Universidad de las Ciencias Informáticas

Facultad 2

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

TÍTULO: "DESARROLLO DEL MÓDULO EXTRACCIÓN PARA EL SISTEMA DE GESTIÓN DE ENSAYOS CLÍNICOS"

Autores:

Gleidys Gil García

Juan Miguel Lorenzo León

Tutores:

Ing. Lisette Bravo León

Ing. Yasