de teclas CTRL+P.
5. El Monitor presiona la combinación de teclas CTRL+P imprimiendo así el resultado del reporte.	
6. El Monitor cierra la ventana y escoge otra variable del sistema para visualizar sus trazas. Ir al paso 4.1 del flujo normal de eventos.	
Flujos Alternos	
Flujo alterno de eventos acción 5	
Acción del Actor	Respuesta del Sistema
5.a El Monitor después de imprimir el resultado del reporte, cierra la ventana y escoge la opción Salir terminando así el caso de uso.	

Prototipo de Interfaz

Figura 11 Prototipo de Interfaz Listado de momentos de seguimientos por pacientes

Figura 12 Prototipo de Interfaz Cuadernos asociados a momentos de seguimientos.

2.8.3 Caso de uso Gestionar IP Usuario

Descripción de Casos de Uso Gestionar IP Usuario

Caso de Uso:	Gestionar IP Usuario
Actores:	Administrador (Inicia)
Resumen:	El administrador selecciona la opción Usuarios dentro del módulo Administrar Empresa. El sistema muestra listado con los usuarios existentes, visualizando los IPs y rangos de IPs asignados. El administrador indica gestionar IP de algún usuario específico, el sistema muestra las opciones disponibles. El administrador confirma

	sus acciones y el caso de uso termina si el usuario selecciona otra opción del sistema.
Precondiciones:	El usuario debe estar registrado como Administrador.
Referencias	RF 7, RF 8, RF 9; CU Validar Gestión IP
Prioridad	4

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción Usuarios del módulo Administrar Empresa.	1.1 El sistema muestra un listado con los usuarios que existen en el sistema visualizando por cada rol los IPs y los rangos de IPs que tienen asignados, además de las acciones disponibles para cada uno de los roles. Solo estará disponible la opción gestionar IP para los roles a los cuales se les puede restringir los IPs y/o rangos de IPs: • Coordinador • Investigador principal
2. El administrador indica Gestionar IP de un usuario en específico.	2.1 El sistema muestra una interfaz con los IPs y rangos de IPs asignados al estudio o centro al cual dado un rol el usuario pertenece permitiendo al administrador realizar varias operaciones tanto con los IPs como con los rangos: • Adicionar • Eliminar
3. El administrador puede indicar cualquiera de las dos operaciones sin necesidad de enviar el formulario al seleccionar la opción Siguiente. Si el administrador escoge la opción: • Adicionar. Ir a sección Adicionar IPs y/o	

rango de IPs. • Eliminar: Ir a sección Eliminar IPs y/o rango de IPs.	
4. El administrador selecciona la opción Confirmar.	4.1 El sistema muestra una interfaz de confirmación de los datos que se gestionaron.
5. El administrador selecciona la opción Enviar.	5.1 El sistema actualiza la información relacionada con los IPs y/o rango de IPs del usuario en cuestión y redirecciona al listado de usuarios. Ir a paso 1.1 del flujo normal de eventos.
Sección “Adicionar IPs y/o rango de IPs.”	
Acción del Actor	Respuesta del Sistema
1. El administrador entra los datos ya sea IP y/o rango de IPs y selecciona la opción Adicionar.	1.1 El sistema realiza una validación común a la del caso de uso Gestionar IP Estudio/Centro, (Ver CU Validar Gestión IP) además el sistema valida que las direcciones que se vayan a asignar deben estar comprendidas dentro de las ya existentes para el centro o estudio en el cual dicho usuario juega un rol determinado. Si no existen errores adiciona los datos insertados al listado de IPs y/o rango de IPs respectivamente. Ir al paso 3 del flujo normal de eventos.
Sección “Eliminar IPs y/o rango de IPs.”	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona del listado de IPs y/o rango de IPs el valor que desea eliminar e indica realizar la operación con la opción Eliminar.	1.1 El sistema elimina por cada petición del administrador un IP y/o un rango de IPs. Ir al paso 3 del flujo normal de eventos.
Flujos Alternos	

Flujo alterno de eventos acción 2	
Acción del Actor	Respuesta del Sistema
2.a El caso de uso termina si el administrador escoge otra opción disponible en el sistema.	
Flujo alterno de eventos acción 4	
Acción del Actor	Respuesta del Sistema
4.a El administrador selecciona la opción Cancelar.	4.b El sistema redirecciona al listado de usuarios. Ir a paso 1.1 del flujo normal de eventos.
Flujo alterno de eventos acción 5	
Acción del Actor	Respuesta del Sistema
5.a El administrador escoge la opción Atrás.	5.b El sistema vuelve a la página anterior.
Flujo alterno para Sección Adicionar IPs y/o rango de IPs acción 1.1	
Acción del Actor	Respuesta del Sistema
	1.1a Si el sistema al validar los datos encuentra errores, muestra el mensaje de error en dependencia del mismo y no envía la información obligando al administrador a corregir los datos.
1.1b El administrador corrige los datos y procede a introducir los datos nuevamente. Ir al paso 1 del flujo normal de eventos de la sección “Adicionar IPs y/o rango de IPs”.	
Prototipo de Interfaz	

Figura 13 Prototipo de interfaz Administrar Usuarios

Figura 14 Prototipo de interfaz Gestionar IP Usuario

Figura 15 Prototipo de interfaz Pantalla de confirmación Gestionar IP Usuario

Poscondiciones	Se modifica la información referente a los IPs y/ rango de IPs de los usuarios, añadiendo o eliminando IP y/o rangos de IPs.

2.8.4 Caso de Uso Modificar Permisos Usuario

Descripción Caso de Uso Modificar Permisos Usuario

Caso de Uso:	Modificar Permisos de Usuario
Actores:	
Resumen:	El caso de uso es abstracto, se desencadena cuando el usuario entra al sistema, de forma que este verifica que usuario exactamente se registró y le asigna los permisos correspondientes según el rol que desempeña y al estudio o centro a que pertenece, además valida que el usuario en cuestión acceda al sistema desde una PC permitida.
Precondiciones:	El usuario debe estar registrado.
Referencias	RF 10, RF 11
Prioridad	4

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
	1. Obtiene el usuario que intenta acceder así como el rol que presenta en el estudio, asignándole los permisos a los módulos que le corresponden según su rol. **Módulo Gestionar Estudio:** Rol Gerente de Datos • Crear estudio • Gestionar hojas CRD • Gestionar Cronograma de Ejecución • Gestionar reglas • Gestionar grupos Rol Investigador Promotor • Firmar Cronograma de Ejecución del estudio • Cerrar estudio **Módulo Gestionar Datos:** Rol Investigador Principal (IP), Coordinador de la Investigación Clínica (CIC) y Monitor (M) • Añadir sujeto (IP y CIC) • Gestionar sujetos (IP y CIC) • Monitorear datos (M) • Cerrar centro (IP) • Notas y discrepancias (M, IP y C) **Módulo Extraer Datos:** Todos los roles • Extraer Conjunto de datos • Ver conjunto de datos • Mostar trazas 2. Valida que el usuario que intenta

	acceder en dependencia del rol, esté conectado desde una PC con acceso al sistema.
Poscondiciones	El usuario queda con los permisos que le corresponden en el sistema.

Conclusiones:

En este capítulo se muestra detalladamente como fluye la información actualmente en el dominio mediante sus artefactos correspondientes. Quedaron definidos los requerimientos funcionales y no funcionales que debe tener la aplicación y los casos de uso del sistema con sus descripciones correspondientes.

Capítulo 3: Diseño del Sistema

Introducción

Este capítulo contiene una valoración crítica del diseño a desarrollar y la construcción de los artefactos inherentes a este. Se hace énfasis en la arquitectura sobre la que se desarrolla y los patrones de diseño para agilizar y perfeccionar el diseño. Se describe el flujo principal de los diagramas y las principales clases que serán posteriormente implementadas así como un acercamiento al modelo físico de la base de datos que sustenta la aplicación.

3.1.1 Estilos arquitectónicos

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas.

3.1.2 Patrones de diseño

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones. Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software se denominan patrones de diseño. [16]

3.1.3 OpenClínica: Estilo arquitectónico Modelo-Vista-Controlador

OpenClínica se basa en un clásico estilo o patrón arquitectónico conocido como Modelo-Vista-Controlador o MVC para abreviar. La arquitectura MVC se utiliza para mapa tradicional de entrada, procesamiento y salida de las funciones del modelo de interacción gráfica de usuario. Se ha utilizado en muchas aplicaciones, desde su surgimiento, debido a su sencillo enfoque flexible a la creación de aplicaciones de varios niveles.

Para esto, utiliza las siguientes abstracciones dividiendo el código de forma tal que la estrategia seguida durante la implementación del software sea fácil de mantener y ampliar:

- **Componentes de la Vista:** colaboran encaminados a redireccionar al usuario a las páginas JSP (Java Server Pages por sus siglas en inglés) y HTML las cuales se encargan de mostrar el contenido al mismo, es decir al usuario que interactúa con la aplicación.
- **Componentes del modelo:** trabajan de forma tal que proveen una interfaz a la base de dato para interactuar con la misma y crear objetos java basados análogamente en los datos residentes en ella.
- **Componentes controladores:** regulan el tráfico entre la vista y el modelo, regulando el acceso seguro y proveyendo la lógica de negocio.

Adicionalmente, el modelo se auxilia de una capa de abstracción, esta capa permite abstraer la aplicación del gestor de base de datos que se instale para su despliegue es decir es independiente de esta, permitiendo el acceso a la información almacenada en bases de datos relacionales diferentes. Por lo tanto cambiar de gestor es sumamente fácil. Conformar esta capa es producto del trabajo en conjunto del paquete Apache Commons Digester (Paquete Digestor) y el patrón de diseño DAO por sus siglas en inglés Data Access Object (Objeto de Acceso a Datos).

Apache Commons Digester: es un paquete de código abierto que permite de forma fácil guardar e importar variables desde un archivo XML, usando el digestor se pueden almacenar todas las consultas SQL fuera de la aplicación eliminando así la necesidad de crear clases especificas para determinada base de datos. Este paquete será referenciado desde el IDE en el que se desarrolle la aplicación y desde el contenedor WEB que en caso específico de la investigación es Apache Tomcat, además debe estar acompañado de otras librerías: BeanUtils (commons-beanutils), Collections (commons-collections-2.1.1), Logging (commons-logging-api) y un parseador XML que puede ser SAX ó JAXP.

Data Access Object (DAO): DAO es un Patrón de Diseño enmarcado dentro de la arquitectura J2EE y es considerado una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula. Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente, la cual podría ser JDBC, JDO, EJB CMP, TopLink, Hibernate, o cualquier otra tecnología de persistencia. Usando Objetos de Acceso de Datos significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación. En general se usan los DAOs para

abstraer y encapsular todos los accesos a la fuente de datos, administra las conexiones con esta para almacenar y obtener datos.

El uso del patrón MVC con la capa de abstracción brinda a la aplicación las siguientes fortalezas:

- Fácil modificación y actualización por múltiples desarrolladores.
- Reutilización de componentes del modelo y del controlador evitando de esta forma la repetición de código y manteniendo el tamaño de la aplicación sin grandes crecimientos en cuanto a capacidad.
- Soporte de forma más sencilla para necesidades especificas de tecnologías.

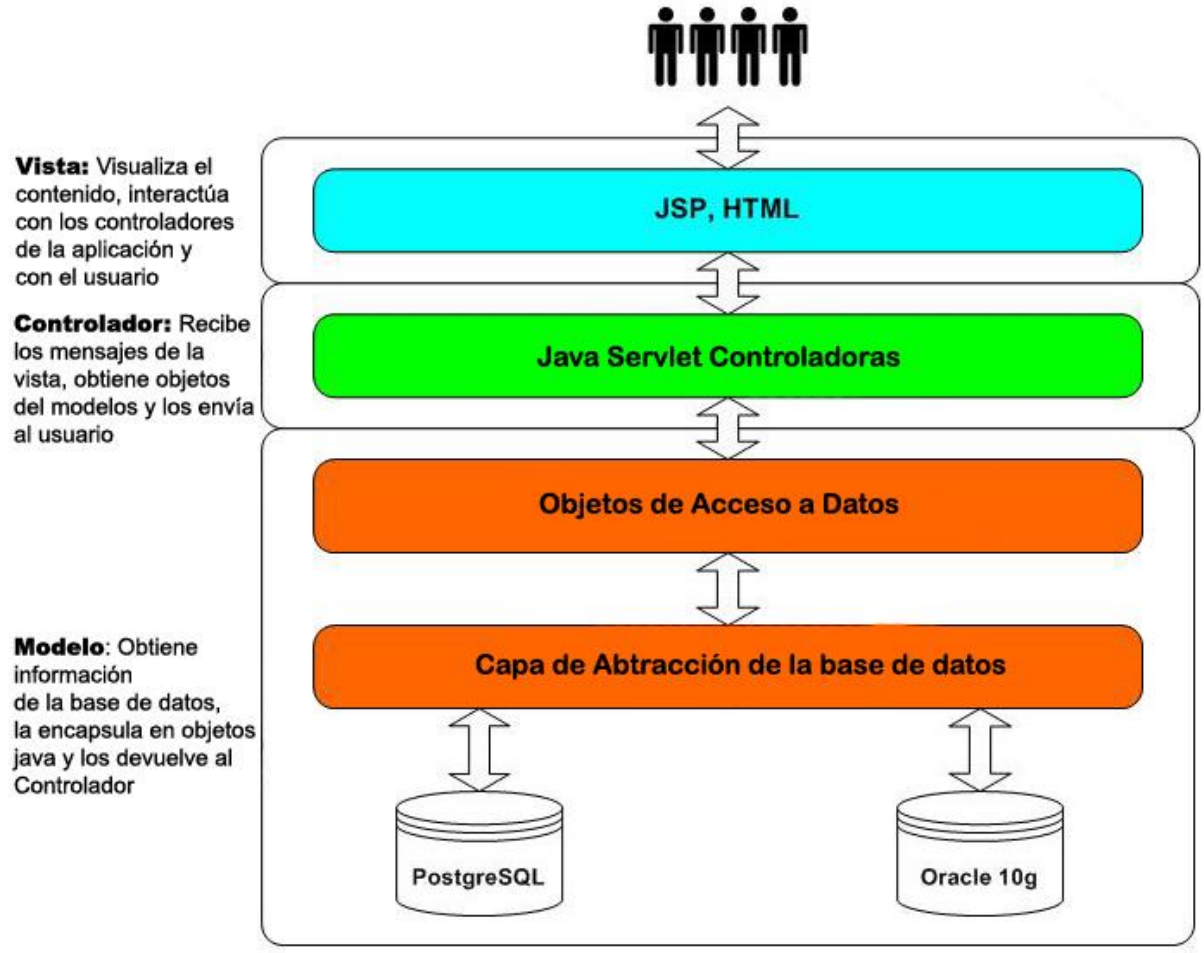

Figura 16 Arquitectura MVC

3.1.4 Patrones de diseños empleados

Como se mencionaba anteriormente el patrón DAO se utiliza para abstraer todo lo referente a la fuente de datos. Este patrón se auxilia de otros con los cuales está relacionado:

- **Data Transfer Object J2EE (Objeto de transferencia de datos)**

  Un DAO usa este patrón para transportar datos a y desde sus clientes. Pertenece al catalogo de patrones empleados por la arquitectura J2EE.

- **Factory Method and Abstract Factory**

  Factory Method (Método de fabricación) centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear. El patrón Abstract Factory (Fabrica abstracta) puede ser empleado para añadir flexibilidad a la estrategia a seguir con el Método de Fabricación ya que este permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. [17]

  Todo proyecto tiene numerosa factorías potenciales, para intercambiar fácilmente comportamientos:

  - decoradores y familias de componentes gráficos
  - Sistemas de log
  - Clases de acceso a gestor a bases de datos
  - Sistemas multi-lenguaje
  - Privilegios en base al rol de usuario.

  De las cuales se usan en esta investigación las relacionadas con:

  - Clases de acceso a gestor a bases de datos
  - Sistemas multi-lenguaje
  - Privilegios en base al rol de usuario.

  Como ejemplo de esto se encuentra la clase SQLFactory la cual es la encargada de abstraer el acceso a gestor a bases de datos.

```
public class SQLFactory {

    // DAO KEYS TO USE FOR RETRIEVING DIGESTER
    public final String DAO_USERACCOUNT = "useraccount";
    public final String DAO_STUDY = "study";
    public final String DAO_STUDYEVENTDEFNITION = "studyeventdefintion";
    public final String DAO_SUBJECT = "subject";
    public final String DAO_STUDYSUBJECT = "study_subject";
    public final String DAO_STUDYGROUP = "study_group";
    public final String DAO_STUDYGROUPCLASS = "study_group_class";
    public final String DAO_SUBJECTGROUPMAP = "subject_group_map";
    public final String DAO_STUDYEVENT = "study_event";
    public final String DAO_EVENTDEFINITIONCRF = "event_definition_crf";
    public final String DAO_AUDITEVENT = "audit_event";
    public final String DAO_AUDIT = "audit";
    public final String DAO_DATAVIEW = "dataview_dao";
    public final String DAO_ITEM = "item";
    public final String DAO_ITEMDATA = "item_data";
    public final String DAO_ITEMFORMMETADATA = "item_form_metadata";
    public final String DAO_CRF = "crf";
    public final String DAO_CRFVERSION = "crfversion";
    public final String DAO_DATASET = "dataset";
    public final String DAO_SECTION = "section";
    public final String DAO_MASKING = "masking";
    public final String DAO_FILTER = "filter";
```

Figura 17 Ejemplo de código de la clase SQLFactory.

Estos dos últimos patrones pertenecen a la familia GoF (Gang of Four: Pandilla de los Cuatro) de los cuales podemos citar además:

- **Builder** (Constructor virtual)

  Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto. Un ejemplo que evidencia este patrón es la creación de objetos de tipo tablas que se utiliza posteriormente para la visualización de los datos.

```
protected void processRequest() throws Exception {
    FormProcessor fp = new FormProcessor(request);

    UserAccountDAO udao = new UserAccountDAO(sm.getDataSource());
    EntityBeanTable table = fp.getEntityBeanTable();
    // table.setSortingIfNotExplicitlySet(1, false);

    ArrayList allUsers = getAllUsers(udao);
    setStudyNamesInStudyUserRoles(allUsers);
    ArrayList allUserRows = UserAccountRow.generateRowsFromBeans(allUsers);

    String[] columns =
        { resword.getString("user_name"), resword.getString("first_name"), resword.(
          resword.getString("IP"), resword.getString("Rango_IP"), resword.getString
    table.setColumns(new ArrayList(Arrays.asList(columns)));
    table.hideColumnLink(4);
    table.setQuery("ListUserAccounts", new HashMap());
    table.addLink(resword.getString("create_a_new_user"), "CreateUserAccount");

    table.setRows(allUserRows);
    table.computeDisplay();

    request.setAttribute("table", table);
```

Figura 18 Ejemplo de código del patrón Builder.

- **Iterator** (Iterador)

  Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos. En la implementación se 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.

```
public int findNextKey() {
    this.unsetTypeExpected();
    Integer keyInt = new Integer(0);
    this.setTypeExpected(1, TypeNames.INT);
    ArrayList alist = this.select(digester.getQuery("findNextKey"));
    Iterator it = alist.iterator();
    if (it.hasNext()) {
        HashMap key = (HashMap) it.next();
        keyInt = (Integer) key.get("key");
    }
    return keyInt.intValue();
}
```

Figura 19 Ejemplo de código del patrón Iterator.

Catalogo de patrones J2EE

- **Session Façade** (Fachada de Sesión)

  El uso de un bean de sesión como una fachada para encapsular la complejidad de las interacciones entre los objetos de negocio y participantes en un flujo de trabajo. El Session Façade maneja los objetos de negocio y proporciona un servicio de acceso uniforme a los clientes. Se evidencia en la creación de objetos bean y que se almacena en las sesiones permitiendo usar este objeto en distintas clases del flujo.

- **Composite View (Vista compuesta)**

  Un objeto vista que está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include o el action include es un patrón Composite View. Este patrón se evidencia por ejemplo en la página menú.jsp la cual incluye otras páginas JSP además de código HTML para construir la página final que muestra el contenido al usuario.[18]

Patrones GRASP

El uso del patrón de arquitectura MVC introduce el trabajo con otros patrones de diseño para asignar responsabilidades los cuales son:

- **Bajo acoplamiento**

  Surge ante la problemática de cómo dar soporte a una mínima dependencia a un aumento en la reutilización. El acoplamiento mide qué tan fuerte está una clase conectada con otras (es decir, cuántas clases conoce y necesita). Este patrón se evidencia en la implementación de las clases pues están estructuradas de tal forma que se puedan reutilizar y que no depende de muchas otras a pesar de depender de no pocas, a pesar de haber dependencias estas son mínimamente necesarias, es decir no depende de clases que no necesita de forma importante.

  Para determinar el nivel de acoplamiento de clases ver los diagramas de secuencia de los casos de uso.

- **Alta cohesión**

  Surge ante la problemática de cómo mantener la complejidad dentro de límites manejables. Este se evidencia desde el punto de vista que cada una de las clases implementadas tiene responsabilidades específicas y no están cargadas con demasiadas funcionalidades. Es decir, cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable**.** Por ejemplo existen clases destinadas a la

gestión de los formularios y sus validaciones, otras para la gestión de tablas y filas. La representación de este patrón está basada en las colaboraciones que existen entre clases mediante sus métodos.

Figura 20 Ejemplo de clases con funcionalidades específicas.

- **Patrón experto**

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema:

- Lógica de negocio
- Persistencia a la base de datos
- Interfaz de usuario

No tiene sentido considerar que una clase se debe escribir a sí misma en base de datos o formatearse para presentarse en una página HTML por el hecho de poseer los datos. Estos son elementos estructuralmente distintos y deben considerarse desde una perspectiva distinta.

Es la solución a ¿cuál es el principio fundamental en virtud de a quien se asignan las responsabilidades en el diseño orientado a objetos?

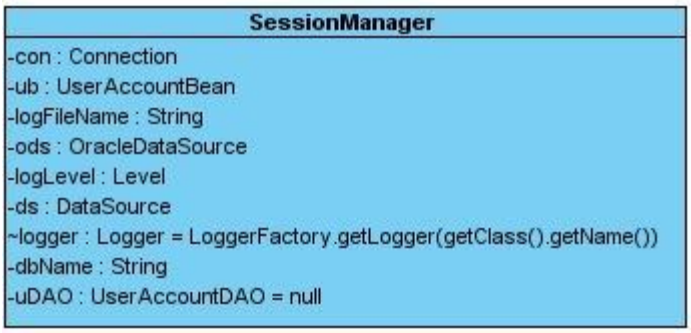

Figura 21 Clase experta en información

- **Creador**

Guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna

situación. Logrando así mayor mantenibilidad y reutilización. Ante la problemática ¿quién debería ser el responsable de la creación de una nueva instancia de alguna clase? En el patrón Creador es donde se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- B contiene a A.
- B es una agregación (o composición) de A.
- B almacena a A.
- B tiene los datos de inicialización de A (datos que requiere su constructor).
- B usa a A.

Ejemplo de uso de este patrón son las clases java servlets que se encargan de crear las instancias de otras clases como son los BEAN y los DAO.[19]

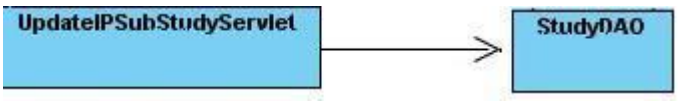

Figura 22 Ejemplo de creación de la clase DAO

❖ **Controlador**

Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

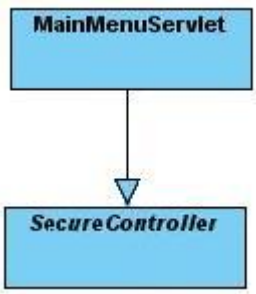

Figura 23 Ejemplo de controlador

3.2 Diagrama de clases del diseño para aplicaciones Web

Estos tipos de diagramas representa la disposición estática de las clases del diseño, la relación que existe entre ellas y sus colaboraciones. Cada clase posee atributos y métodos, con el objetivo de no extender los diagramas estos no visualizan ni atributos ni métodos, en el expediente de proyecto asociado a la presente investigación se podrán encontrar los archivos *.vpp que no son más que la

modelación UML de dichos diagramas. Se utilizó las extensiones que posee UML para aplicaciones Web usando los estereotipos <<Client Page>>, <<HTML Form>> y <<Server Page>>.

Explicación General de diagramas de clases del diseño

Los diagramas responden a la arquitectura MVC, por lo tanto se representan 3 paquetes principales correspondientes a la vista, el modelo y el controlador. La lógica es la siguiente:

El usuario interactúa con una página cliente que puede o no contener uno o varios formularios, esta interactúa con una página Servlet que se encarga de procesar la petición a través del método proccessRequest(), dentro de este método se pueden realizar varias operaciones en dependencia del pedido del usuario; de forma general en la mayoría de los casos se instancia uno o varios DAOs (Objetos de Acceso a Datos) que son los que se encargan de hacer las consultas a la base de datos usando internamente la librería *Digester*, estos se auxilian de los objetos *bean* como una analogía a las tablas existentes. Estos objetos bean conforman la capa de atracción. Una vez que la pagina Servlet procesa los datos, devuelve los mismos y redirecciona esta información a clases controladoras que se encargan de construir la vista, estas páginas se denominan JSP (Java Server Pages) y poseen código HTML embebido, compilan el código con la información que le son enviadas y finalmente construyen las paginas clientes que le son mostradas al usuario. Para todo el gestionamiento antes mencionado existen clases ya definidas de las cuales se auxilian las principales, estas clases se pueden empaquetar de acuerdo a sus funcionalidades. En el paquete Jsp mostrado en los diagramas se encuentran las páginas incluidas por los componentes de la vista para la construcción de las interfaces. El paquete Javascript contiene los archivos *.js necesarios para validaciones, calendarios etc. El paquete Properties contiene los archivos de internacionalización para lograr que la aplicación sea multilenguaje. El paquete Java se modela con el objetivo de representar en este todas aquellas clases que son nativas del lenguaje. En algunos diagramas se especifican archivos *.js con el objetivo de particularizar alguna acción propia del CU en cuestión.

Descripción de las principales clases

Clase	Atributos	Métodos	Descripción
SecureController	context sm logger logDir	init() forwardPage() processRequest() mayProceed()	Una de las clases más relevante por ser una generalización de todas las clases Servlet. Ya sea por post

	logLevel session request response ub currentStudy currentRole errors	doGet() doPost() process() forwardPage()	o por get; el método process() es invocado, procesa la petición y crea las variables de sesión más importantes. Por su parte mayProceed () y processRequest() son abstractos y son implementados en las clases hijas.
ManageTracesServlet	locale input_startdate input_enddate user momentdefinition subject crf var print	mayProceed() processRequest() BuildPageSelections()	Se encarga de gestionar todas las trazas del sistema en dependencia de las opciones seleccionadas por el usuario.
MainMenuServlet	locale	mayProceed() processRequest() getRoles() EsIgualIP() EsRangoValido() ip2long()	Esta es la primea página que se llama y en ella se restringe al acceso a los módulos en dependencia del rol del usuario y de la dirección IP desde donde accede en caso que sea necesario.
UpdateIPUserServlet		mayProceed() processRequest() confirmIPUser() submitIPUser() createStudyUserRolBean() getAdminServlet()	Es la encargada de preparar y actualizar las direcciones de IP desde las cuales un usuario puede acceder al sistema, las que tienen que estar en correlación con las direcciones IP de el estudio o centro a que pertenece el mismo.

3.2.1 Caso de Uso Gestionar Trazas

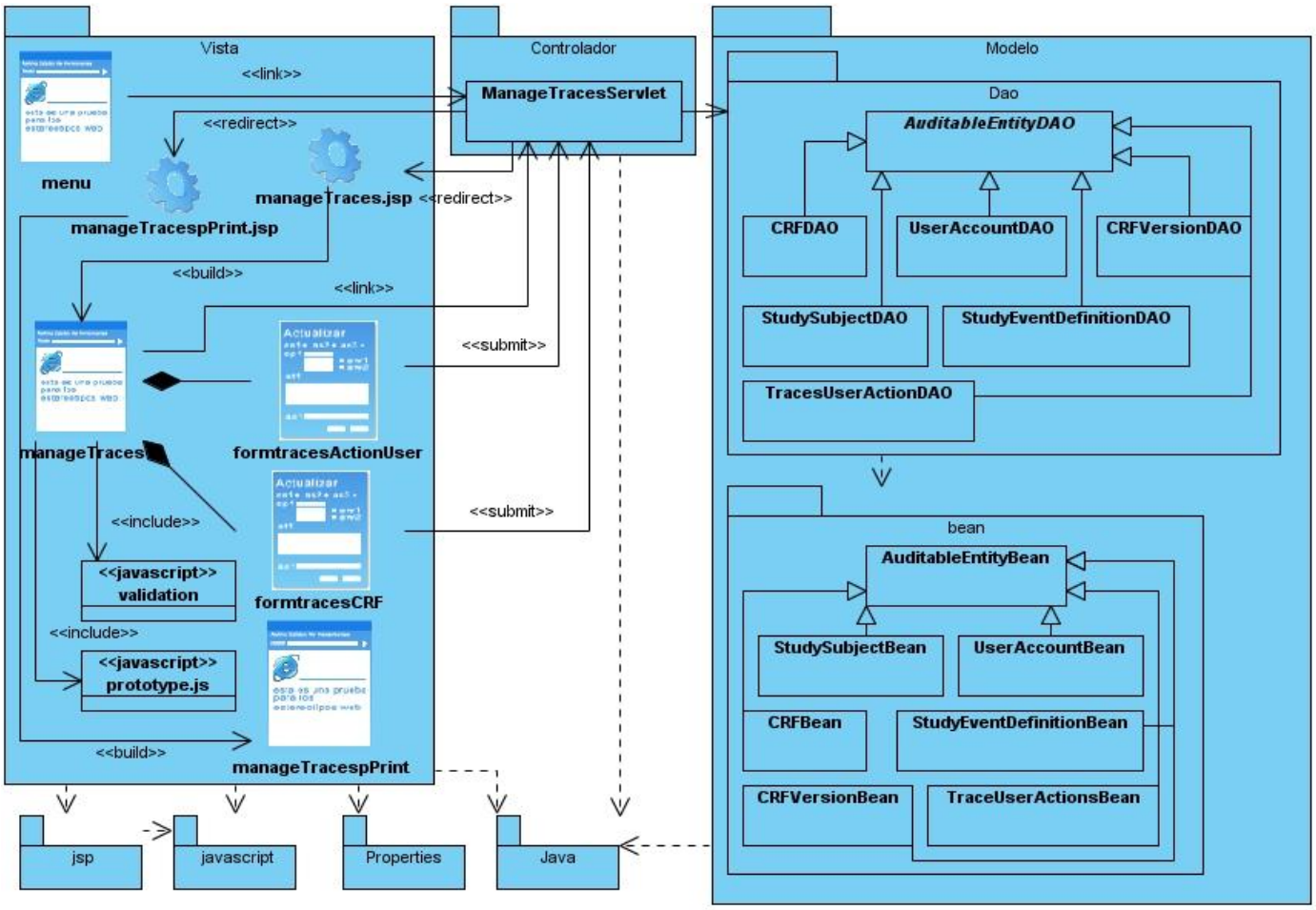

Figura 24 Diagrama de Clases del Diseño. Caso de Uso Gestionar Trazas.

3.2.2 Caso de Uso Visualizar Trazas CRD

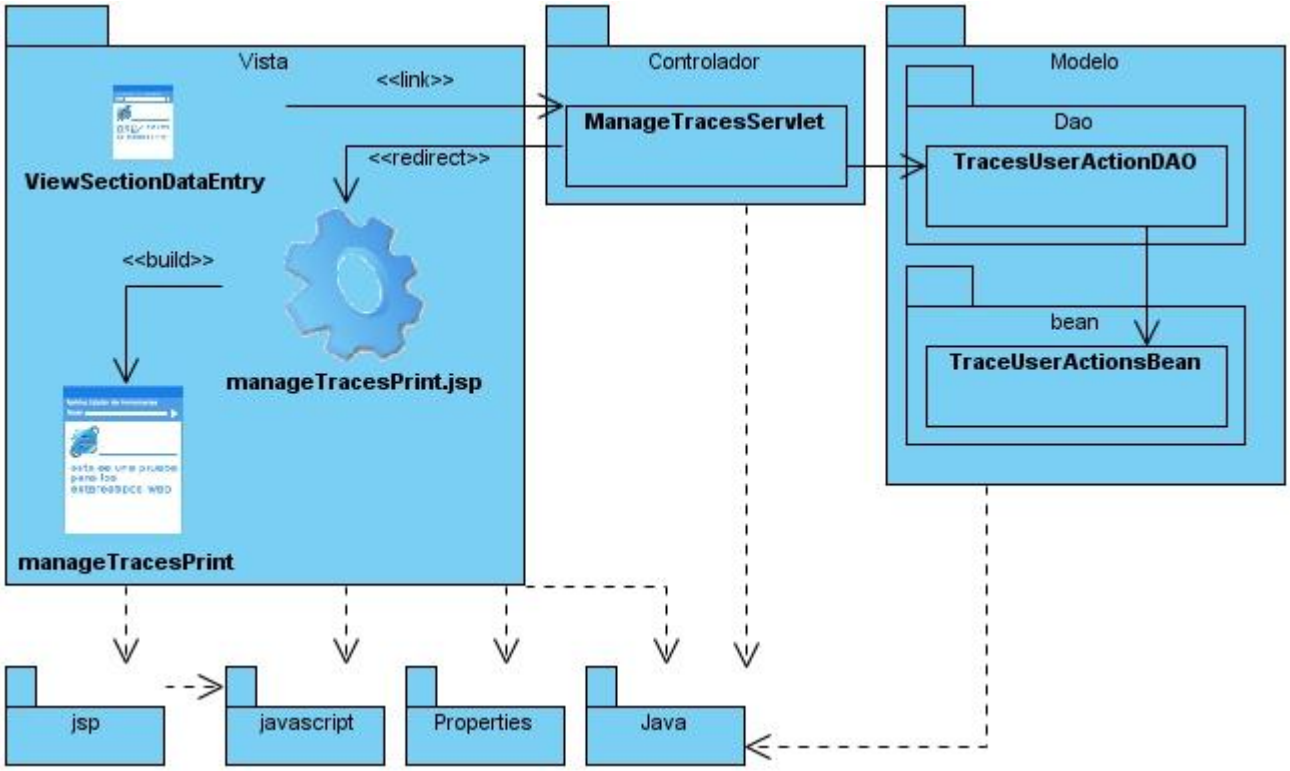

Figura 25 Diagrama de Clases del Diseño. Caso de uso Visualizar Trazas CRD.

3.2.3 Caso de Uso Gestionar IP Usuario

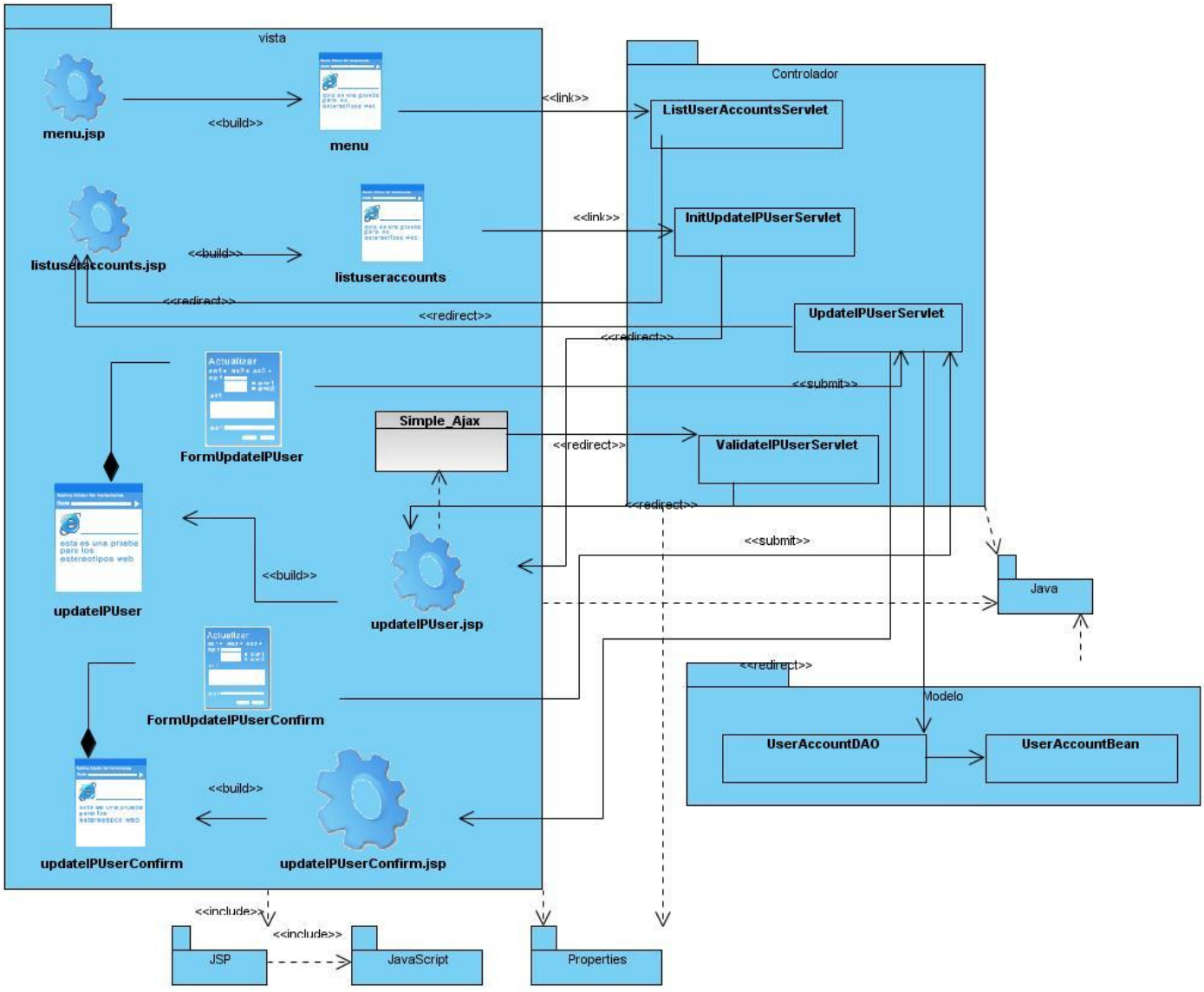

Figura 26 Diagrama de clases del diseño. Caso de uso Gestionar IP Usuario

3.2.4 Caso de Uso Modificar Permisos Usuario

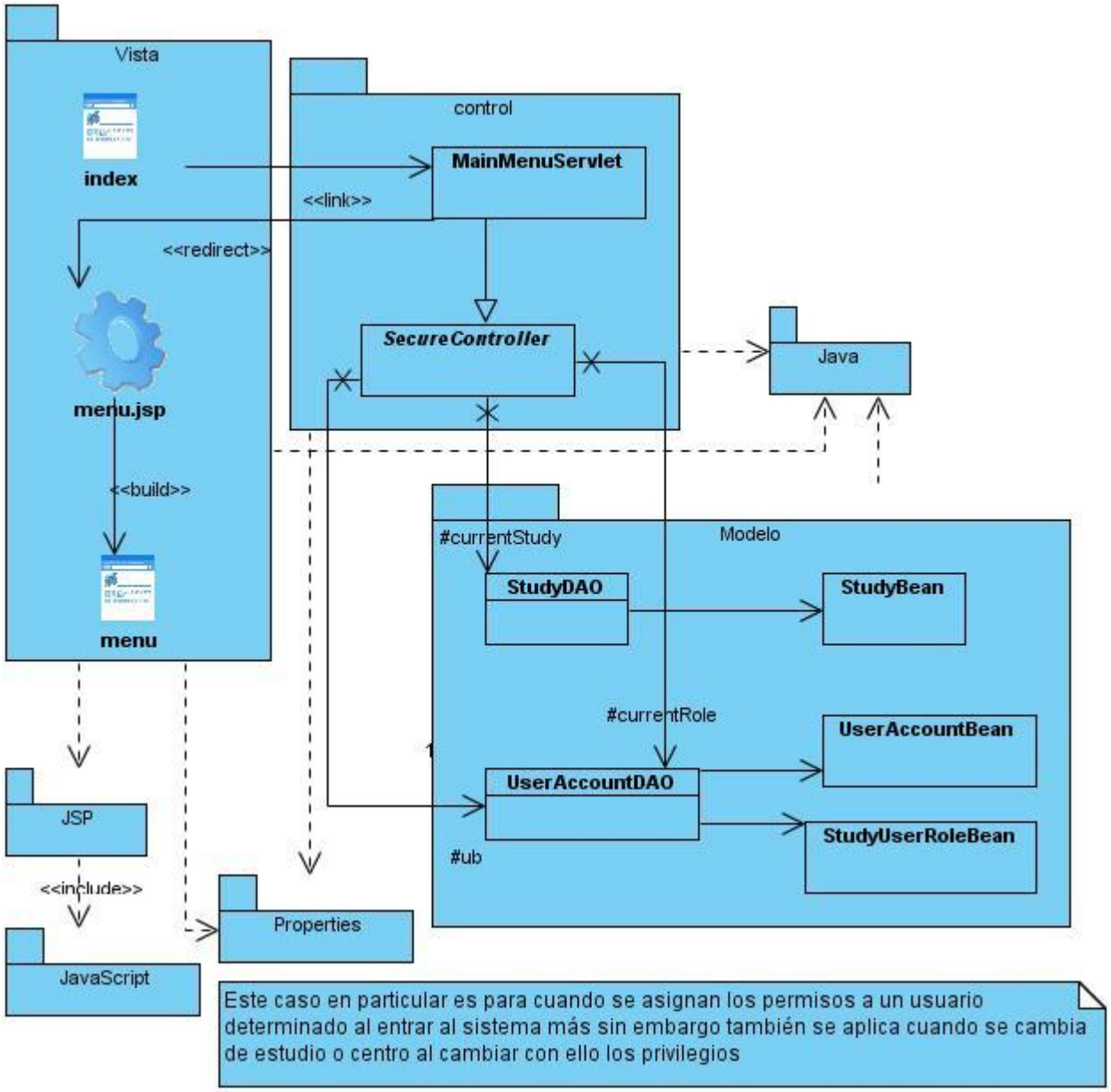

Figura 27 Diagrama de clases del diseño. Caso de Uso Modificar Permisos Usuario

3.3 Diagramas de interacción (secuencia) para web.

Los diagramas de secuencias son herramientas UML que permiten describir gráficamente el orden temporal de las interacciones entre distintos entes relacionados con el desarrollo de un sistema

software. A continuación se muestran los diagramas correspondientes a la investigación siendo un diagrama de secuencia por cada escenario del caso de uso que se pretende modelar, puede darse el caso que se modele un diagrama para alguna funcionalidad en específico con el objetivo de hacer más sencilla la interpretación de los mismos. La lógica de los diagrama es la misma que la descrita anteriormente en los diagramas de clases del diseño con la diferencia que en este caso se muestran secuencialmente los pasos para desarrollar el caso de uso y más específicamente el escenario en cuestión. Los mensajes los constituyen las peticiones de los usuarios y los métodos que son invocados por las clases. Como se sigue el mismo principio, las peticiones van desde los usuarios, interactúa con los elementos de la vista, las clases del paquete controlador manejan las peticiones, recuperan la información de la base de datos y la envían a una clase que construye la interfaz para finalmente mostrar los datos el usuario.

3.3.1 Caso de uso Gestionar Trazas. Escenario "Crear Interfaz"

Este escenario se modela con el objetivo de hacer más simples los diagramas pertenecientes a este caso de uso. La explicación es simple, una vez que el usuario decide gestionar las trazas del sistema, la aplicación crea la interfaz para que este decida por cual de los dos criterios realizar la búsqueda y además permitirle la opción Imprimir. El diagrama a continuación modela los pasos secuenciales para la creación antes mencionada. Cuando la página cliente *menu* envía el mensaje a la ManageTracesServlet se ejecuta el método processRequest() el cual a su vez invoca otro llamado BuildPageSelections() que es quien se encarga de preparar los datos para ser insertados dinámicamente en la interfaz. Este flujo de pasos ocurre cada vez que la información es recuperada desde la base de datos y procesada para construir la interfaz.

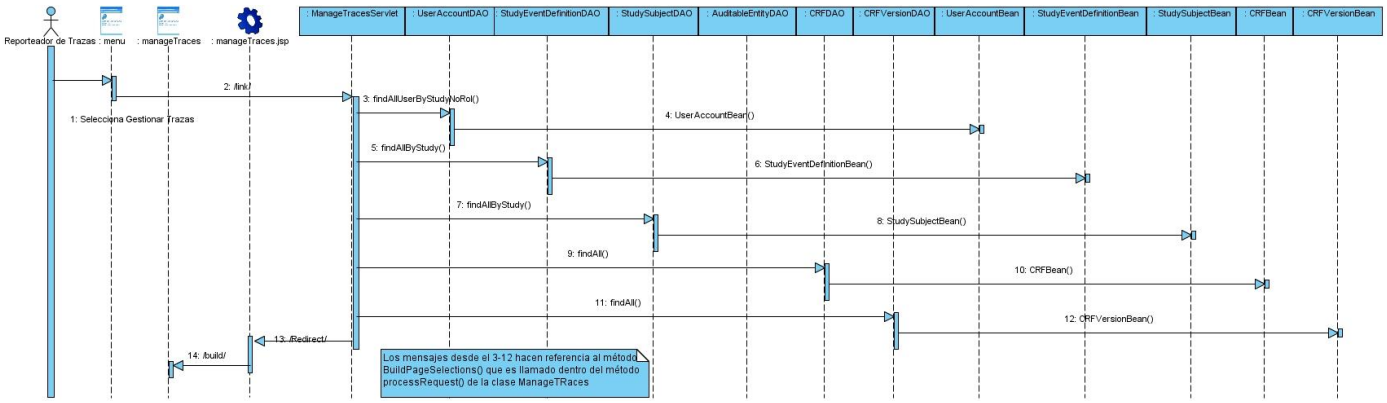

Figura 28 Caso de uso Gestionar Trazas. Escenario "Crear Interfaz"

3.3.2 Caso de uso Gestionar Trazas. Escenario “Criterios Generales”

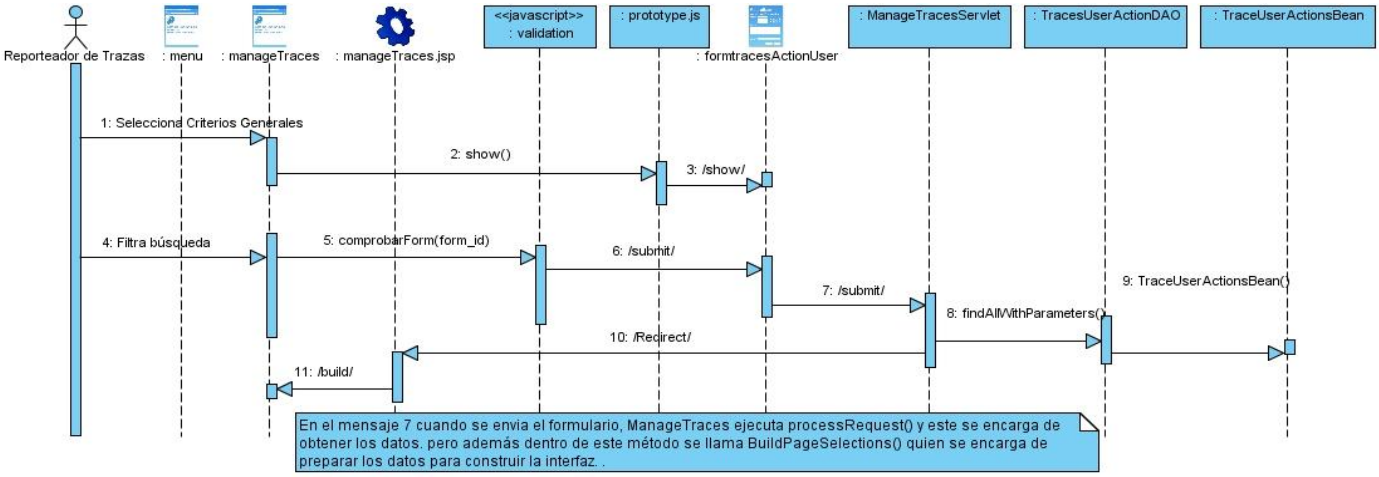

Figura 29 Diagrama de secuencia. Escenario “Criterios Generales

3.3.3 Escenario “Variables de CRD”

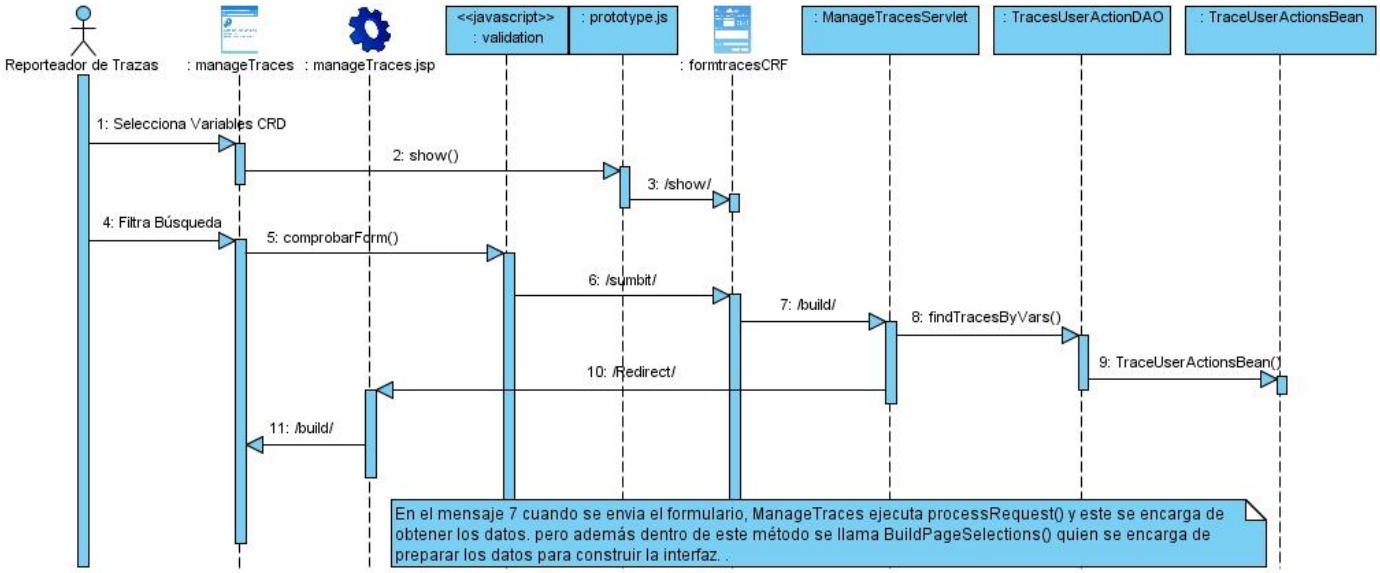

Figura 30 Diagrama de secuencia. Escenario “Variables de CRD”.

3.3.4 Escenario “Imprimir”

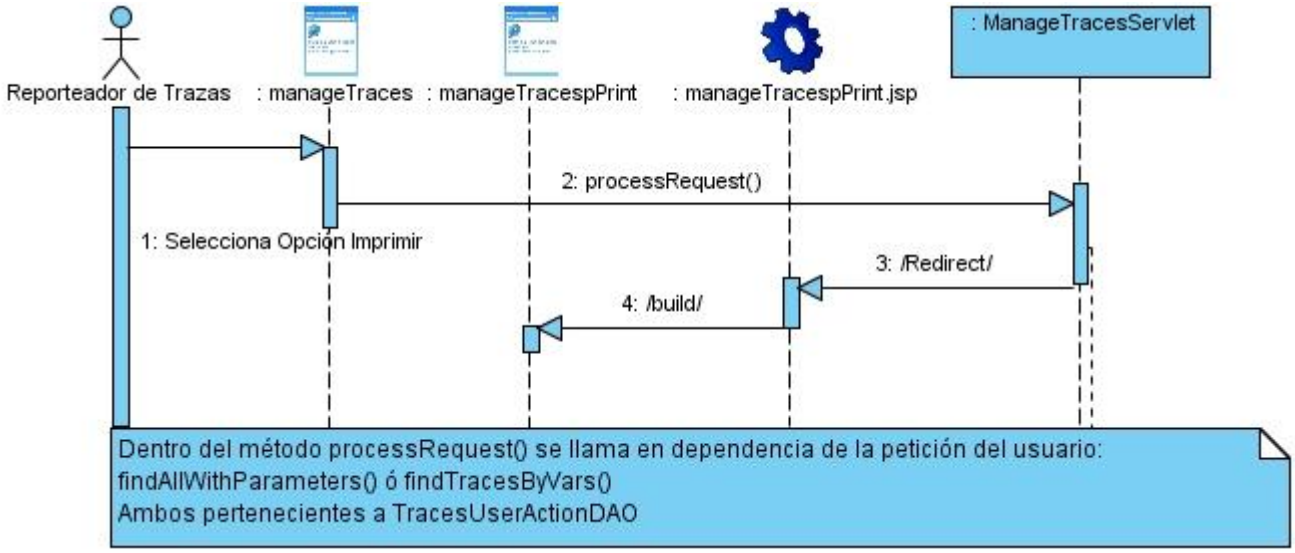

Figura 31 Diagrama de secuencia. Escenario “Imprimir”.

3.3.5 Caso de Uso Visualizar Trazas CRD

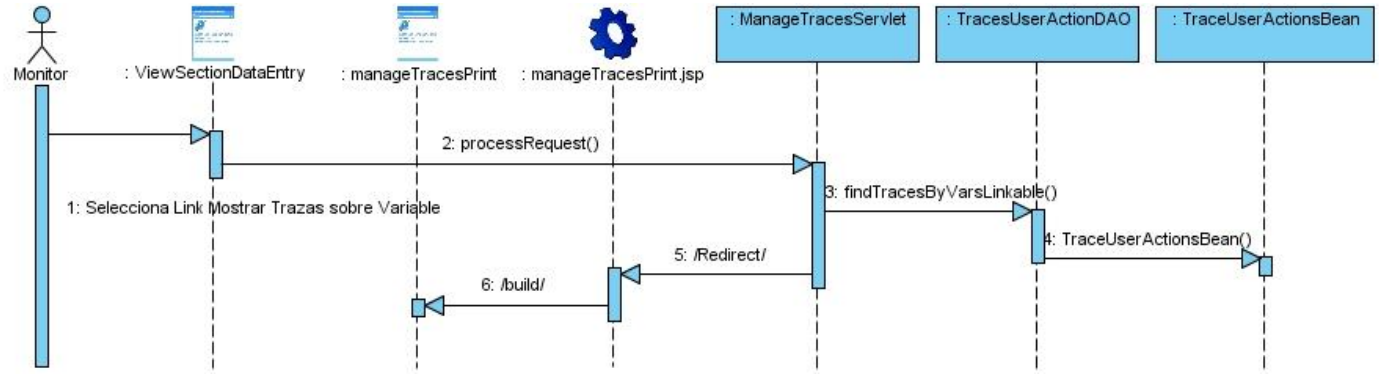

Figura 32 Diagrama de secuencia. Caso de uso Visualizar Trazas CRD

3.3.6 Caso de Uso Modificar Permisos Usuario

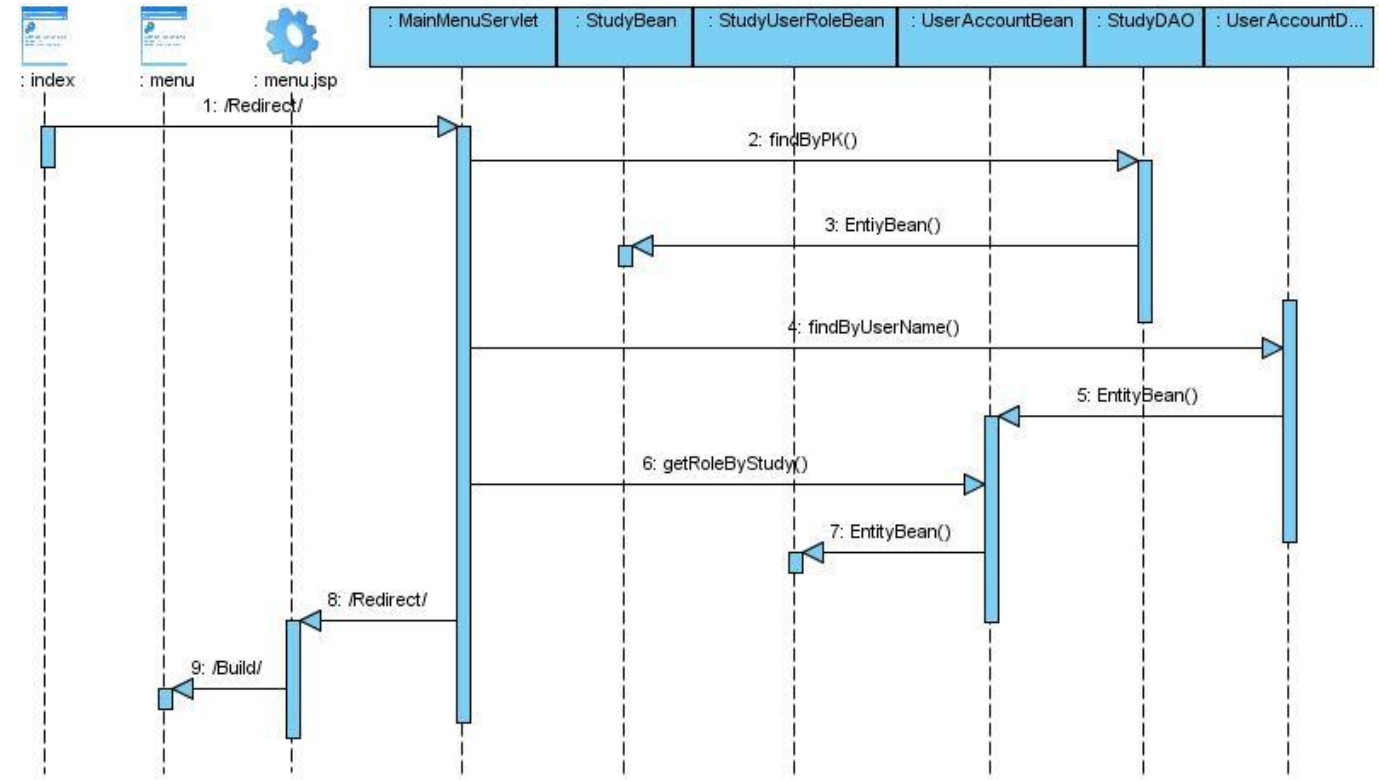

Figura 33 Diagrama de secuencia. Caso de Uso Modificar Permisos Usuario.

3.3.7 Caso de Uso Gestionar IP Usuario. Escenario “Adicionar”

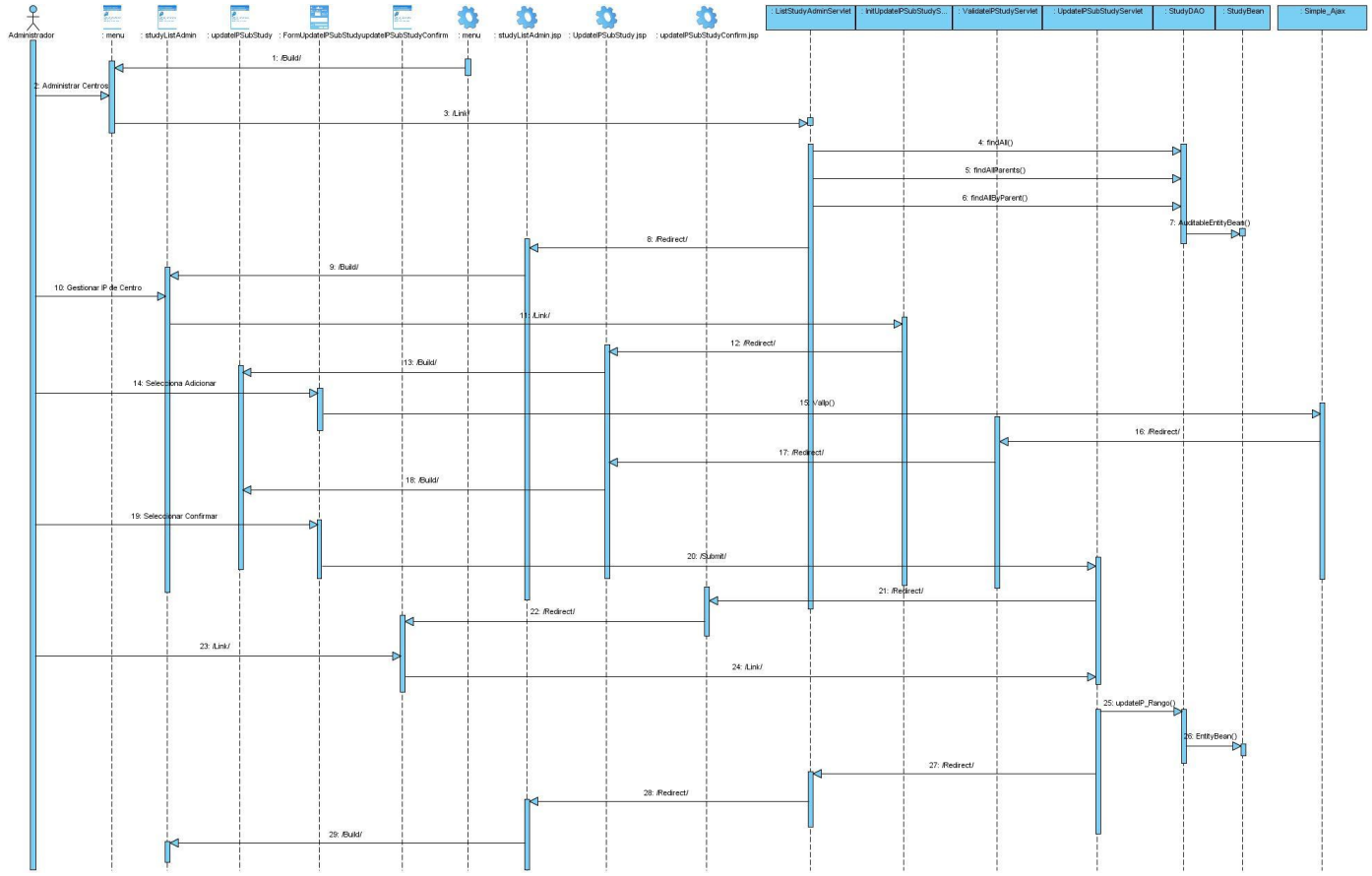

Figura 34 Diagrama de secuencia. Caso de Uso Gestionar IP Centro. Escenario. “Adicionar”.

3.3.8 Caso de Uso Gestionar IP Usuario. Escenario “Eliminar”

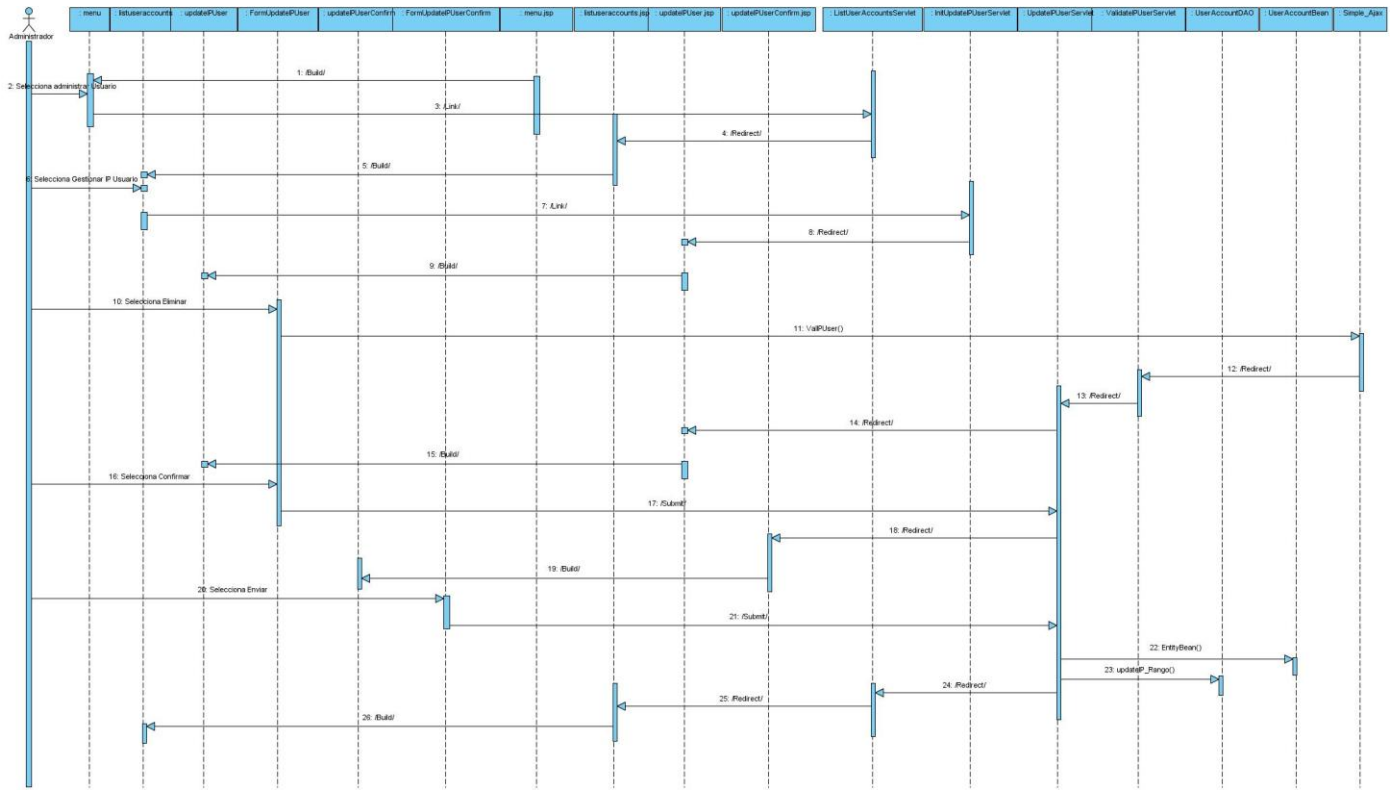

Figura 35 Diagrama de secuencia. Caso de Uso Gestionar IP Centro. Escenario. “Eliminar”.

3.4 Modelo de Datos

A continuación se muestra la interacción entre las clases persistentes de la aplicación mediante el modelo de datos. Este modelo de datos corresponde a las funcionalidades nuevas que se desarrollaron y las que fueron modificadas, es decir este modelo solo refleja las clases sobre las que se trabajó.

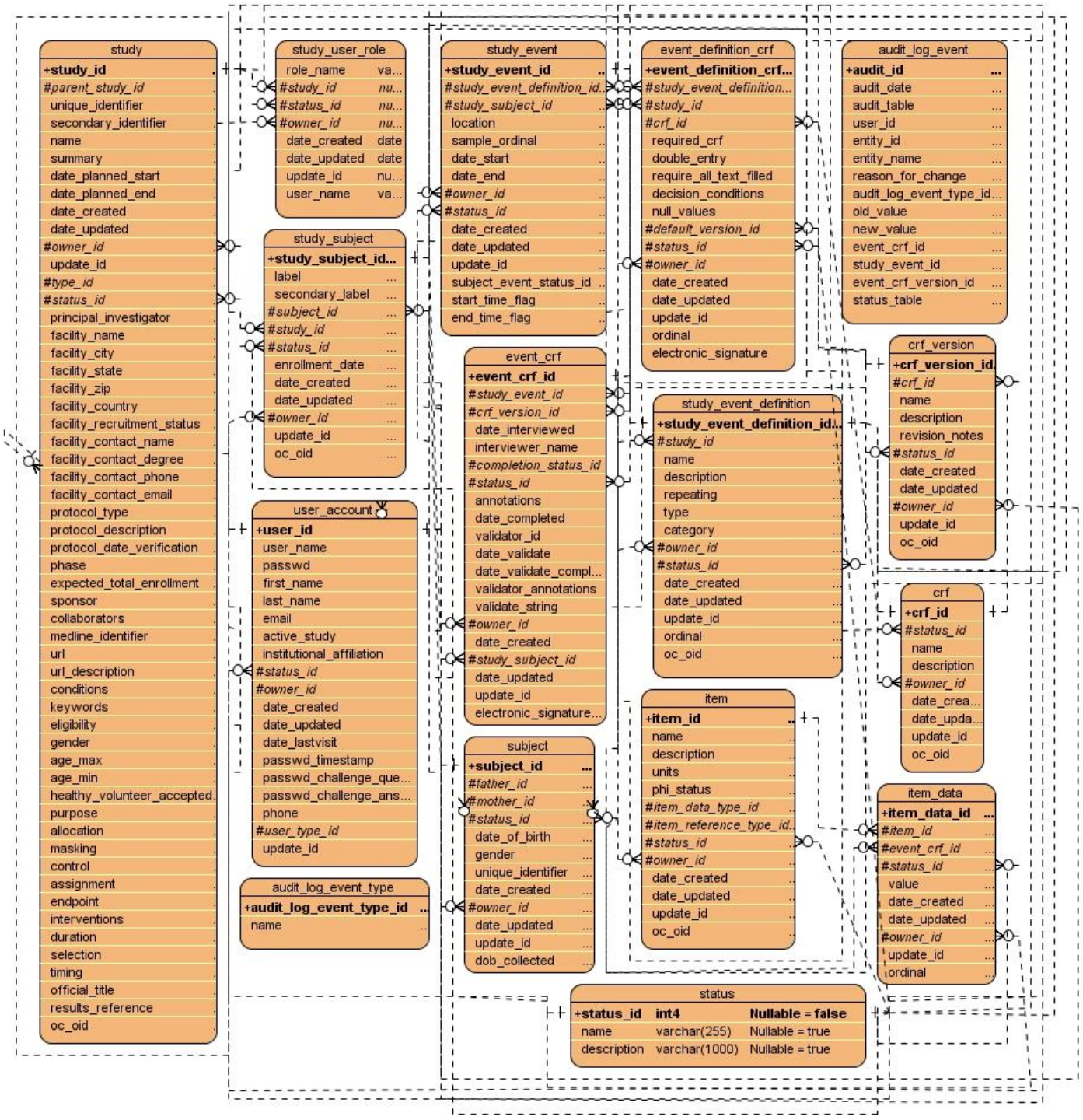

Figura 36 Modelo de Datos

Breve descripción de las clases

- Study: información referente al estudio o al centro, es recursiva debido a que los centros pertenecen a los estudios.
- Study_user_rol: información asociada al rol que ocupa un usuario dentro de un estudio específico.
- Study_subject: información de la asociación que existe entre los pacientes a los estudios.
- User_account: información de las cuentas de usuarios.
- Audit_log_event_type: son los tipos de acciones sobre las entidades del negocio
- Study_event: información de los momentos de seguimientos asignados a un paciente dentro de un estudio.
- Event_crf: información asociada al llenado de los cuadernos de datos que pertenecen a un momento de seguimiento especifico dentro de un estudio.
- Event_definition_crf: asociación de los cuadernos de recogida de datos a los momentos de seguimientos dentro de un estudio, esta tabla es parecida a la anterior con la diferencia que la primera comienza a guardar datos después que se comienza alguna acción sobre los cuadernos, sin embargo event_definition_crf es iniciada cuando se asigna un cuaderno a un momento de seguimiento.
- Subject: información de los pacientes.
- Status: información sobre los distintos estados por los que puedes pasar los objetos. Ej.: estudio *activo.*
- Ítem: información referente a las variables, son los campos que se encuentran en los modelos, análogamente constituyen elementos de las hojas electrónicas en formato Excel. En esta tabla no se recoge los valores reales solo información como el nombre y las descripciones etc.
- Ítem_data: información de los datos de las variables, es decir los valores reales.
- Audit_log_event: recoge la información para gestionar las trazas del sistema. Existe un campo dentro de esta tabla que es de suma importancia pues a través de este se puede acceder a la tabla que ha sido modificada en alguna de sus variantes: inserción, actualización, eliminación. Para lograr este objetivo se hace uso de los procedimientos almacenados (Triggers).
- Crf_version: información referente a las versiones de los cuadernos de datos, por medio de esta tabla es que los cuadernos se relacionan con el resto de las tablas.
- Crf: información sobre los cuadernos de recogía de datos. Esta tabla es nomencladora, es decir solo guarda datos de los CRDs.

3.5 Modelo de despliegue

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

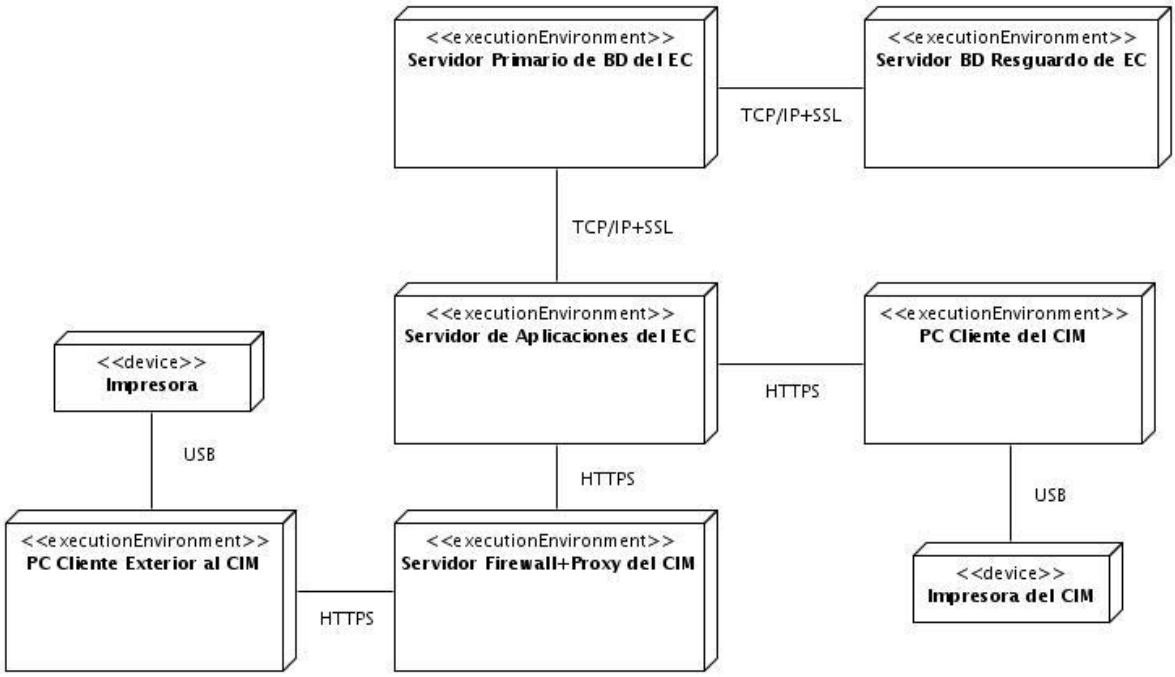

Figura 37 Diagrama de Despliegue

Breve descripción del Modelo de Despliegue

El servidor de aplicación se alojará en el centro promotor principal y el filtrado de paquetes será a través de un servidor firewall en conjunto con un proxy para la conexión con el exterior. El servidor de aplicación se integra con un servidor primario de base de datos para el almacenamiento de la información y este a su vez cuenta con un servidor auxiliar para el resguardo o duplicación de los datos y evitar pérdidas. Los clientes se pueden conectar vía HTTPS a la aplicación desde dentro del centro promotor o desde fuera. Cada una de las maquinas clientes contaran con servicios de impresión por medio de impresoras conectadas vía USB a las computadoras clientes. Los protocolos se emplean con el objetivo de lograr la seguridad en la transferencia de los datos, la rapidez de los mismos y evitar la intromisión de intrusos a la aplicación.

Conclusiones

Después de descrito todo el proceso de diseño están creadas las bases para la implementación y por tanto la investigación va arribando a los objetivos planteados.

Capitulo 4: Implementación del Sistema

Introducción

El objetivo de este capítulo es mostrar cómo está implementada la aplicación en términos de componentes. Se describen las principales funcionalidades que aportan a la investigación, ejemplificando con fragmentos de líneas de código. Se muestran ejemplos de las pantallas más relevantes o de las interfaces más importantes.

4.1 Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño tomando como ejemplo las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

Con la implementación la presente investigación sigue los siguientes propósitos:

- Distribuir el sistema asignando componentes ejecutables a nodos en el Diagrama de Despliegue.
- Implementar las clases y subsistemas encontrados durante el diseño. Las clases se implementan como componentes de ficheros que contienen código fuente.

4.2 Diagramas de Componentes.

A continuación se presenta el Diagrama de Componentes propuesto para la solución del software, los componentes no son más que el empaquetamiento físico de los elementos de un modelo, como el Modelo de Diseño, donde presentan varios estereotipos como ficheros ejecutables, ficheros de código fuente, entre otros. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas.

En el sistema desarrollado los componentes son ficheros con extensión .java y ficheros con extensión .jsp, los que contienen código Java y utilizan varias librerías del lenguaje, además de componentes de para la interacción con la Base de Datos que son ficheros con extensión .XML . Esto se puede apreciar en las figuras, en las cuales aparecen cada uno de ellos agrupados en paquetes.

Paquetes de implementación

Los paquetes de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en partes manejables. Un paquete puede estar formado por componentes, interfaces y otros paquetes. Estos están muy relacionados con los paquetes de diseño en el modelo del diseño, por lo que los paquetes de implementación deberían seguir la traza uno a uno de sus paquetes de diseño correspondientes. A continuación se mencionan cumpliendo con lo antes mencionado los paquetes de implementación:

1. Vista
2. Controlador
3. Modelo
4. Jsp's que crean la vista principal
5. Estilos y Scripts
6. Ficheros para internalización del lenguaje
7. Java.util
8. Propiedades DAO
9. DAO
10. BEAN
11. Base de Datos

El diagrama de componentes general en el que se evidencia la distribución a grandes rasgos de los paquetes de implementación según el patrón Modelo-Vista-Controlador se representa de la siguiente forma:

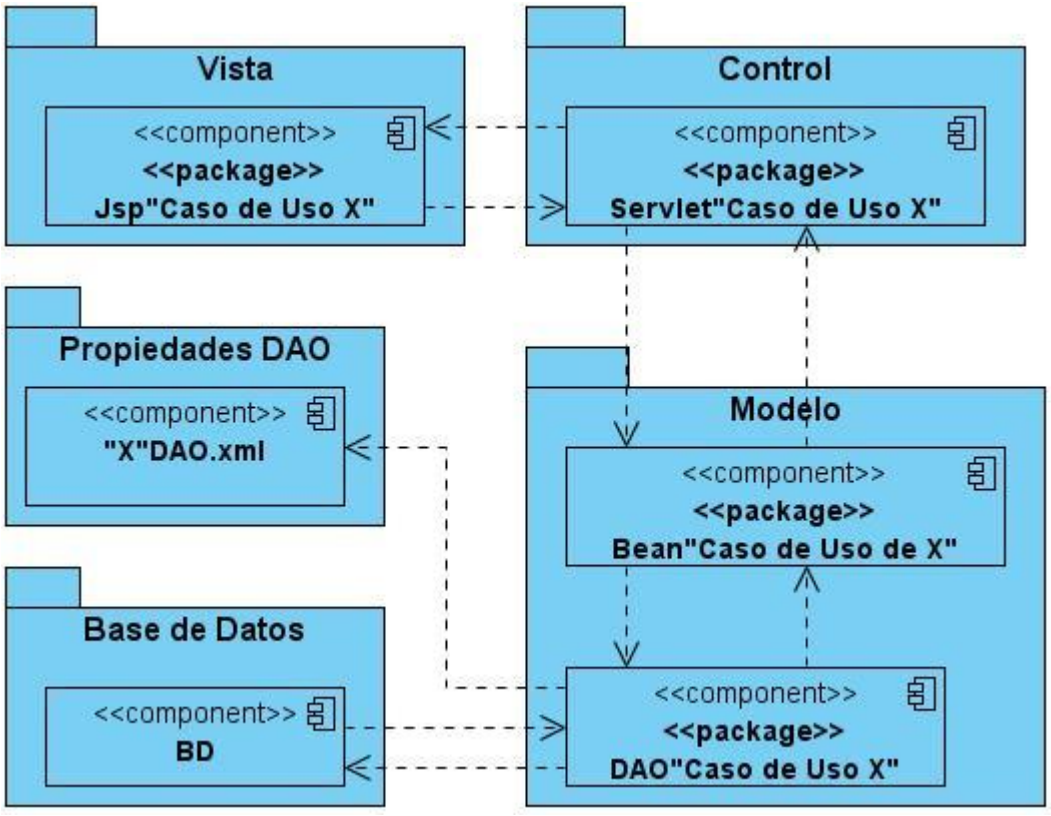

Figura 38 Diagrama General de Componente

A continuación se presentan los diagramas de componentes por los casos de del sistema trabajados hasta el momento, así como la distribución por paquetes de implementación de los componentes antes mencionados a un mayor número de detalle.

4.2.1 Caso de Uso Gestionar Trazas

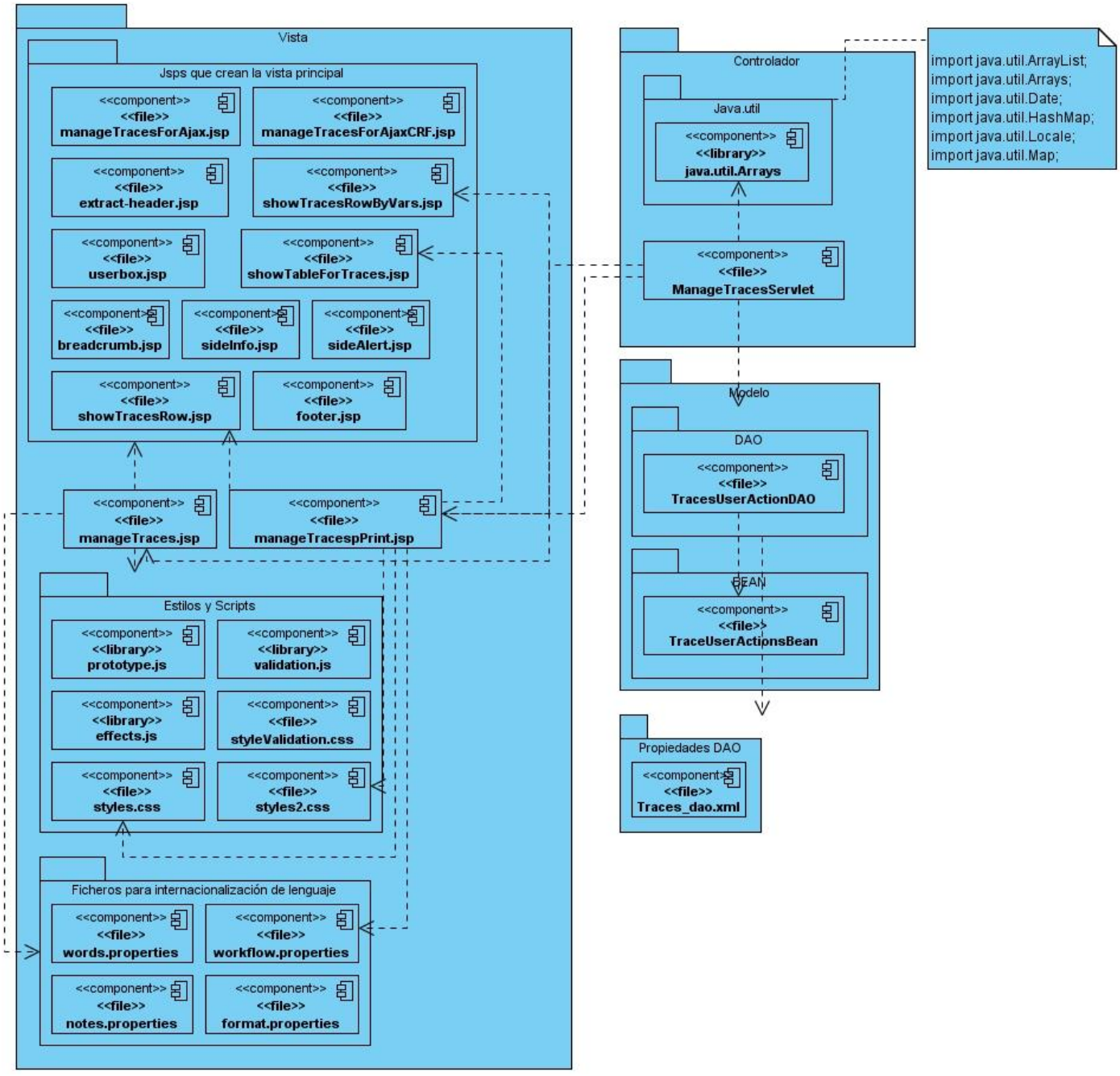

Figura 39 Diagrama de Componente Caso de Uso Gestionar Trazas

Como se ha venido explicando hasta el momento cada uno de los paquetes puede agrupar otros componentes. En la Figura 33, ejemplifica como es la colaboración entre los distintos paquetes. La Vista está compuesta por otros paquetes:

- Jsp's que crean la vista principal: los componentes contenidos construyen partes específicas de las interfaces que se le muestran al usuario y también insertan contenidos dinámicos en las mismas.
- Estilos y Scripts: contiene los estilos *.css para el diseño de las interfaces además de librerías con diversas funcionalidades. Por ejemplo prototype.js es una librería considerada un framework en sí mismo y actualmente es el más utilizado en el mundo entero debido a sus funcionalidades ya que simplifican la comunicación Ajax con el servidor y la inserción de los datos recibidos en el documento final así como el fácil acceso a elementos HTML. Validation.js es otra librería ampliamente utilizada y abstrae al desarrollador del proceso de validación de los formularios; se le pueden agregar efectos a las páginas con effects.js. Estas tres librerías generalmente trabajan en conjunto y son muy útiles.
- Ficheros para internalización del lenguaje: contienen todos los archivos para gestionar la internacionalización de la aplicación.

El paquete Vista además contiene todos los componentes fundamentales que introducen en la interfaz los datos dinámicos obtenidos desde el servidor, comportándose como controladores. Ej.: manageTraces.jsp.

- Controlador contiene los componentes que procesan todas las peticiones e interactúan con los componentes del Modelo existiendo dependencia entre ellos.
- Modelo: siguiendo las trazas desde el diseño los paquetes DAO y BEAN contienen los archivos que permiten la abstracción con la base de datos, la cual se modela como el componente Base de Datos.

En las figuras siguientes se modelan el resto de los diagramas correspondientes al resto de los Casos de Uso sobre los cuales ha estado dirigida la modificación y desarrollo de la aplicación.

4.2.2 Caso de Uso Visualizar Variables CRD

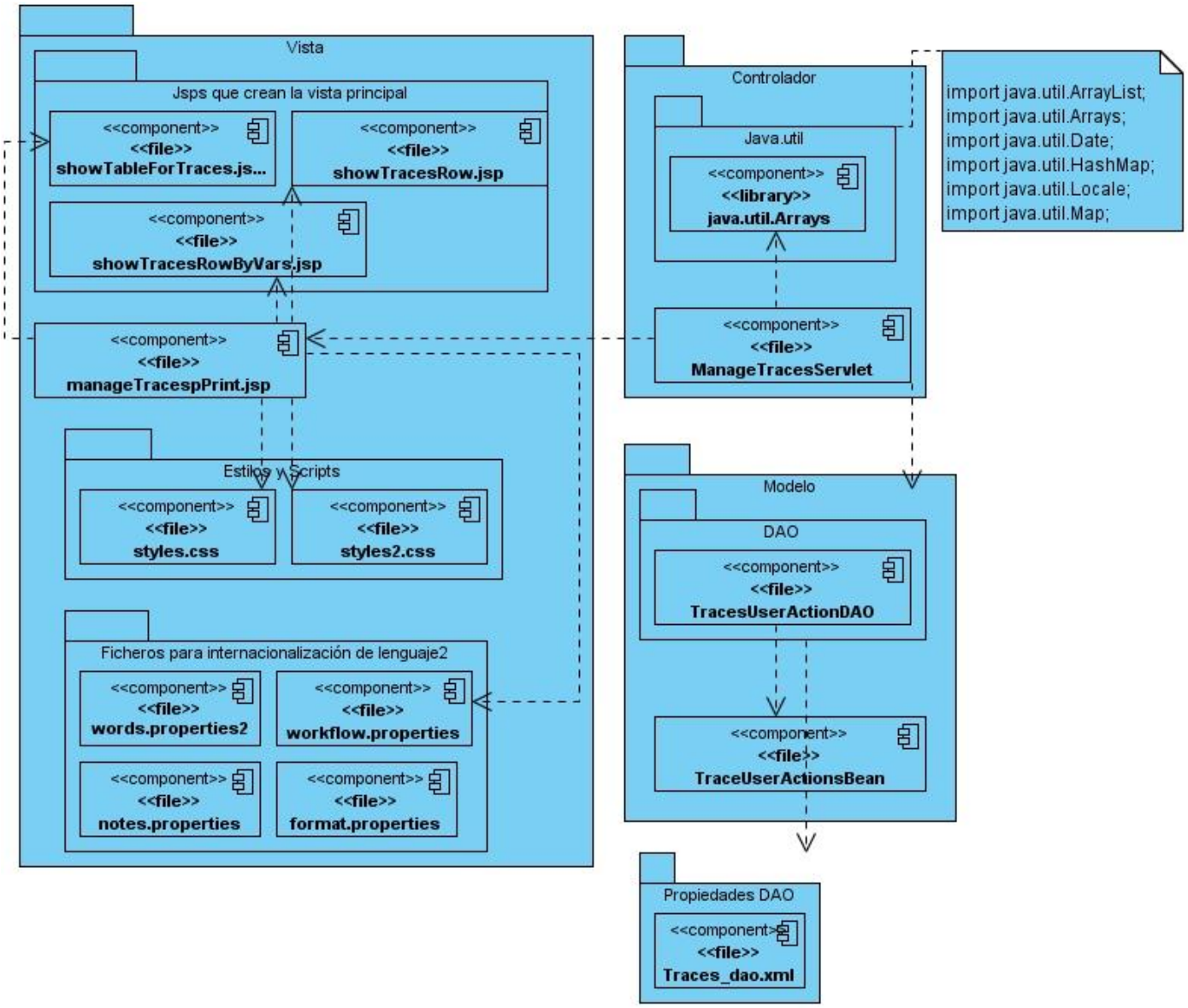

Figura 40 Diagrama de Componentes. Caso de Uso Visualizar Variables CRD

4.2.3 Caso de Uso Modificar Permisos Usuario

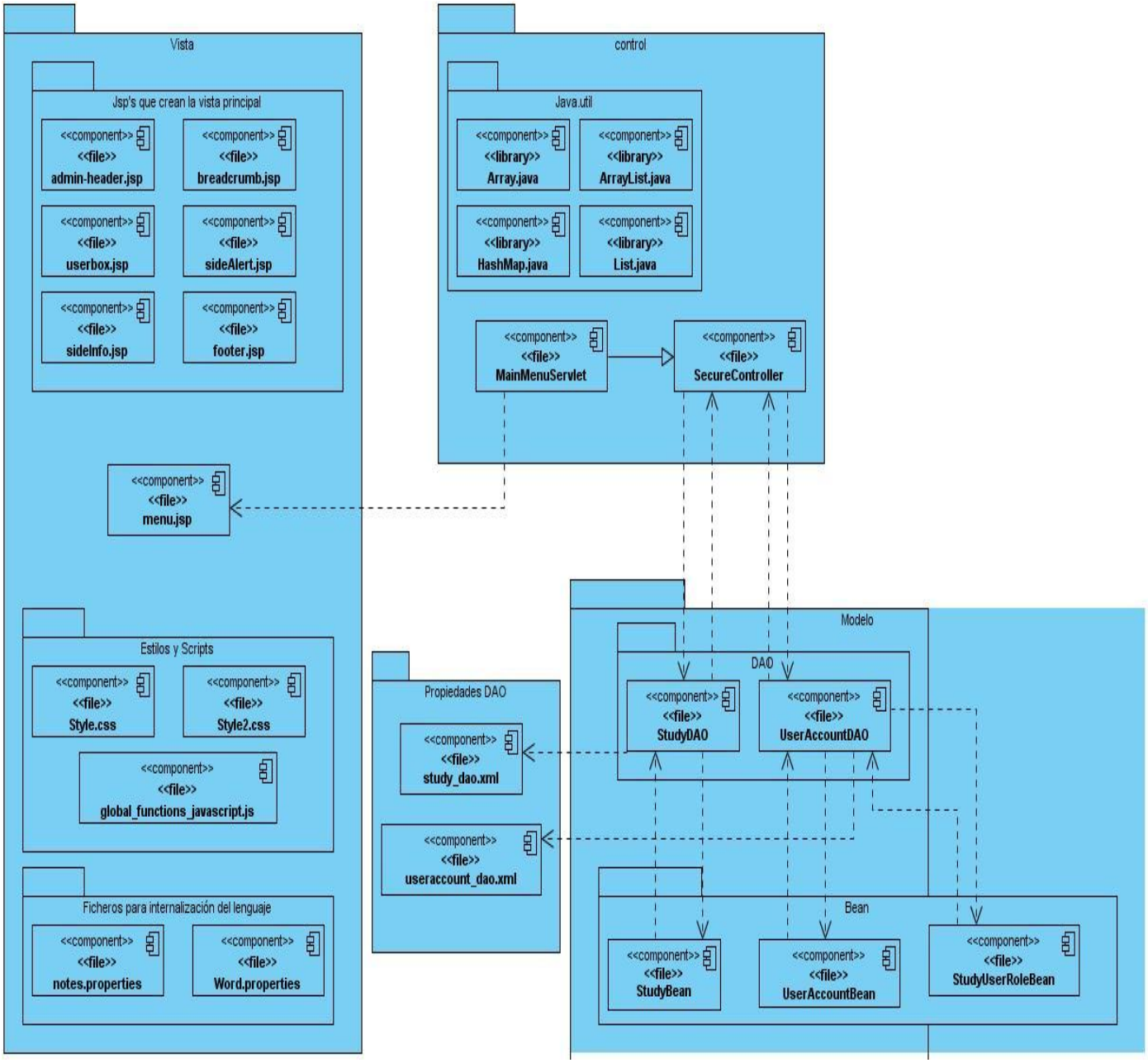

Figura 41 Diagrama de Componentes. Caso de Uso Modificar Permisos Usuarios.

4.2.4 Caso de Uso Gestionar IP Usuario

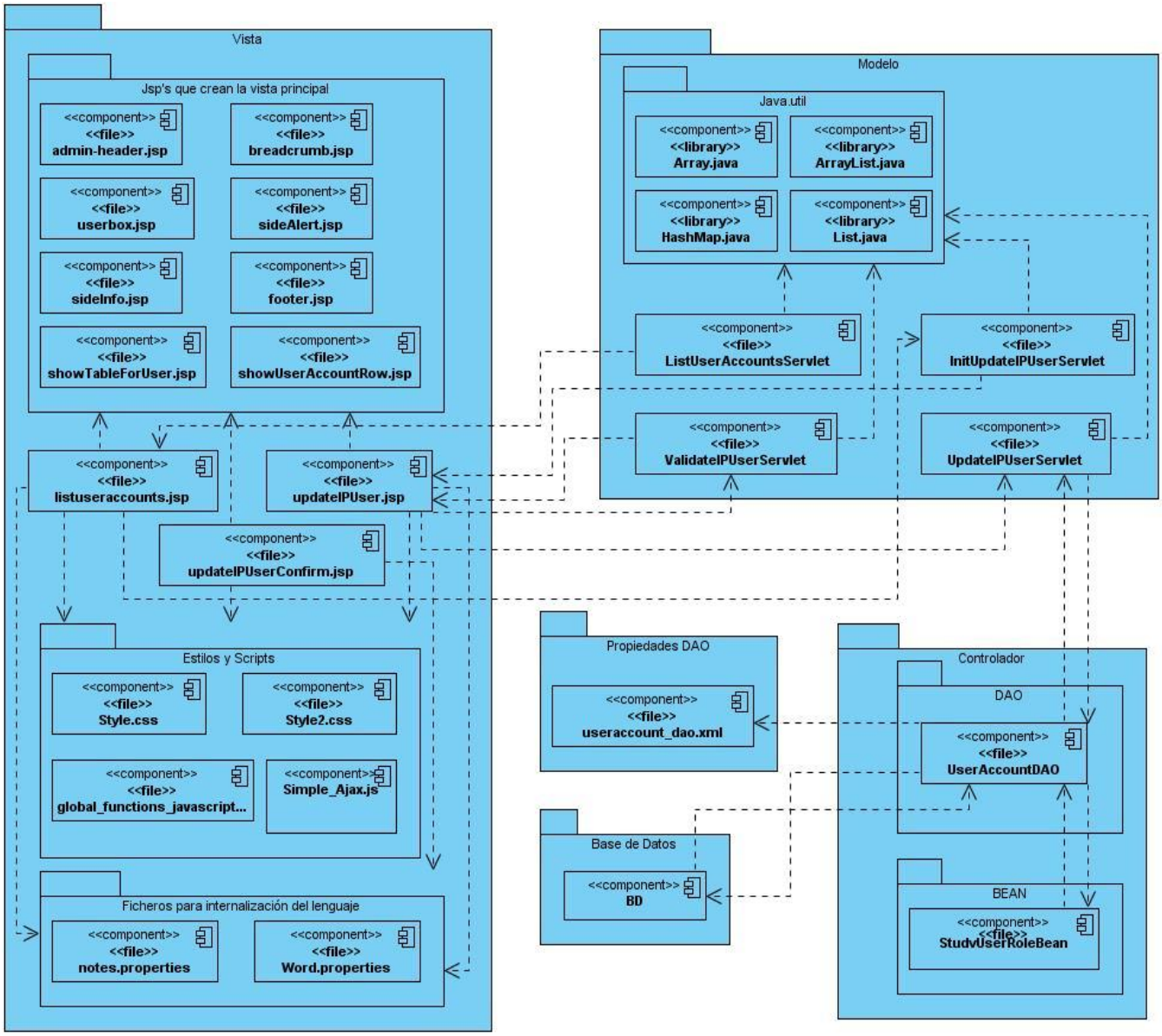

Figura 42 Diagrama de componentes Caso de Uso Gestionar IP Usuario

4.3 Funcionalidades del Sistema

Explicación del código

A continuación se presentan un grupo de las principales funcionalidades que se desarrollaron para la realización exitosa de la presente investigación.

El siguiente segmento de código se utiliza como parte de la seguridad del sistema y es el encargado de controlar el acceso desde una PC autorizada o no, a la página principal, es decir, el usuario que intente entrar en el sistema y tenga uno de los roles restringidos (Investigador Principal y Coordinador de la Investigación Clínica), tendrá que acceder al sistema desde una de las PC predefinidas en el centro a que pertenece el usuario en cuestión, sino será redireccionado a una página de error. Para esta gestión se toma en cuenta la dirección IP del cliente. Es decir que la dirección IP del usuario tiene que estar corresponderse a la dirección o rango de direcciones del centro o estudio al cual intenta conectarse si y solo si el rol del usuario es alguno de los anteriormente mencionados.

```
public void mayProceed() throws InsufficientPermissionException {
    locale = request.getLocale();
    String ip= request.getRemoteAddr();
    if(ub.isSysAdmin()||ub.isTechAdmin())
    {
        currentRole.setRole(Role.INVALID);
    }
    if((currentRole.getRole() == Role.RESEARCHASSISTANT || currentRole.isDirector()) && !ub.isSysAdmin())
    {
        if (currentRole.getRestring() == 1) {
            if (EsIgualIP(ip, currentRole.getIP()) || this.EsRangoValido(currentRole.getIp_Rango(),ip))
                return;
            else{
                addPageMessage(respage.getString("no_have_correct_privilege_current_study") +
                               respage.getString("change_study_contact_sysadmin"));
                forwardPage(Page.ERROR_IP);
            }
        }
        else if (  EsIgualIP(ip, currentStudy.getIP()) || this.EsRangoValido(currentStudy.getIP_RANGO(),ip))
        {
            return;
        }
        else{
            addPageMessage(respage.getString("no_have_correct_privilege_current_study") +
                           respage.getString("change_study_contact_sysadmin"));
            forwardPage(Page.ERROR_IP);
        }
    }
    else{

        return;
    }
}
```

Figura 43 Ejemplo de código empleada en la validación del acceso de los usuarios al sistema.

En el siguiente fragmento de código se muestran los métodos más importantes relacionados con la validación de IP. El primero verifica que el IP que se le pasa como parámetro sea una dirección IP válida de una institución, es decir, verifica si el IP está en uno de los rangos accesible para una institución, haciendo uso de expresiones regulares. El segundo método se encarga de convertir una

cadena de caracteres que forman una dirección IP en un número entero para su posterior trabajo y el tercer método retorna verdadero si una dirección IP está contenida en un rango de direcciones y falso si no se cumple esta condición.

```
function verificaHost(ip) {
    var n = "\\.(([1-9]?[0-9])|(1[0-9]{2})|(2([0-4][0-9]|5[0-5])))";   /* patrón para los números de .0 a .255
    var m = "\\.((1[6-9])|(2[0-9])|(3[01]))";          /* patrón para los números de .16 a .31 */
    if(ip.match("^192\\.168(" + n + "){2}$"))          /* 192.168.0.0 a 192.168.255.255 */
        return true;
    else if(ip.match("^10(" + n + "){3}$"))            /* 10.0.0.0 a 10.255.255.255 */
        return true;
    else if(ip.match("^172" + m + "(" + n + "){2}$")) /* 172.16.0.0 a 172.31.255.255 */
        return true;
    else if(ip.match("127.0.0.1"))
        return true;
    return false;          /* la dirección no coincidió con ninguna de las anteriores */
}
function ip2long(IPaddress) {
    var output;
    var parts = [];
        parts  = IPaddress.split('.');
        output = ( parts[0] * 16777216 +
        ( parts[1] * 65536 ) +
        ( parts[2] * 256 ) +
        ( parts[3] * 1 ) );
    return output;
}
function EsRangoValido(IP,Rango) {
    var ip=IP;
    var rango=new Array();
    rango=Rango.split("-");
        if ( (ip2long(rango[0]) <= ip) && (ip2long(rango[1])>= ip) )
            return true;
    return false;
}
```

Figura 44 Métodos empleados en la validación de direcciones IPs.

El código siguiente es un fragmento tomado del archivo de código Java encargado de insertar un usuario en el sistema. En la figura se muestra como se guardan en la variable de tipo HashMap los roles a que puede acceder el usuario en dependencia si es un Estudio o un Centro donde va a ser asignado el rol.

```
if (changeRoles) {
    StudyBean study = (StudyBean) sdao.findByPK(studyId);
    roleMap = new LinkedHashMap();
    ResourceBundle resterm = org.akaza.openclinica.i18n.util.ResourceBundleProvider.getTermsBundle();

    if (study.getParentStudyId() > 0) {
        for (Iterator it = getRoles().iterator(); it.hasNext();) {
            Role role = (Role) it.next();
            switch (role.getId()) {
                case 3: roleMap.put(role.getId(), resterm.getString("Study_Director").trim());
                    break;
                case 5: roleMap.put(role.getId(), resterm.getString("site_Data_Entry_Person").trim());
                    break;
                case 6: roleMap.put(role.getId(), resterm.getString("site_monitor").trim());
                    break;
                default: logger.info("No role matched when setting role description");
            }
        }
    } else {
        for (Iterator it = getRoles().iterator(); it.hasNext();) {
            Role role = (Role) it.next();
            switch (role.getId()) {
                case 2: roleMap.put(role.getId(), resterm.getString("Study_Coordinator").trim());
                    break;
                case 3: roleMap.put(role.getId(), resterm.getString("Study_Director").trim());
                    break;
                case 4: roleMap.put(role.getId(), resterm.getString("Investigator").trim());
                    break;
                case 5: roleMap.put(role.getId(), resterm.getString("Data_Entry_Person").trim());
                    break;
                case 6: roleMap.put(role.getId(), resterm.getString("Monitor").trim());
                    break;
                default: logger.info("No role matched when setting role description");
            }
```

¿Cómo colaboran los distintos componentes presente en los diagramas?

Para responder a estar interrogante se realizará la explicación del método *processRequest()* que responde al Caso de Uso Visualizar Variables CRD. El código se muestra a continuación:

```
public void processRequest() throws Exception {

    FormProcessor fp = new FormProcessor(request);  //crea una instancia de formulario
    String display="none";
    String display2="none";
    String formulario= fp.getString("formulario");  //obtiene los datos del formulario
    TracesUserActionDAO tracesdao= new TracesUserActionDAO(sm.getDataSource()); //instancia Objeto DAO
    ArrayList trazas=new ArrayList();               //guarda el resultado de la consulta
    String[] columns =null;
    if(!(request.getParameter("item_id") == null))  //Para el CU Visualizar Variables CRD
    {
        int item= fp.getInt("item_id");
        int eventCrf=fp.getInt("eventCRf");
        String []columnas={ resword.getString("user_name"), resword.getString("patient"),
        resword.getString("moment_definition"),resword.getString("CRF"),resword.getString("crf_version"),

        trazas=(ArrayList)tracesdao.findTracesByVarsLinkable(currentStudy.getId(), item,eventCrf);
        EntityBeanTable table = fp.getEntityBeanTable();  //intancia Tabla que contenrá los datos
        ArrayList allTraces = ManageTracesRow.generateRowsFromBeans(trazas); //guarda un arreglo de filas
        table.setColumns(new ArrayList(Arrays.asList(columnas))); // sitúa columnas en la tabla
        table.setQuery("ManageTraces", new HashMap());
        for (int i = 0; i < columnas.length; i++) {              //deshabilita vínculos de columnas
            table.hideColumnLink(i);
        }
        table.setRows(allTraces);                                //sitúa filas en la tabla
        table.setPaginated(false);                               //indica que no haya paginado
        table.computeDisplay();
        request.setAttribute("table", table);  //pasa el objeto tabla para HttpServletRequest
        resetPanel();
        formulario="1";
        request.setAttribute("formulario", formulario);
        forwardPage(Page.MANAGE_TRACES_PRINT);  //reenvia para la página que construirá la interfaz
        }
```

Figura 45 Método que responde al Caso de Uso Gestionar Variables CRD.

En el método *proccessRequest ()* una de las líneas más interesantes es la relacionada la instanciación del DAO. La clase SQLInitServlet es cargada cuando el servidor contenedor Apache Tomcat inicia lo cual está definido en el web.XML, archivo que proporciona la configuración y el despliegue de información para los componentes *web* que conforman la aplicación. Luego la clase mencionada crea un objeto SqlFactory, encargado de leer del directorio properties los archivos *dao.XML haciendo uso de la librería Digester. Este proceso se realiza con el objetivo de mantener en memoria desde el

principio todas las consultas a la base de datos y a través de los DAO llamarlas en el momento indicado. Volviendo al código después de hacer un breve resumen; con el objeto *tracesdao* se invoca el método *findTracesByVarsLinkable (),* el cual recibe por parámetros los identificadores de un estudio, una variable y un cuaderno, este método se explica a continuación y ejemplifica el uso de los DAO:

```
public Collection findTracesByVarsLinkable(int studyId, int item,int eventCRF) throws OpenClinicaException {

        this.unsetTypeExpected();
        this.setTypeExpected(1, TypeNames.STRING); //indica tipo de datos serán los campos de las tuplas
        this.setTypeExpected(2, TypeNames.DATE);
        this.setTypeExpected(3, TypeNames.STRING);
        this.setTypeExpected(4, TypeNames.STRING);
        this.setTypeExpected(5, TypeNames.STRING);
        this.setTypeExpected(6, TypeNames.STRING);
        this.setTypeExpected(7, TypeNames.STRING);
        this.setTypeExpected(8, TypeNames.STRING);
        this.setTypeExpected(9, TypeNames.STRING);
        this.setTypeExpected(10, TypeNames.STRING);
        this.setTypeExpected(11, TypeNames.STRING);

        ArrayList alist = null;
        HashMap variables=new HashMap();
        variables.put(new Integer(1), new Integer(studyId)); //para especificar los parámetros
        variables.put(new Integer(2), new Integer(eventCRF));
        variables.put(new Integer(3), new Integer(item));

        String query=digester.getQuery("findTracesByVarsLinkable"); //busca la consulta especificada
                                                                    //en el dao.xml correspondiente
        alist = this.select(query,variables);//ejecuta la consulta y guarda el resultado en la variable
        ArrayList al = new ArrayList();
        Iterator it = alist.iterator();

        while (it.hasNext()) {  //itera el resultado y lo adiciona en un ArrayList de objetos bean.
        TraceUserActionsBean tuactionbean = (TraceUserActionsBean) this.getEntityFromHashMap((HashMap) it.next()
        al.add(tuactionbean);
        }
        return al;  //devuelve el ArrayList con todos los objetos de tipo TraceUserActionBean
    }
```

Figura 46 Ejemplo de funcionalidad de un DAO

Este método se encarga de devolver el resultado de una consulta. La explicación se puede comprobar en los comentarios. El método *getentityFromHashMap ()* recibe una tupla, es decir campos que contienen los datos y convierte estos campos en atributos de un objeto bean que en este caso particular es un TraceUserActionBean, este objeto contiene la información de las trazas del sistema en dependencia de los parámetros pasados a la función. Estos objetos solo poseen atributos y los métodos set() y get() para acceder y modificarlos.

Una vez listo el resultado de la consulta el método principal prosigue con la creación de un objeto *table,* a este posteriormente se le añaden columnas y filas para su posterior visualización al usuario. El objeto se guarda en el objeto HttpServletRequest y es enviado a la clase manageTracesPrint.jsp para

crear la interfaz de la tabla. Estas páginas jsp (Java Server Pages) combinan HTML con fragmentos de Java para producir páginas web dinámicas. La tecnología permite generar páginas dinámicas a través de la inclusión de código Java en la propia HTML. Para la visualización de los elementos de la tabla se hace:

<jsp:useBean scope="request" id="table" class="org.akaza.openclinica.core.EntityBeanTable" />
……
* <c:import url="../include/showTableForTraces.jsp"><c:param name="rowURL"*
*value="showTracesRowByVars.jsp"/></c:import> *

Estas no son las únicas líneas de la página manageTraces.jsp, pero si las más importantes pues son las que permiten visualizar el contenido de la tablas. La primera hace uso de la etiqueta JSTL jsp:useBean, para recuperar el objeto *table* de tipo *EntityBeanTable* que se paso por HttpRequest. Los archivos *showTableForTraces.jsp y showTracesRowByVars.jsp* trabajan en conjunto para dibujar la tabla. *Table;* que es pasado por request contiene objetos de tipo *ManageTracesRow y estos de igual forma auxiliándose de los JavaBean son usados desde la página showTracesRowByVars.jsp* para pintar cada una de las filas que contendrá la tabla final, es decir en *showTracesRowByVars.jsp se hace:*

<jsp:useBean scope="request" id="currRow"
……..
class="org.akaza.openclinica.bean.extract.ManageTracesRow" ></jsp:useBean>
<tr valign="top" >
<td class="table_cell_left" align="center"><c:out value="${currRow.bean.user_name}"/></td>
</tr>

Como se ha podido apreciar el proceso de va desde las Servlet y hasta las JSP y construir las interfaces es muy fácil. Hasta el momento se ha explicado de forma sencilla los principales elementos del código y como colaboran cada uno de los componentes de la aplicación.

4.4 Pantallas de la aplicación

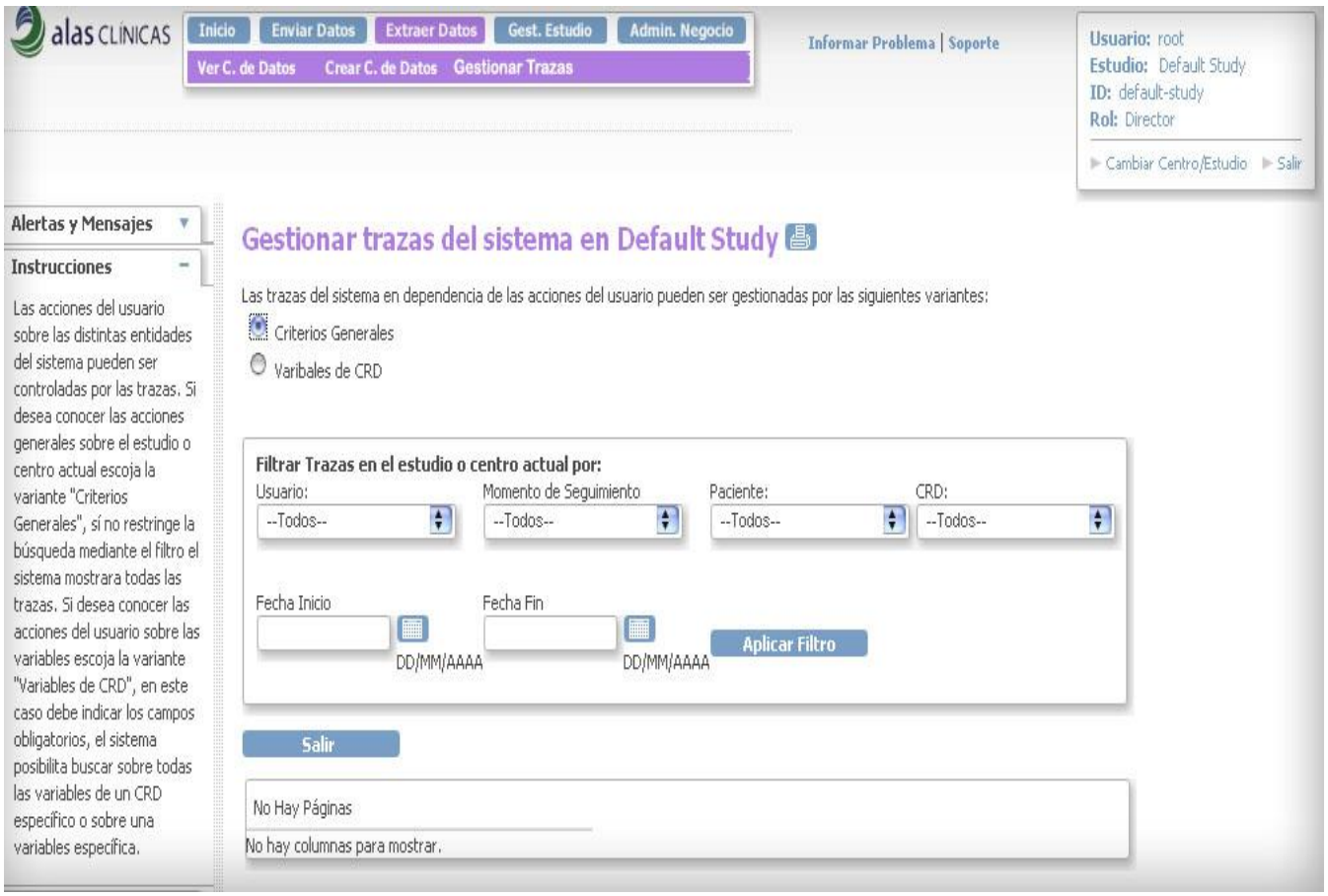

Figura 47 Pantalla correspondiente al Caso de Uso Gestionar Trazas

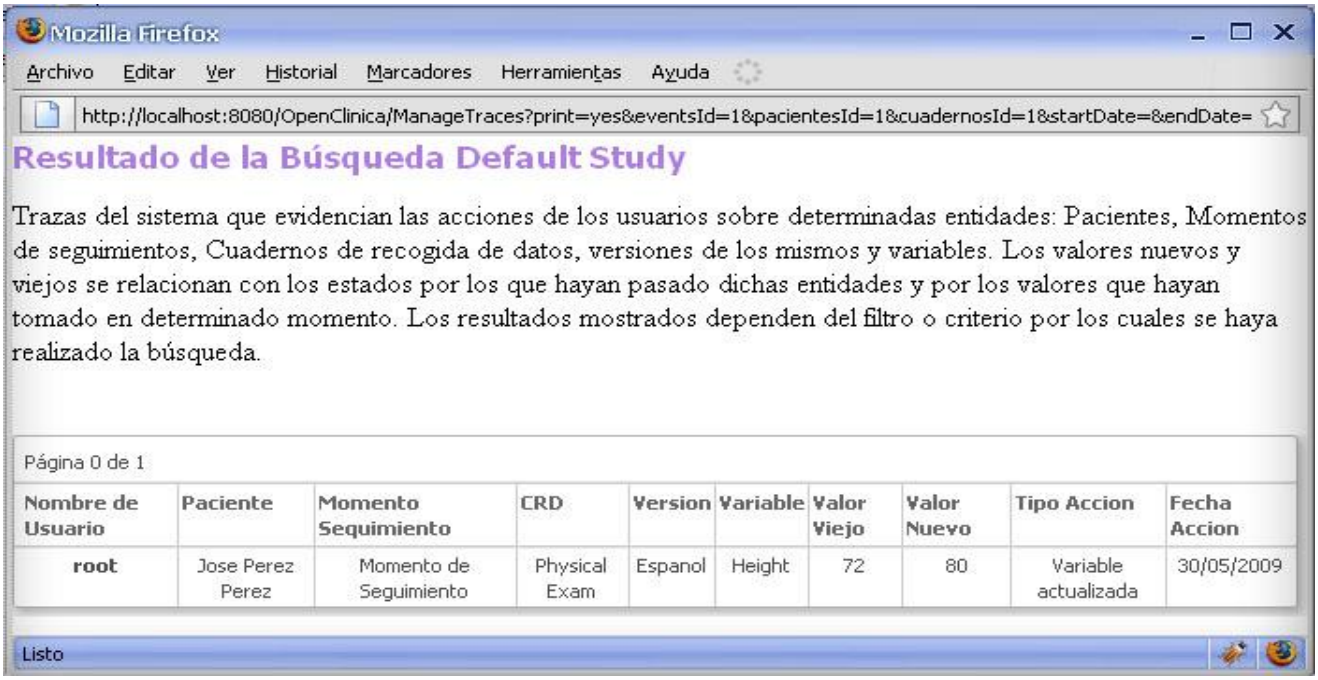

Figura 48 Pantalla Escenario Imprimir. Caso de Uso Gestionar Trazas

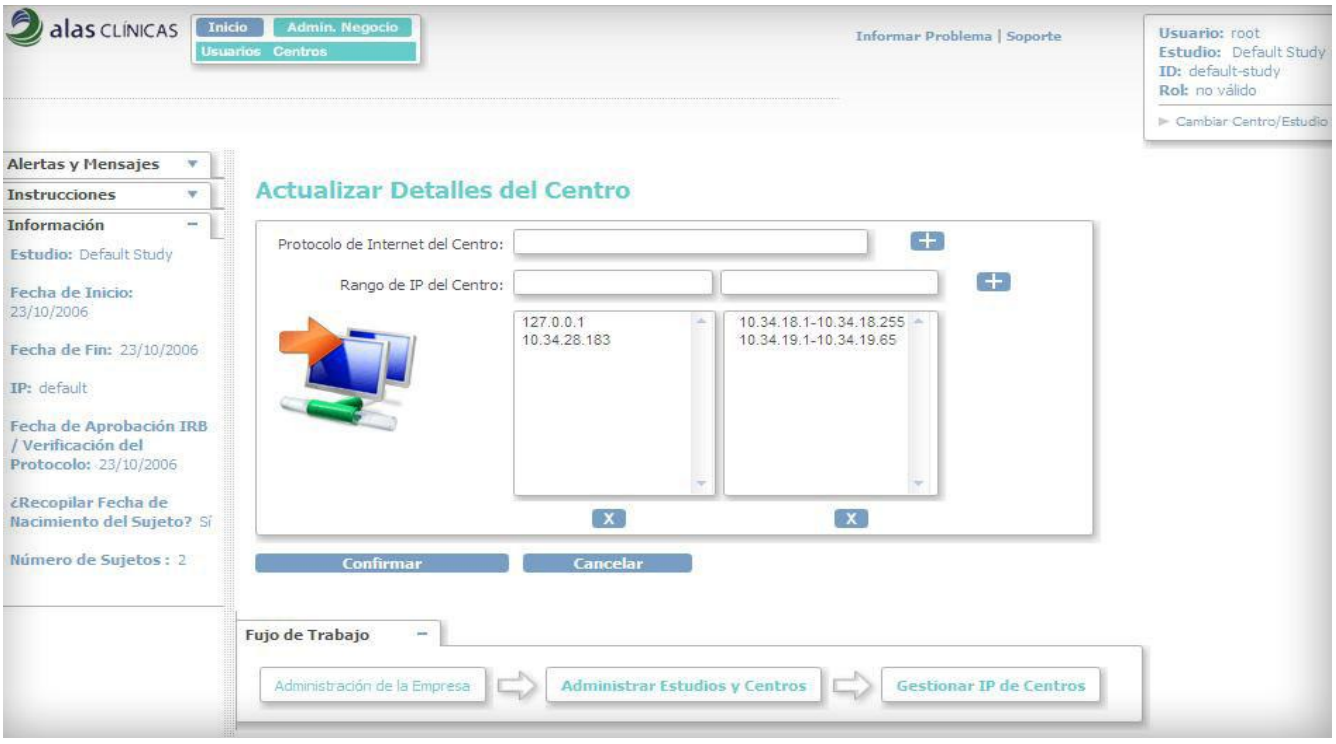

Figura 49 Pantalla correspondiente al Caso de Uso Gestionar IP Centro.

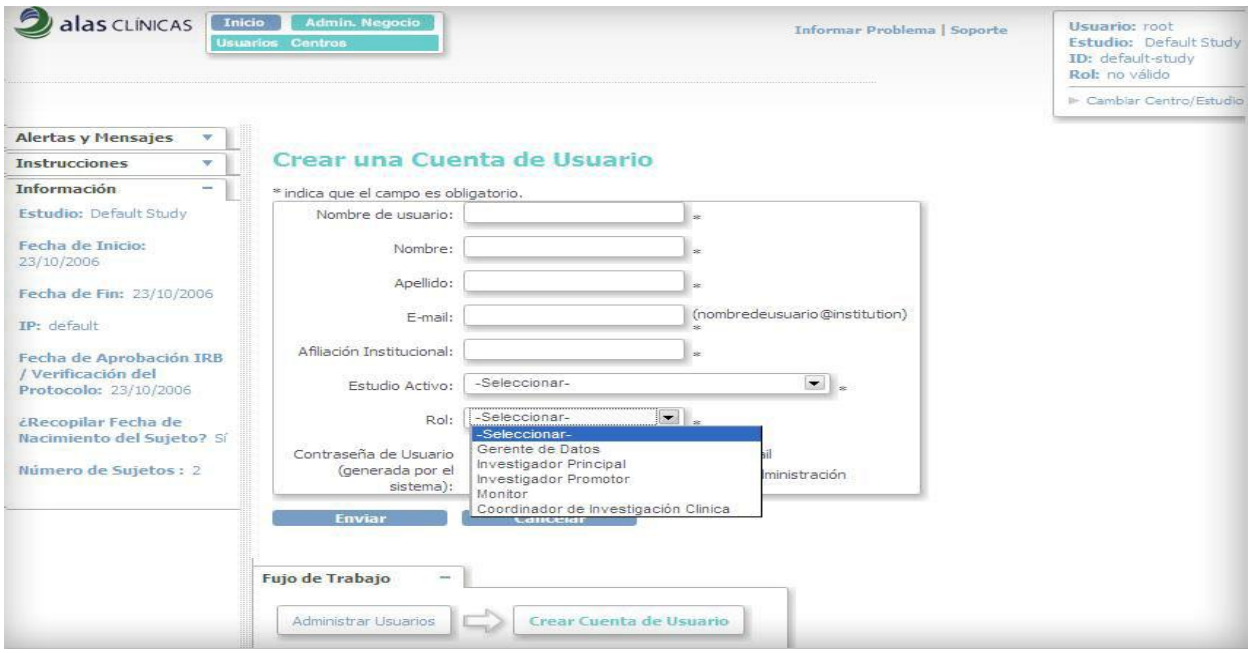

Figura 50 Pantalla correspondiente al Caso de Uso Insertar Usuario.

Figura 51 Pantalla de error correspondiente al Caso de Uso Modificar Permisos Usuarios.

Conclusiones

Con este capítulo se describió la implementación del sistema en términos de componentes, modelando los artefactos asociados, así como los principales métodos y la filosofía de trabajo entre estos. Se ejemplificó con fragmentos de códigos y se mostraron pantallas correspondientes a las funcionalidades desarrolladas en la investigación.

Conclusiones Generales

Después de llevar a cabo un conjunto de tareas se pudieron concretar los objetivos de la investigación. Se realizó un estudio exhaustivo de OpenClínica, resultando ser un paso primordial para el entendimiento del negocio particular que resuelve esta herramienta, además para poder adaptarla a las necesidades cubanas en cuanto al proceso de conducción de los ensayos clínicos en Cuba. Se puntualizaron los requisitos que debía cumplir la aplicación para dar respuesta a las necesidades particulares del Centro de Inmunología Molecular y a partir de aquí se fueron implementando cada una de las funcionalidades necesarias, documentando en todo momento con los artefactos correspondientes a cada fase del ciclo de vida del producto final. La investigación aportó además un acercamiento a las nuevas tecnologías y tendencias del mundo actual en el desarrollo de software, en cuanto al uso de patrones de diseño, estilos arquitectónicos, lenguajes de programación para desarrollo WEB, herramientas de desarrollo, gestores de base de datos y protocolos de comunicación. Quedaron abiertas las puertas para el aprendizaje y mejoramiento de la aplicación informática modificada.

Recomendaciones

Se recomienda seguir profundizando en los procesos de conducción de ensayos clínicos en Cuba para seguir añadiendo funcionalidades útiles a la aplicación. Esta no fue implementada sobre ningún framework de desarrollo por lo que en un futuro pudiera ser migrada a alguno como por ejemplo Spring.

Referencias Bibliográficas

[1]. ALFONSO, Carmen. El problema del cáncer, ¿cómo podemos enfrentarlo? Trabajadores.CU [en línea] (2006). [Consulta el: 2 de enero 2009]. Disponible en: <http://www.trabajadores.cu/materiales_especiales/suplementos/salud/enfermedades-cronicas-no-trasmisibles-1/el-problema-del-cancer-bfcomo-podemos-enfrentarlo>

[2]. OMS. Cáncer, 2008. [Consulta el: 2 de enero 2009]. Disponible en: <http://www.who.int/mediacentre/factsheets/fs297/es/index.html >

[3]. ALFONSO, Carmen. El problema del cáncer, ¿cómo podemos enfrentarlo? Trabajadores.CU [en línea] (2006). [Consulta el: 2 de enero 2009]. Disponible en: <http://www.trabajadores.cu/materiales_especiales/suplementos/salud/enfermedades-cronicas-no-trasmisibles-1/el-problema-del-cancer-bfcomo-podemos-enfrentarlo>

[4]. RODRÍGUEZ CLAVIJO, Kenia, SEGURA DURÁN, Daili. *Sistema de Manejo de Datos de Ensayos Clínicos: Módulo Publicador: Diseño e implementación del submódulo: "Gestión de información de pacientes y Cuadernos de Recogida de Datos".* Tesis inédita, Universidad de Ciencias Informáticas, Ciudad de la Habana, 2008.

[5]. RODRÍGUEZ CLAVIJO, Kenia, SEGURA DURÁN, Daili. *Sistema de Manejo de Datos de Ensayos Clínicos: Módulo Publicador: Diseño e implementación del submódulo: "Gestión de información de pacientes y Cuadernos de Recogida de Datos".* Tesis inédita, Universidad de Ciencias Informáticas, Ciudad de la Habana, 2008

[6]. VALERIO, María. ¿Qué es un Ensayo Clínicos? Actualizado 29 de septiembre de 2003. [en línea] [Consulta el : 3 de enero 2009.] Disponible en: <http://www.elmundo.es/elmundosalud/2003/09/29/oncodossiers/1064858962.html>

[7]. AZNAR-SALATTI, J. Diseño de los protocolos de un estudio clínico: las denominadas "case report forms" o cuadernos de recogida de datos. Revista Jano. [en línea] (1997); 51(35):67-74. [Consulta el : 5 de enero de 2009]. Disponible en : <http://www.uv.es/~docmed/documed/documed/591.html >

[8]. LÓPEZ DÍAS, Aidacelys, RODRÍGUEZ GARCÍA, Lucia. *Sistema de Manejo de Datos de Ensayos Clínicos: Módulo de Diseño.* Tesis inédita, Universidad de las Ciencias Informáticas. Ciudad de la Habana, 2007.

[9]. Introducción al proceso de desarrollo de software. [en línea]. [Consultado el : 6 de enero de 2009] Disponible en: <http://teleformacion.uci.cu/file.php/102/Curso_2008-2009/Materiales_Basicos/Materiales_Basicos_Conf_1/Clase_1_Curso_Proceso_de_Desarrollo_de_Software.pdf >

[10]. Metodologías Ágiles en el Desarrollo de SW. [en línea](2007) [Consultado el: 7 de enero de 2009.] Disponible en: <http://teleformacion.uci.cu/file.php/102/Curso_2008-2009/Materiales_Complementarios/Materiales_Complementarios_Conf_1/Metodologias_Agiles_en_el_Desarrollo_de_SW.pdf>

[11]. ROMINA LEDESMA, Claudia, ALEJANDRA ALÍ, Claudia. Trabajo de Investigación: J2EE. [Consultado el: 15 de marzo de 2009].

[12]. VARGAS, Alan. El fenómeno de la tecnología Java. [en línea] [Consultado el: 13 de marzo del 2009] Disponible en:< http://java-team-uaem.googlegroups.com/web/QueEsJava.pdf?gda=isoviT8AAABGd9T681z5DpzhkclInlFVt7VqMa6G4hs1O-LVG7TB_0rUwu70cYXcXQOwnJjszyaccyFKn-rNKC-d1pM_IdV0 >

[13]. Gestor de Base de Datos: MySQL, PostgreSQL, SQLite. [en línea] [Consultado el: 30 de noviembre del 2008] Disponible en: <http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html>.

[14]. Visual Paradigm para UML. [en línea](2007) [Consultado el : 17 de marzo de 2009]. Disponible en: <http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D%20_14720_p/ >

[15]. GUTIÉRREZ MAYORAL, Antonio. Front-end y biblioteca de interacción con Subversión para GNOME con tecnologías C# y Mono. [en línea] (2006) [Consultado el: 17 de marzo de 2009.] Disponible en: <http://gsyc.es/~agutierr/pfc-tecnica-html/node11.html>

[16]. Conferencia_2_de_Arquitectura_2009. [en línea][Consultado el 4 de abril de 2009] Disponible en: <http://teleformacion.uci.cu/file.php/259/CURSO_2008-2009/Materiales_Basicos/Semana_3/Conf/Conferencia_2_de_Arquitectura_2009ok.doc>

[17]. Core J2EE Pattern Catalog. [en línea] [Consultado el 4 de mayo de 2009] Disponible en: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>

[18]. Patrones de Diseño en aplicaciones Web con Java J2EE. [en línea][Consultado el 4 de mayo de 2009] Disponible en: <http://java.ciberaula.com/articulo/diseno_patrones_j2ee/>

[19]. Patrones para asignar responsabilidades. [en línea] [Consultado el 4 de mayo de 2009] Disponible en: <http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/PATRONES_A9.pdf>

ANEXOS

Caso de uso Gestionar IP Estudio/Centro

Descripción de Casos de Uso Gestionar IP Estudio/Centro

Caso de Uso:	Gestionar IP Estudio/Centro
Actores:	Administrador (Inicia)
Resumen:	El administrador selecciona la opción Gestionar Centros y estudios dentro del módulo Administrar Empresa. El sistema muestra listado con los estudios y los centros que pertenecen a estos, visualizando los IPs y rangos de IPs asignados. El administrador indica gestionar IP de un estudio específico, el sistema muestra las opciones disponibles. El administrador confirma sus acciones y el caso de uso termina si el usuario selecciona otra opción del sistema.
Precondiciones:	El usuario debe estar registrado como Administrador.
Referencias	RF 1, RF 2, RF 3, RF 4, RF 5, RF 6 :CU Validar Gestión IP
Prioridad	4

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción Gestionar Estudios y centros del módulo Administrar Empresa.	1.1 El sistema muestra un listado con los estudios que existen en el sistema visualizando los IPs y los rangos de IPs que tienen asignados, además de las acciones disponibles para cada uno de ellos.
2. El administrador indica Gestionar IP. Si el administrador escoge la opción: • Gestionar IP de Centro. Ir a sección “Gestionar IP de Centro”. • Gestionar IP de Estudio. Ir a sección “Gestionar IP de Estudio”.	

3. El administrador puede indicar cualquiera de las dos operaciones sin necesidad de guardar los datos en la Base de Datos. Si el administrador escoge la opción: • Adicionar. Ir a sección Adicionar IPs y/o rango de IPs. • Eliminar: Ir a sección Eliminar IPs y/o rango de IPs.	
4. El administrador selecciona la opción Confirmar.	4.1 El sistema muestra una interfaz de confirmación de los datos que se gestionaron.
5. El administrador selecciona la opción Enviar.	5.1 El sistema actualiza la información relacionada con los IPs y/o rango de IPs del estudio en cuestión y redirecciona al listado de estudios. Ir a paso 1.1 del flujo normal de eventos.
Sección “Gestionar IP de Centro.”	
Acción del Actor	Respuesta del Sistema
1. El administrador indica Gestionar IP de un Centro específico.	1.1 El sistema muestra una interfaz con los IPs y rangos de IPs asignados al Centro escogido. Además permite al administrador realizar varias operaciones tanto con los IPs como con los rangos: • Adicionar • Eliminar Ir al paso 3 del flujo normal de eventos.
Sección “Gestionar IP de Estudio.”	
Acción del Actor	Respuesta del Sistema
1. El administrador indica Gestionar IP de un Estudio específico.	1.1 El sistema muestra una interfaz con los IPs y rangos de IPs asignados al Estudio escogido. Además permite al administrador realizar varias operaciones

	tanto con los IPs como con los rangos: • Adicionar • Eliminar Ir al paso 3 del flujo normal de eventos.
Sección “Adicionar IPs y/o rango de IPs.”	
Acción del Actor	Respuesta del Sistema
1. El administrador entra los datos ya sea IP y/o rango de IPs y selecciona la opción Adicionar.	1.1 El sistema valida los datos y si no existen errores adiciona los datos insertados al listado de IPs y/o rango de IPs respectivamente. (Ver CU Validar Gestión IP). Ir al paso 3 del flujo normal de eventos.
Sección “Eliminar IPs y/o rango de IPs.”	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona del listado de IPs y/o rango de IPs el valor que desea eliminar e indica realizar la operación con la opción Eliminar.	1.1 El sistema elimina por cada petición del usuario un IP y/o un rango de IPs. Ir al paso 3 del flujo normal de eventos.
Flujos Alternos	
Flujo alterno de eventos acción 2	
Acción del Actor	Respuesta del Sistema
2.a El caso de uso termina si el administrador escoge otra opción disponible en el sistema.	
Flujo alterno de eventos acción 4	
Acción del Actor	Respuesta del Sistema
4.a El administrador selecciona la opción Cancelar.	4.b El sistema redirecciona al listado de estudios. Ir a paso 1.1 del flujo normal de eventos.
Flujo alterno de eventos acción 5	
Acción del Actor	Respuesta del Sistema
5.a El administrador escoge la opción atrás.	5.b El sistema vuelve a la página anterior

Flujo alterno para Sección Adicionar IPs y/o rango de IPs acción 1.1	
Acción del Actor	Respuesta del Sistema
	1.1a Si el sistema al validar los datos encuentra errores, muestra el mensaje de error en dependencia del mismo y no envía la información obligando al administrador a corregir los datos.
1.1b El administrador corrige los datos y procede a introducir los datos nuevamente. Ir al paso 1 del flujo normal de eventos de la sección "Adicionar IPs y/o rango de IPs".	

Prototipo de Interfaz

Figura 52 Prototipo de interfaz Administrar Centros y estudios

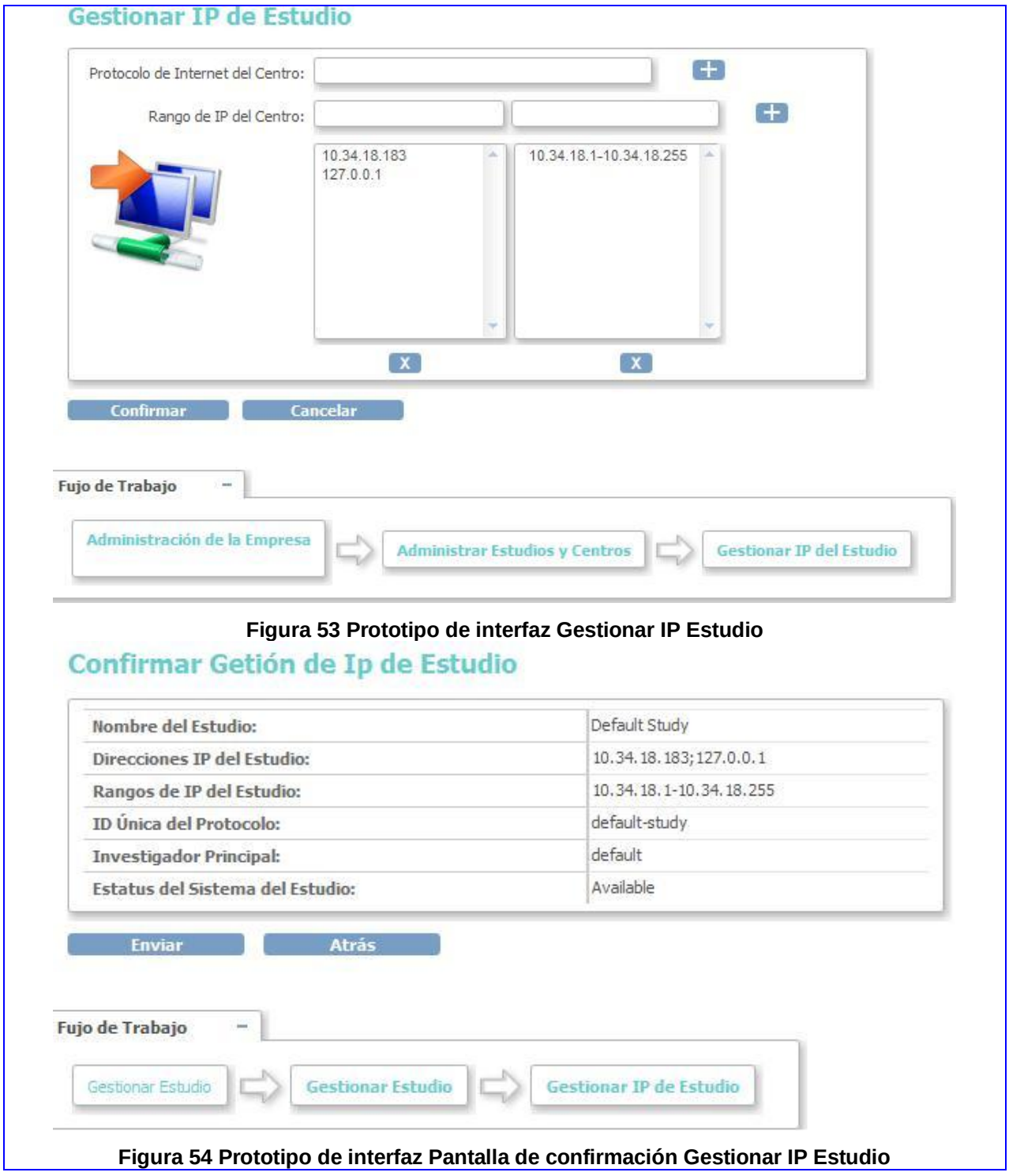

Figura 53 Prototipo de interfaz Gestionar IP Estudio

Figura 54 Prototipo de interfaz Pantalla de confirmación Gestionar IP Estudio

Figura 55 Prototipo de Prototipo de interfaz Gestionar IP Centro

Figura 56 Prototipo de interfaz Pantalla de confirmación Gestionar IP Centro

Poscondiciones	Se modifica la información referente a los IPs y/ rango de IPs de los

	Estudios/Centros, añadiendo, eliminando o editando nuevos IP y/o rangos de IPs.

Caso de uso Insertar Estudio

Descripción de Casos de Uso Insertar Estudio

Caso de Uso:	Insertar Estudio
Actores:	Gerente de Datos (Inicia)
Resumen:	El caso de uso inicia cuando el gerente de datos selecciona la opción Estudios en el módulo Gestionar Estudio. El sistema muestra un listado con los estudios que han sido creados y los centros que están asociados a los mismos, el gerente de datos indica crear un Nuevo Estudio, el sistema permite la entada de los datos, pide confirmación. Una vez creado el estudio este pasa al estado de “Diseño”. El caso de uso termina cuando el gerente de datos selecciona otra opción disponible en el sistema.
Precondiciones:	El usuario se haya registrado como Gerente de Datos.
Referencias	RF 14
Prioridad	3

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El gerente de datos selecciona la opción Estudios en el módulo Gestionar Empresa.	1.1 El sistema muestra un listado con la información asociada a los estudios existentes, además de las acciones disponibles sobre los mismos, deshabilitando la creación de centros pues no es una acción accesible por el gerente de datos.
2. El administrador indica Crear Nuevo Estudio.	2.1 El sistema muestra los campos de la sección que debe llenar el gerente de datos indicando siempre los datos obligatorios.
3. El gerente de datos llena los datos de la	3.1 El sistema valida los datos.

sección en que se encuentra e indica la opción Continuar.	3.2 Si se han llenado los datos correspondientes a todas las secciones el sistema muestra la interfaz de confirmación de creación de estudio.
4. El gerente de datos indica la opción Enviar Estudio.	4.1 El sistema asigna al estudio la dirección IP de la computadora desde donde se crea el estudio y guarda los datos correspondientes. El estudio adquiere un nuevo estado: “Diseño”. El sistema redirecciona al listado de estudios. Ir al paso 1.1 del flujo normal de eventos.
Flujos Alternos	
Flujo alterno de eventos acción 2	
Acción del Actor	Respuesta del Sistema
2.a El caso de uso termina si el gerente de datos escoge otra opción disponible en el sistema.	
Flujo alterno de eventos acción 3	
Acción del Actor	Respuesta del Sistema
3.a El gerente de datos indica la opción Cancelar.	3.b El sistema redirecciona al listado de estudios. Ir al paso 1.1 del flujo normal de eventos.
Flujo alterno de eventos acción 3.1	
Acción del Actor	Respuesta del Sistema
	3.1a El sistema valida los datos y encuentra errores, indicando al gerente de datos la corrección de los mismos con un mensaje sugerente. Ir a paso 3 del flujo normal de eventos.
Flujo alterno de eventos acción 3.2	
Acción del Actor	Respuesta del Sistema
	3.2.a En caso de que no se hayan llenado

	los datos de todas las secciones correspondientes a la creación de un estudio nuevo el sistema pasa a la sección siguiente. Ir al paso 3 del flujo normal de eventos.
Flujo alterno de eventos acción 4	
Acción del Actor	Respuesta del Sistema
4.a El gerente de datos indica la opción Anular Creación.	4.1a El sistema redirecciona al listado de estudios. Ir al paso 1.1 del flujo normal de eventos.

Prototipo de Interfaz

Figura 57 Prototipo de interfaz Gestionar Estudios

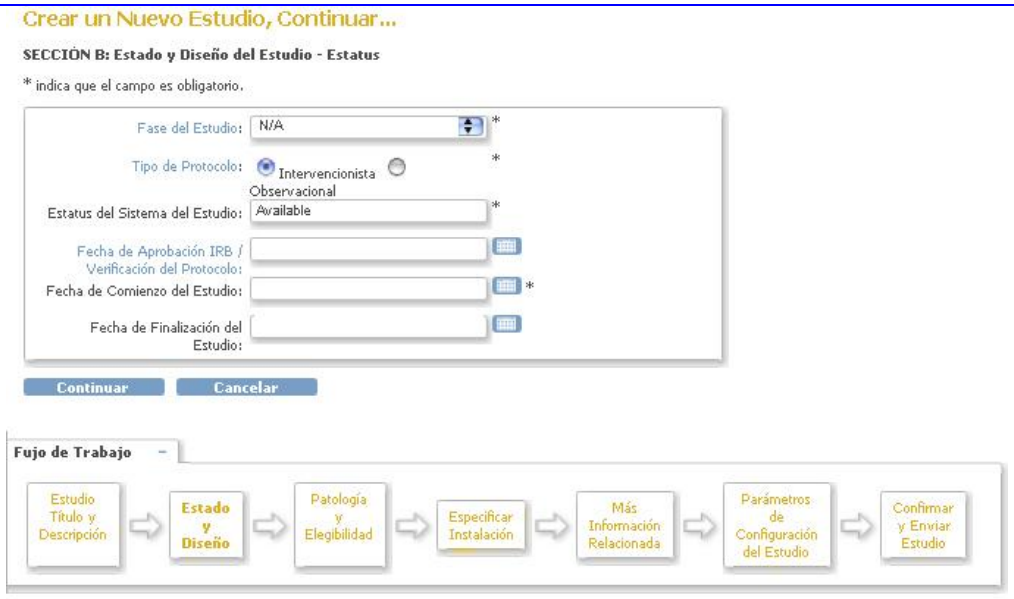

Figura 58 Prototipo de interfaz Crear Estudio

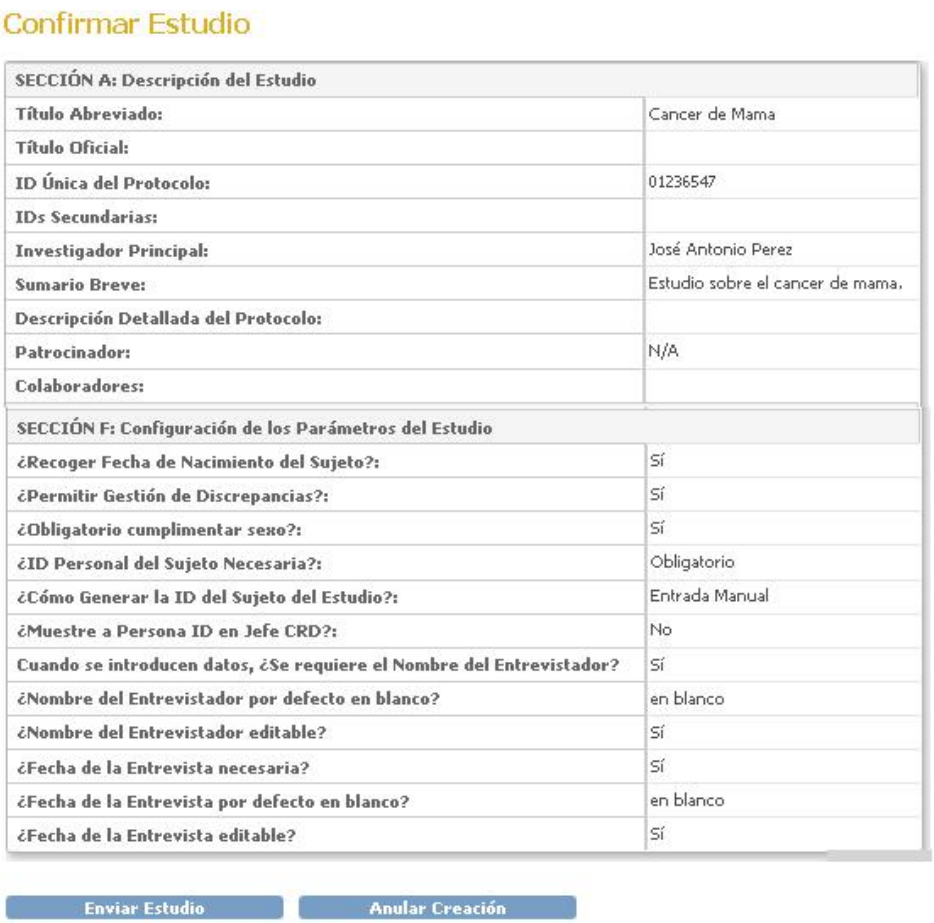

Figura 59 Prototipo de interfaz Pantalla de confirmación Crear Estudio

Poscondiciones	Se adiciona un nuevo estudio al sistema. Adquiere un nuevo estado: “Diseño”.

Caso de uso Insertar Usuario

Descripción de Casos de Uso Insertar Usuario

Caso de Uso:	Insertar Usuario
Actores:	Administrador (Inicia)
Resumen:	El caso de uso inicia cuando el administrador selecciona Usuarios en el módulo Administrar Empresa. El sistema muestra un listado con los usuarios disponibles y las acciones correspondientes. El administrador indica Crear nuevo usuario, y llena los datos. El caso de uso termina si el administrador selecciona otra opción del sistema.
Precondiciones:	El usuario debe estar registrado como Administrador en el sistema.
Referencias	RF 12,RF 13
Prioridad	3

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Administrador selecciona la opción Usuarios en el módulo Administrar Empresa.	1.1 El sistema muestra un listado con los usuarios disponibles, información asociada a los mismos y las acciones que se pueden realizar sobre estos.
2. El Administrador escoge Crear nuevo usuario.	2.1 El sistema muestra los datos que deben ser llenados, indicando aquellos campos que son obligatorios.
3. El Administrador llena los datos para el nuevo usuario asignándole un rol determinado dentro de un estudio o	

centro específico. 4. El Administrador indica enviar.	4.1 El sistema valida los datos y si la información es correcta muestra los datos de la cuenta de usuario recién creada, dándole la posibilidad al administrador de: • Editar cuenta de usuario • Crear nuevo usuario • Salir
5. Si el administrador escoge una de las opciones siguientes: • Editar cuenta de usuario. (Operación Gestionada por OpenClínica). • Crear nuevo usuario: Ir al paso 2.1 del flujo normal de eventos. • Salir: Ir al paso 1.1 del flujo normal de eventos.	
Flujos Alternos	
Flujo Alterno de eventos acción 2.	
Acción del Actor	Respuesta del Sistema
2.a El caso de uso termina si el administrador escoge otra opción disponible en el sistema.	
Flujo Alterno de eventos acción 4.	
Acción del Actor	Respuesta del Sistema
4.a El administrador indica la opción cancelar.	4.b El sistema redirecciona al administrador al listado de usuarios. Ir a paso 1.1 del flujo normal de eventos.
Flujo Alterno de eventos acción 4.1	
Acción del Actor	Respuesta del Sistema
	4.1a Si el sistema al validar los datos encuentra errores, indica el campo donde se cometió el mismo con un mensaje

	sugerente sobre cómo solucionarlo.
4.1b El administrador corrige los datos e indica enviar.	4.1c Ir al paso 4.1 del flujo normal de eventos.

Prototipo de Interfaz

Figura 60 Prototipo de interfaz Administrar Usuario

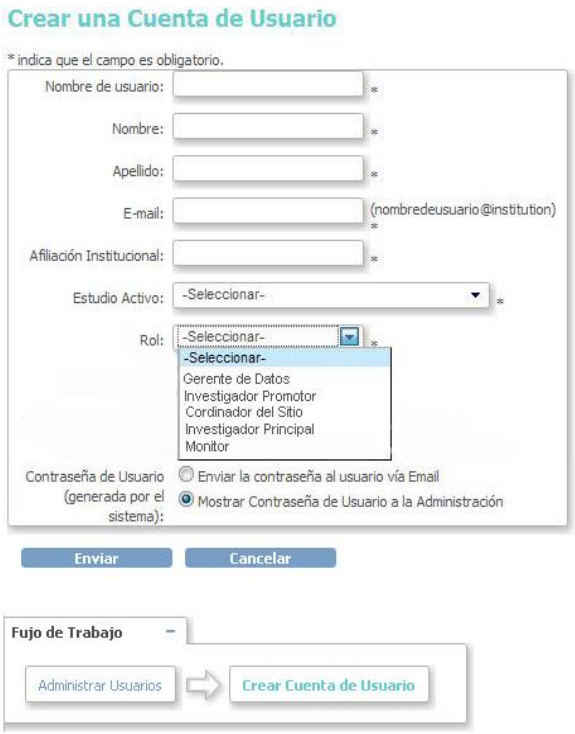

Figura 61 Prototipo de interfaz Insertar Usuario

Figura 62 Prototipo de interfaz Confirmación de creación de usuario

Poscondiciones	Se adiciona un usuario al sistema, asociando este a un estudio determinado con un rol específico.

Caso de uso Gestionar Validación IP

Descripción de Casos de Uso Gestionar Validación IP

Caso de Uso:	Gestionar Validación IP
Actores:	
Resumen:	El caso de uso es incluido por los casos de uso Gestionar IP Centro, Gestionar IP Estudio y Gestionar IP Usuario. El mismo es invocado cuando alguno de los mencionados es invocado. El sistema valida los IPs y/o rangos de IPs que se adicionan en dependencia del caso de uso que lo invoque indicando si hay errores en los datos mediante un mensaje sugerente para corregir el error. El caso de uso termina cuando los datos son validados.

Precondiciones:	Alguno de los casos de uso Gestionar IP Centro, Gestionar IP Estudio y Gestionar IP Usuario es invocado.
Referencias	RF 19
Prioridad	4
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
	1. Si la dirección IP y/o el rango de IPs no es válido el sistema muestra un mensaje de error indicando la corrección de los datos. 2. Si el administrador introduce una dirección IP y/o un rango que ya está asignada, el sistema muestra un mensaje de error indicando que la información no puede ser duplicada. 3. Si el administrador introduce un rango que tienen intersección con otro ya existente el sistema muestra un mensaje de error indicando que dentro del nuevo rango existen direcciones que ya están asignadas.
Poscondiciones	La gestión de IP es validada.

Diagrama de Clases del diseño. Caso de Uso Gestionar IP Estudio/Centro

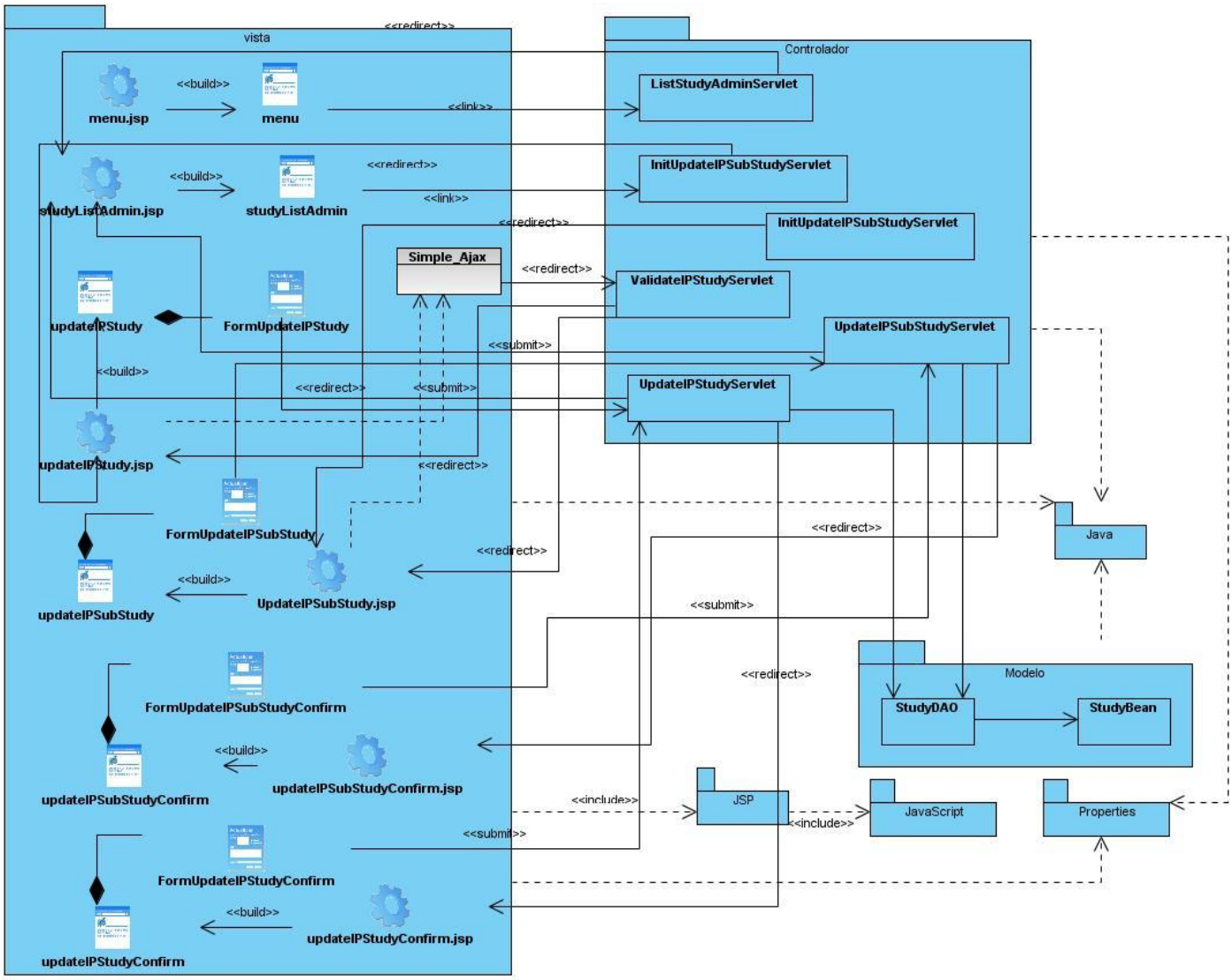

Figura 63 Diagrama de Clases del diseño. Caso de Uso Gestionar IP Estudio/Centro

Diagrama de Clases del diseño. Caso de Uso Gestionar IP Insertar Estudio

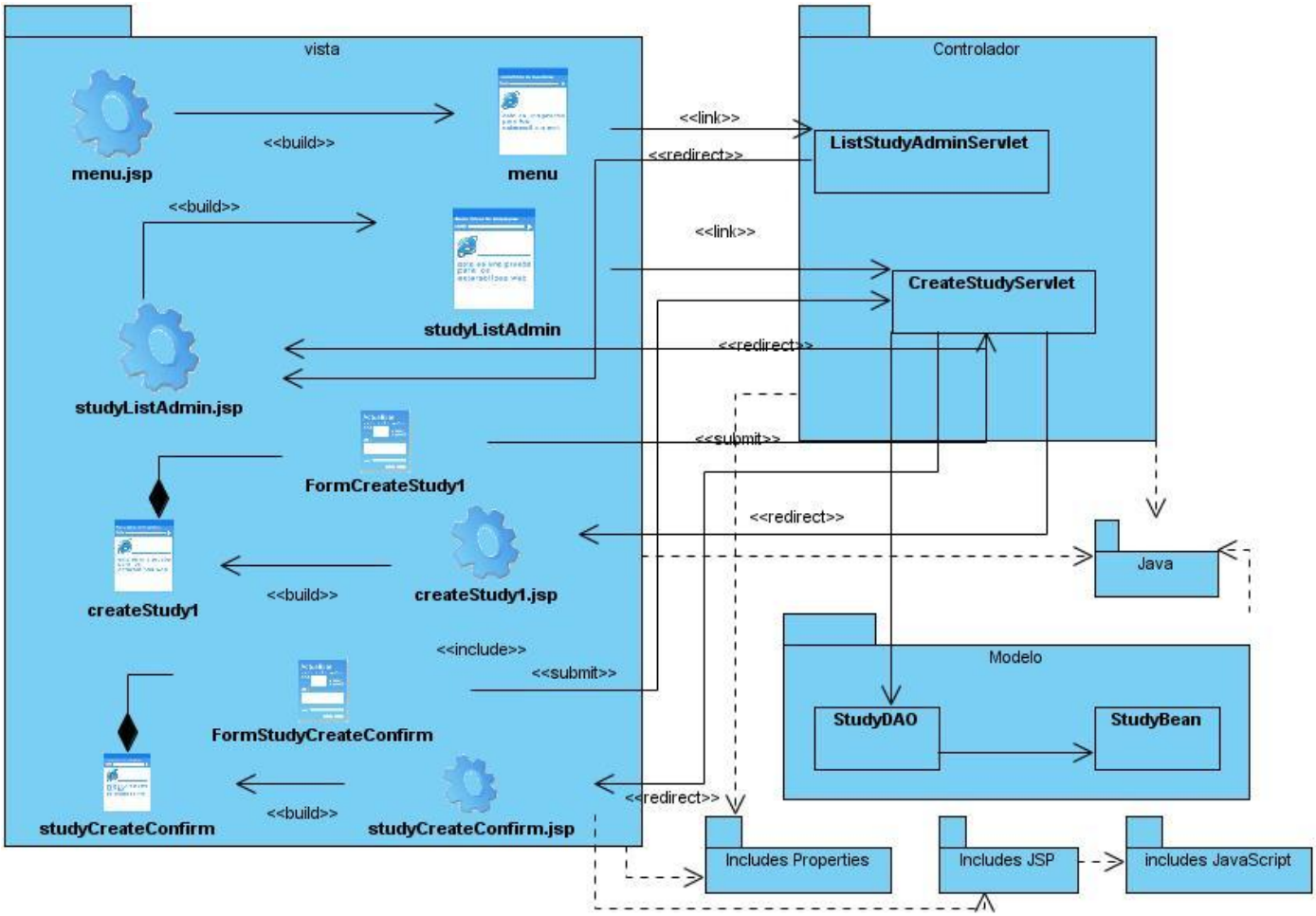

Figura 64 Diagrama de Clases del diseño. Caso de Uso Gestionar IP Insertar Estudio

Diagrama de Clases del diseño. Caso de Uso Gestionar IP Insertar Usuario

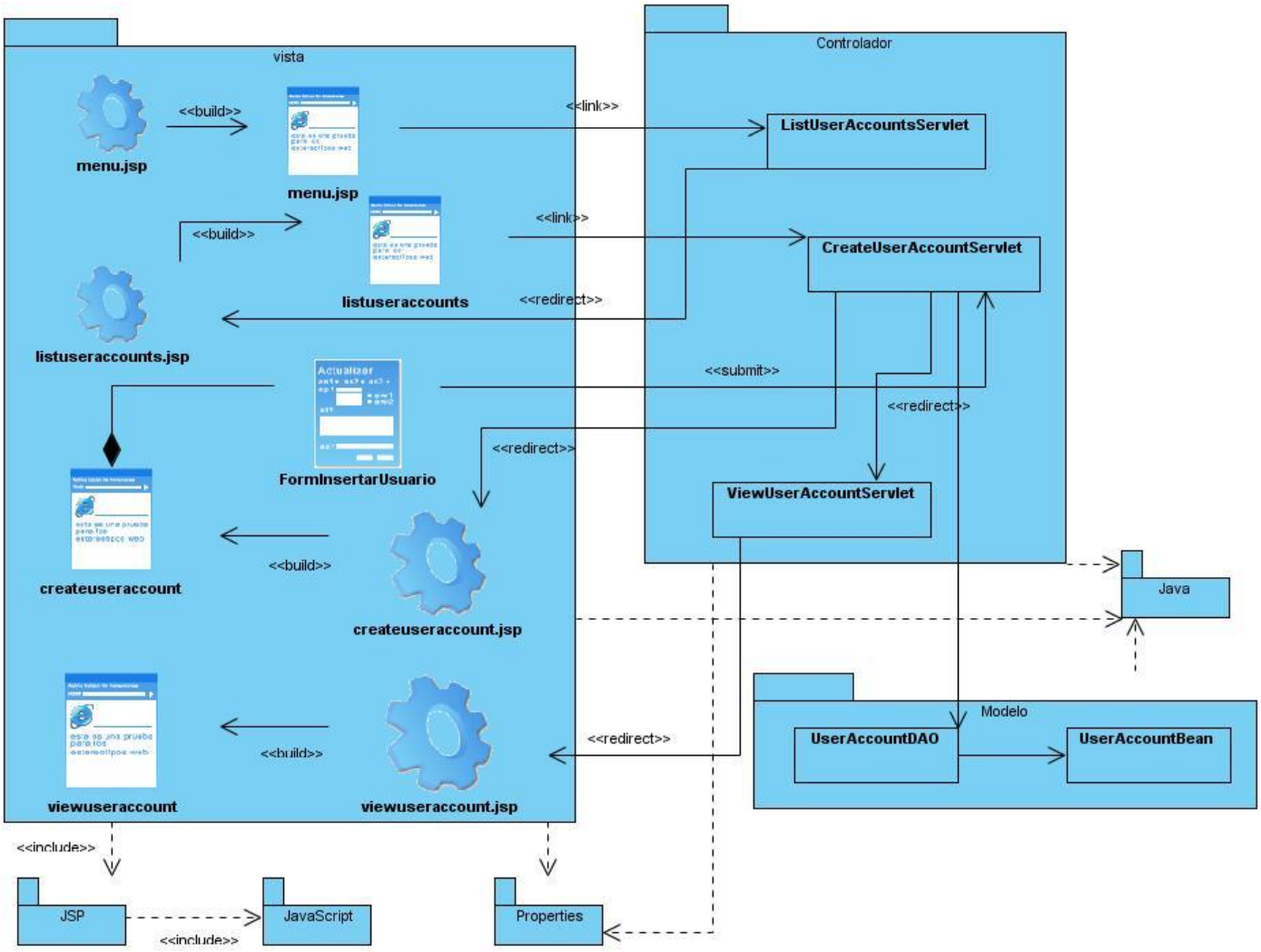

Figura 65 Diagrama de Clases del diseño. Caso de Uso Gestionar IP Insertar Usuario

Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario “Adicionar Estudio”

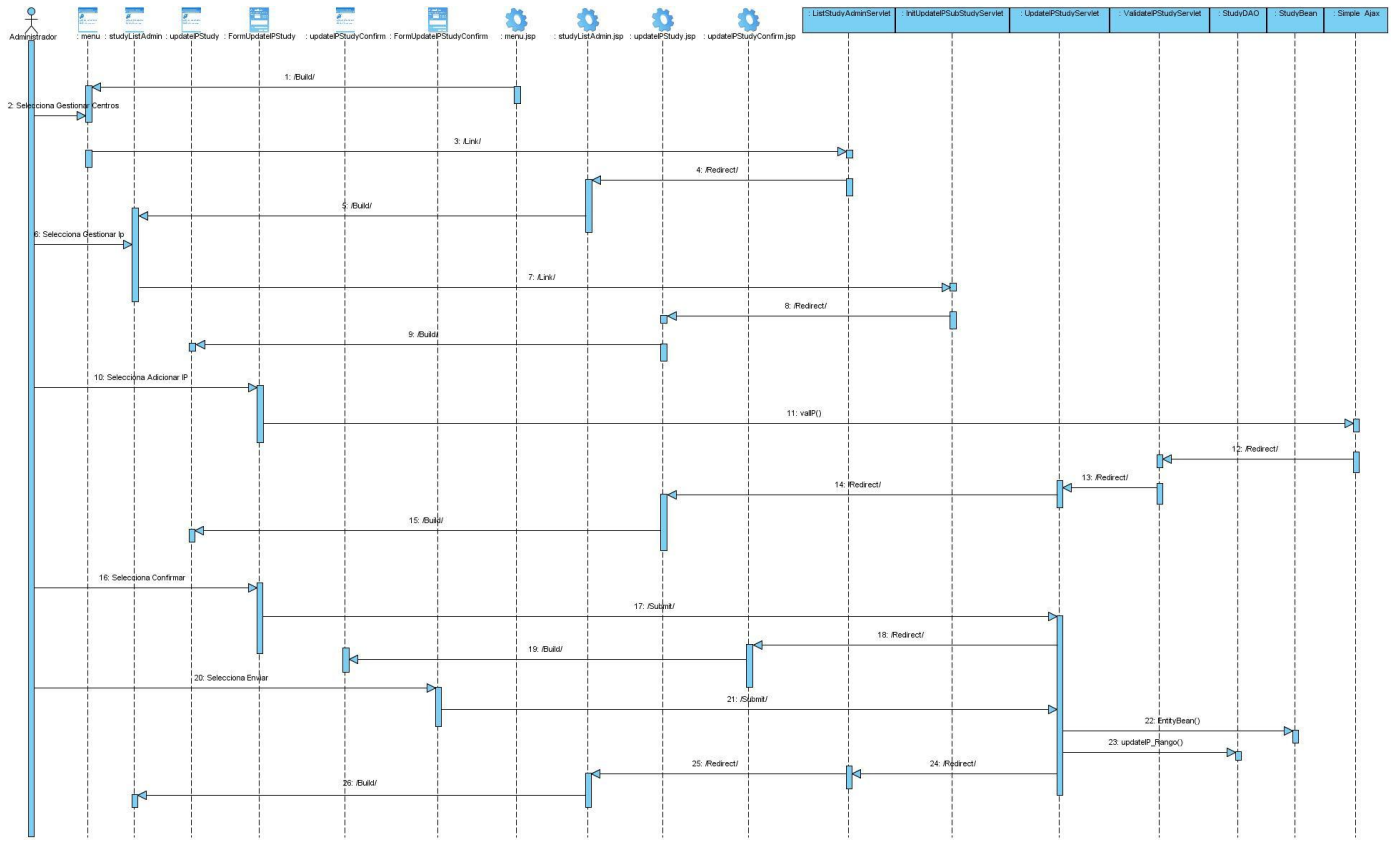

Figura 66 Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Adicionar Estudio"

Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Eliminar Estudio"

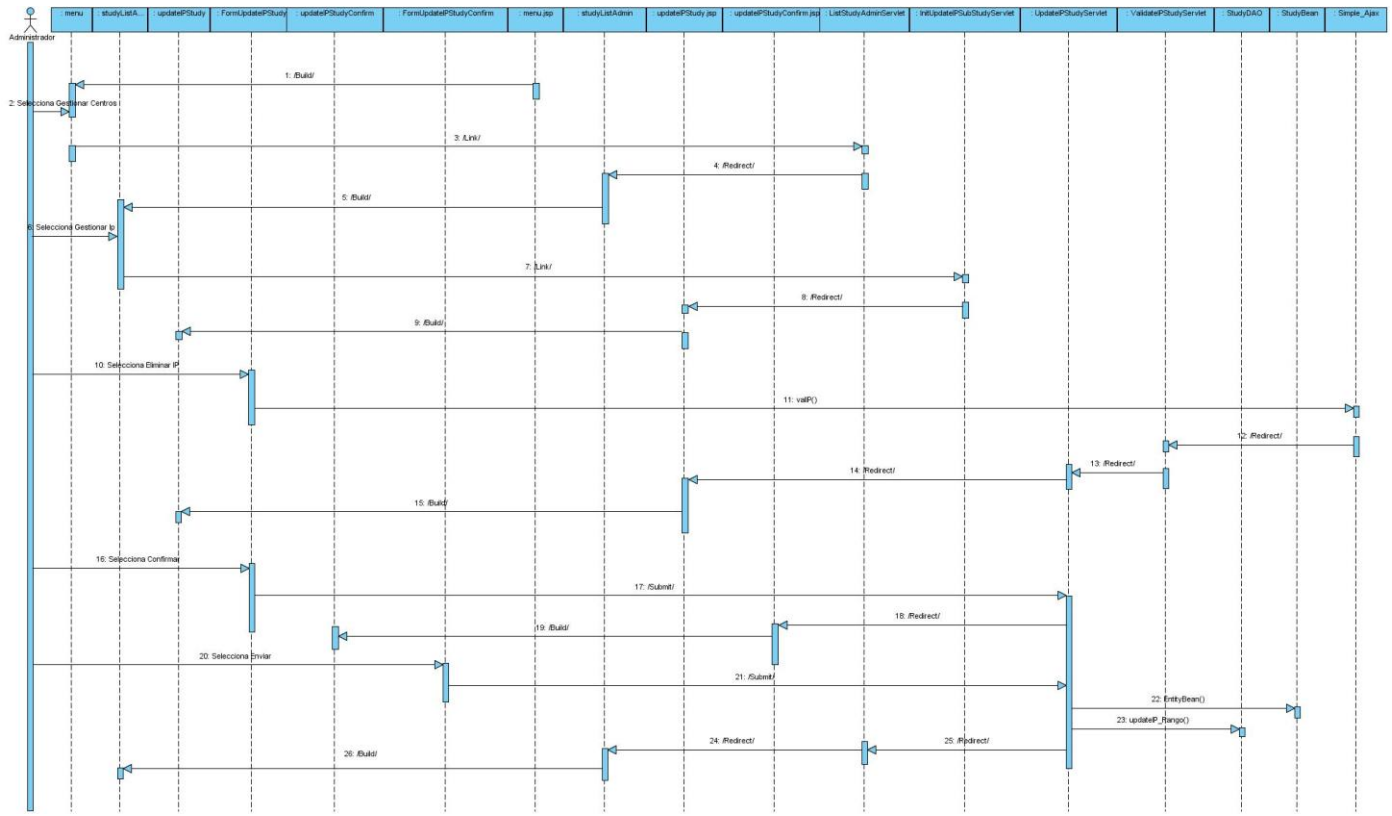

Figura 67 Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Eliminar Estudio"

Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Adicionar Centro"

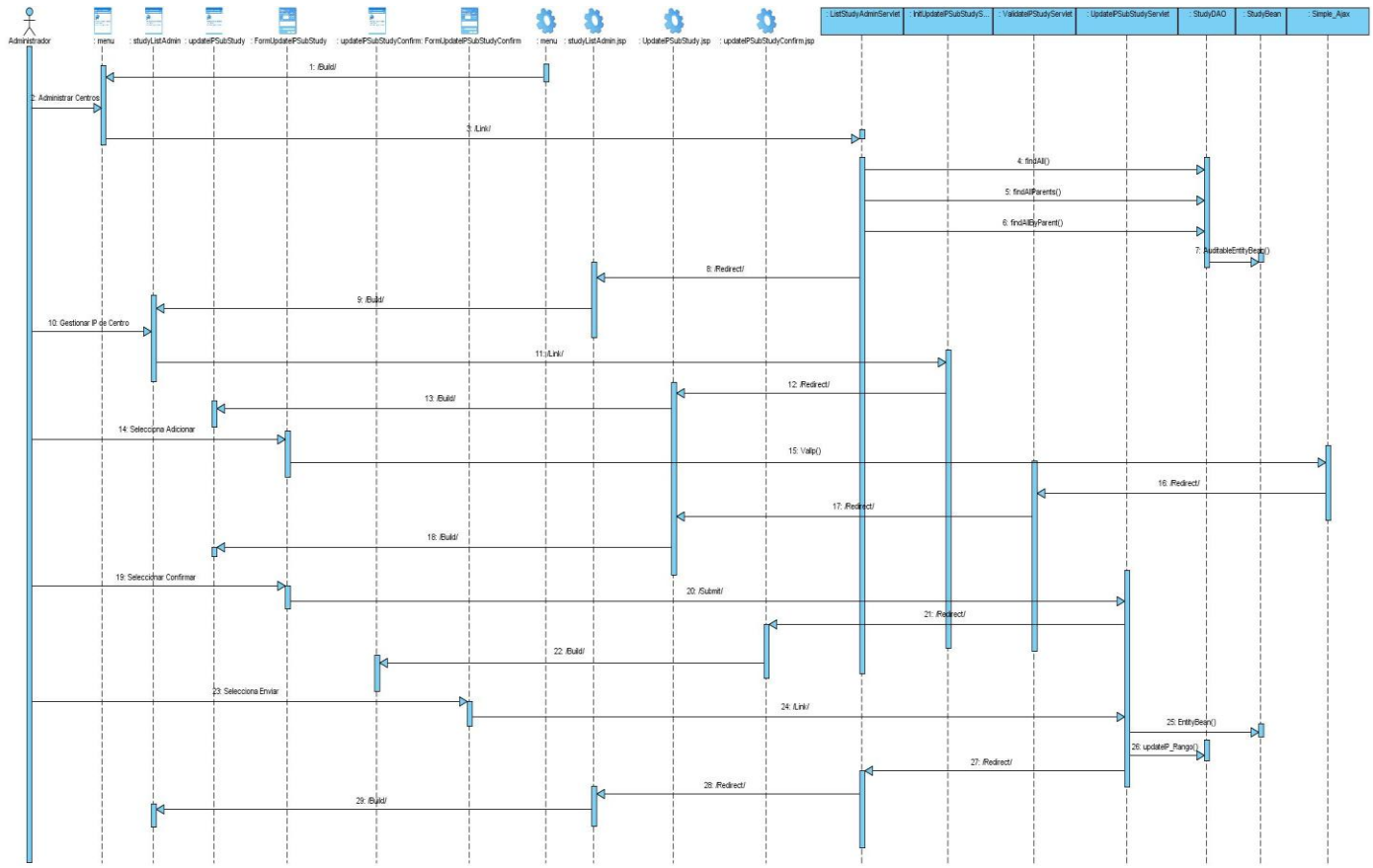

Figura 68 Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Adicionar Centro"

Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Eliminar Centro"

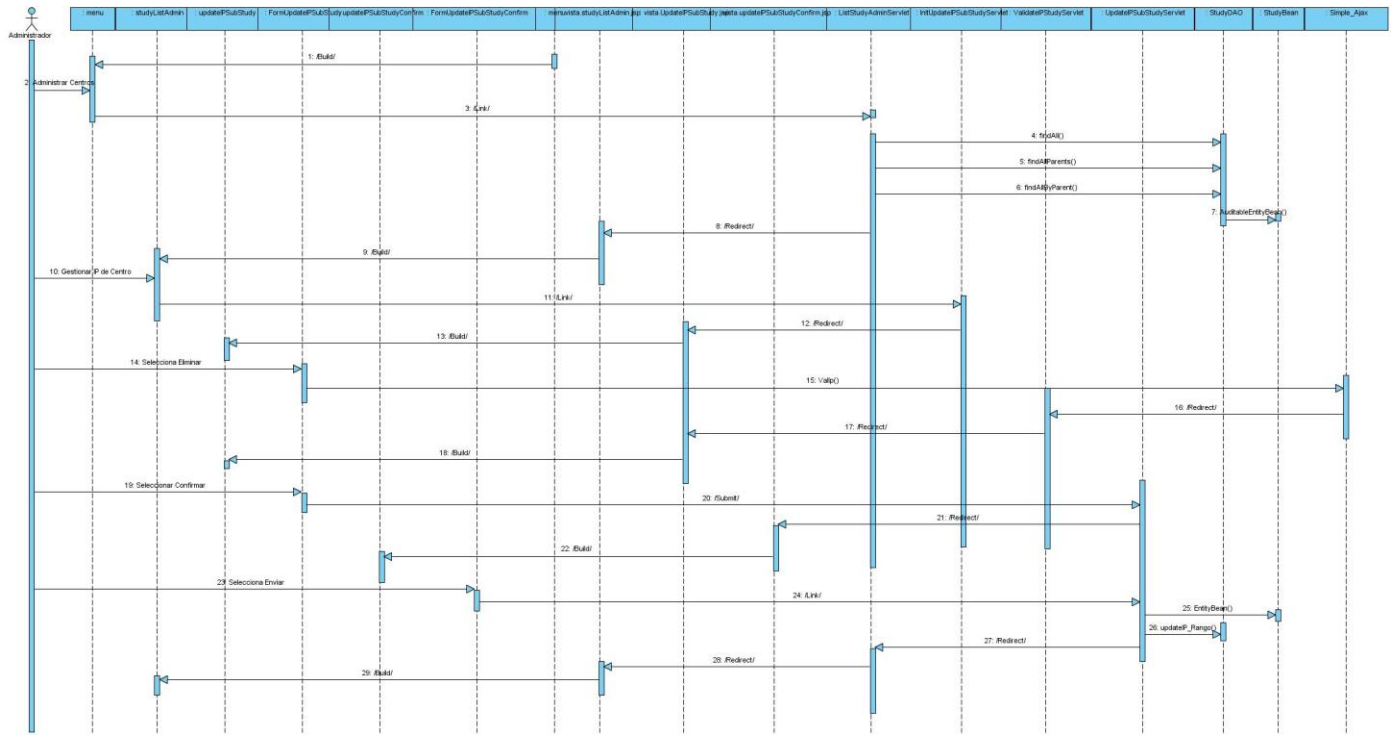

Figura 69 Diagrama de secuencia Caso de Uso Gestionar IP Estudio/Centro. Escenario "Eliminar Centro"

Diagrama de secuencia Caso de Uso Insertar Estudio

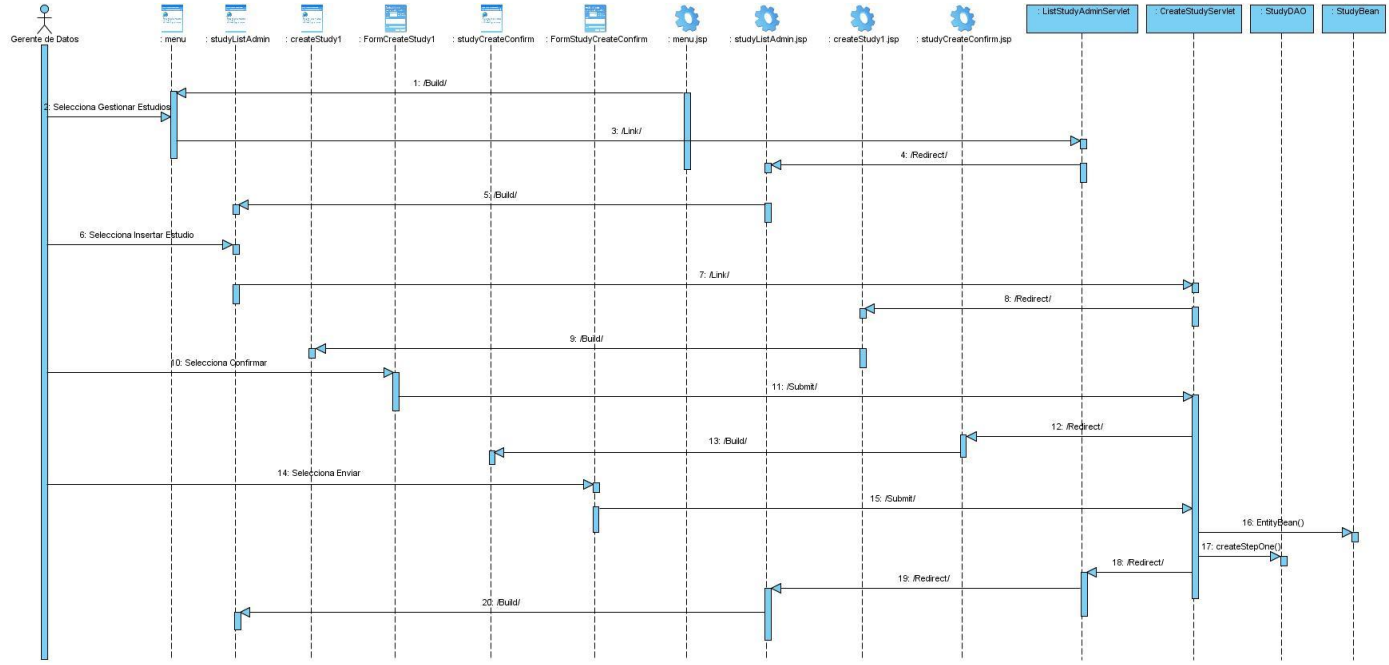

Figura 70 Diagrama de secuencia Caso de Uso Insertar Estudio

Diagrama de secuencia Caso de Uso Insertar Usuario

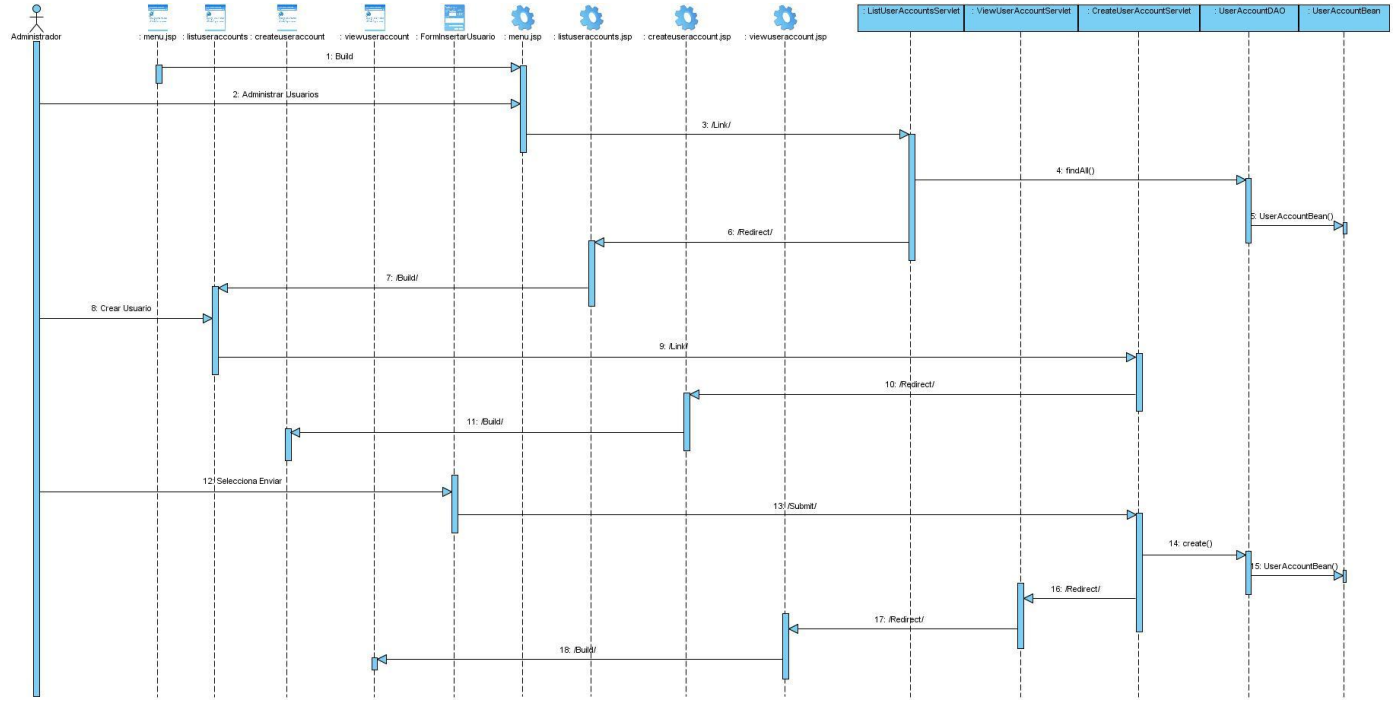

Figura 71 Diagrama de secuencia Caso de Uso Insertar Usuario

Diagrama de Componentes. Caso de Uso Gestionar IP Estudio/Centro

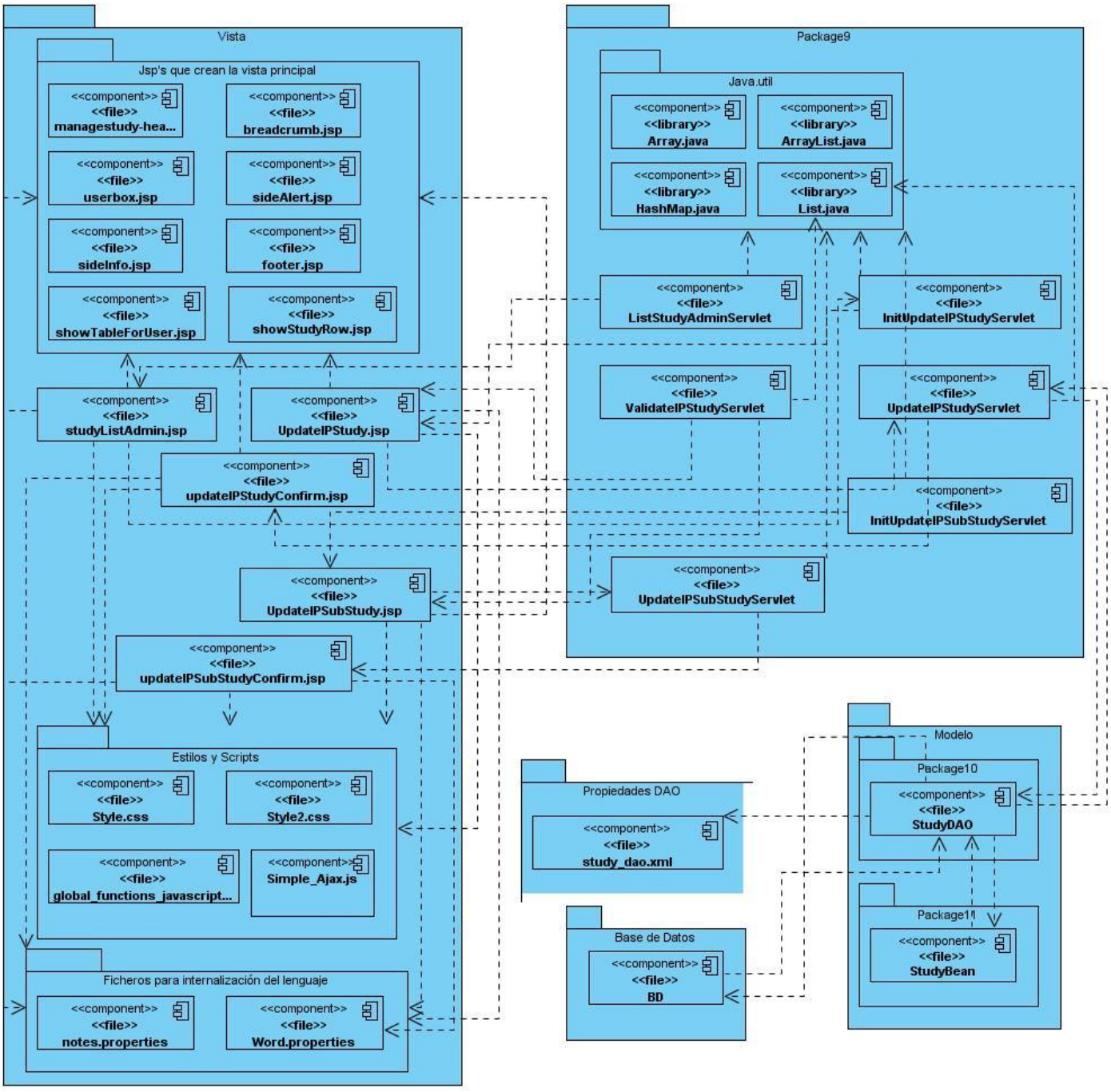

Figura 72 Diagrama de Componentes. Caso de Uso Gestionar IP Estudio/Centro

Diagrama de Componentes. Caso de Uso Insertar Estudio

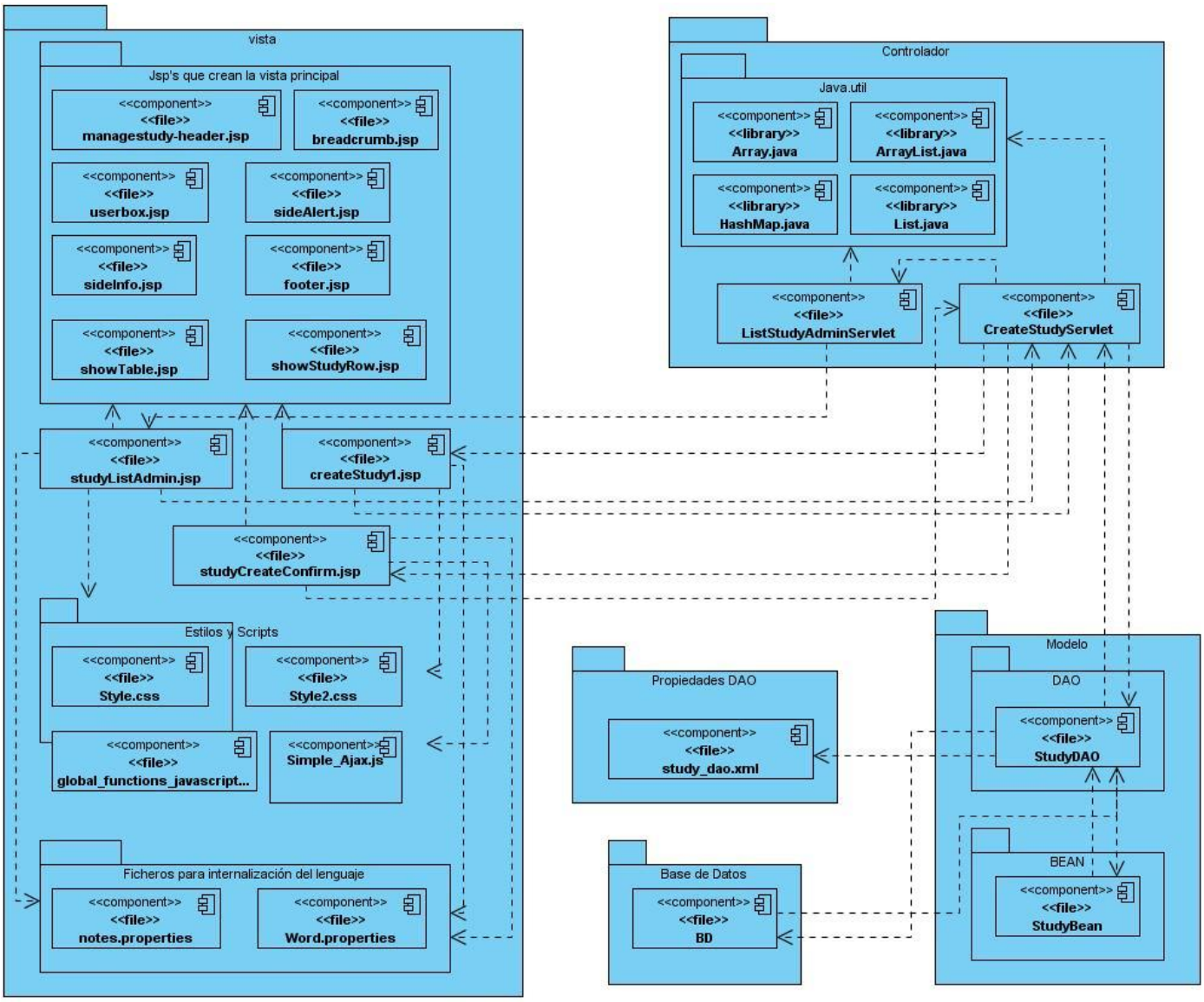

Figura 73 Diagrama de Componentes. Caso de Uso Insertar Estudio

Diagrama de Componentes. Caso de Uso Insertar Usuario

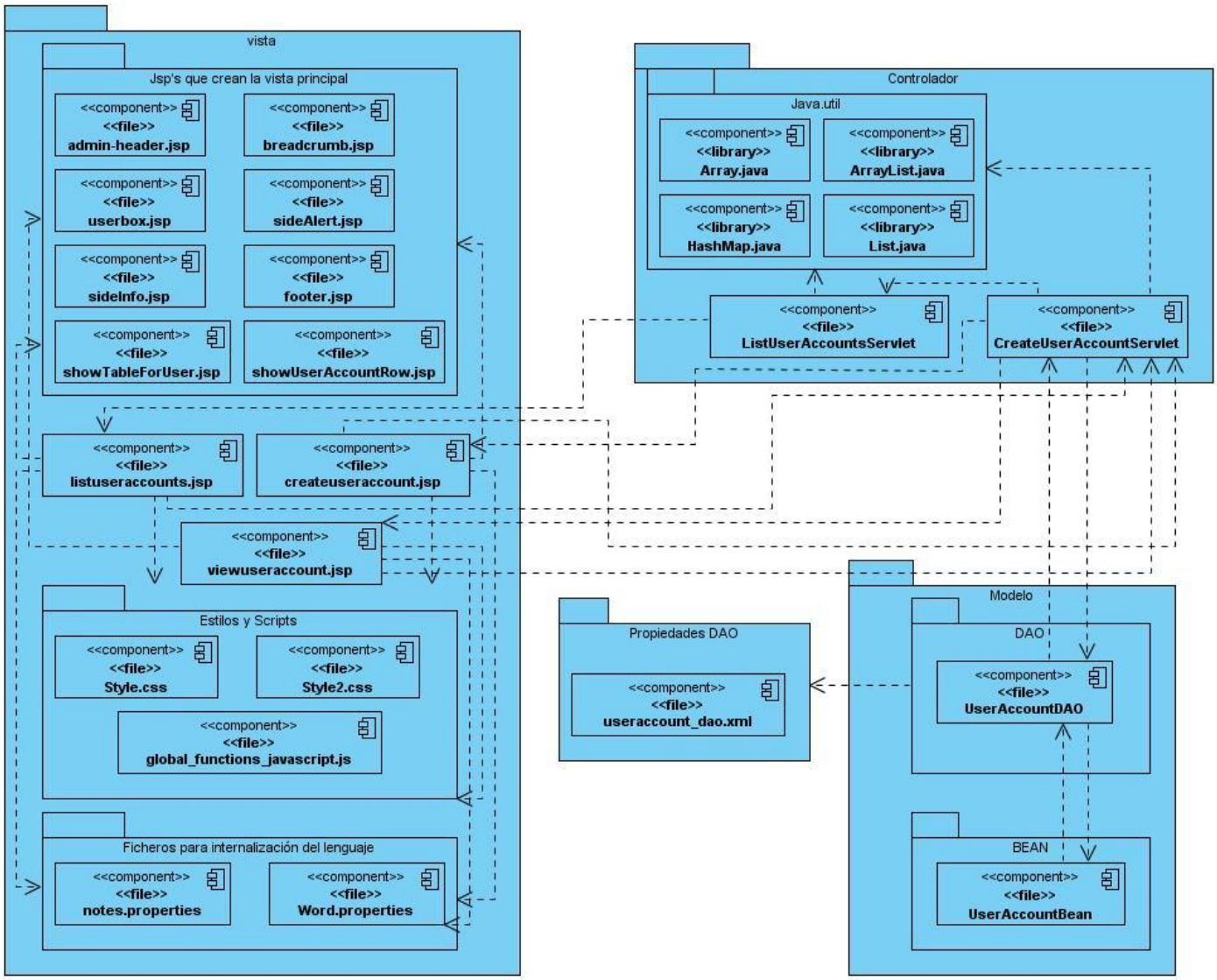

Figura 74 Diagrama de Componentes. Caso de Uso Insertar Usuario

GLOSARIO

A

API: Interfaz de Programación de Aplicaciones (Application Programming Interface), es un conjunto de funciones residentes en bibliotecas generalmente dinámicas, que permiten que una aplicación corra bajo un determinado sistema operativo.

B

Bytecode*:* Código intermedio entre el código fuente y el código final.

C

Cuaderno de Recogida de Datos: El Cuaderno de Recogida de Datos (CRD) ó Case Report Form (CRF) por sus siglas en inglés, es un formulario, con espacios definidos, diseñado para anotar las variables recogidas durante un ensayo clínico.

CRF: Case Report Form por sus siglas en inglés, traducción al inglés de Cuaderno de Recogida de Datos.

E

Ensayos Clínicos: Los ensayos clínicos son una parte fundamental en el proceso de desarrollo, aprobación e introducción en el mercado de nuevos fármacos y tratamientos contra el cáncer.

EJB: Los Enterprise JavaBeans son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE. Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor Los EJB proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.).

F

Framework: Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

G

GRSP: General Responsibility Assignment Software Patterns: Patrones de Software para la asignación General de Responsabilidad.

H

HTML: HyperText Markup Language (Lenguaje de Marcas de Hipertexto).

Hibernate: es un proyecto de código abierto que sirve como marco de trabajo para la persistencia en Java.

HTTP: HyperText Transfer Protocol. Protocolo de transferencia de hipertexto. Usado en cada transacción de la Web.

HTTPS: Hypertext Transfer Protocol Secure. Protocolo seguro de transferencia de hipertexto. Destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

J

JDBC: es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

JDO: Java Data Objects, objetos de datos Java son una especificación del objeto de persistencia Java. Una de sus características es la transparencia de los servicios persistentes del modelo de dominio.

JSTL: Java Server Pages Standard Tag Library.

M

Metástasis: La metástasis es la diseminación a órganos distantes de una infección o de un tumor primario maligno o cáncer, que ocurre generalmente por vía sanguínea o linfática.

N

Neoplasias: Proceso de proliferación anormal de células en un tejido u órgano que desemboca en la formación de un neoplasma.

Neoplasma: Forma una masa diferenciada se denomina tumo.

O

OpenSource: Se refiere al acceso al código del software.

P

Protocolo: documento que describe los objetivos, el diseño, la metodología, las consideraciones estadísticas y la organización de un ensayo clínico. Contiene un plan de estudio sobre el cual el ensayo está basado. Rige la investigación se explican todos los pasos a seguir.

S

SSL: Secure Sockets Layer. Protocolo de Capa de Conexión Segura. Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente.

SUN: Las siglas SUN se derivan de «Stanford University Network», proyecto que se había creado para interconectar en red las bibliotecas de la Universidad de Stanford. Sun Microsystems es una empresa informática de Silicon Valley, fabricante de semiconductores y software.

T

Toplink: Herramienta para abstraer la capa de datos en una aplicación java. Provee un marco de trabajo poderoso y flexible para el almacenamiento de objetos Java en una base de datos relacional ó para convertir objetos de la misma naturaleza a archivos XML.

Triggers*:* Son procedimientos almacenados que se ejecutan cuando se realiza alguna operación sobre la tabla al cual está asociado. Esto son disparadores automáticos.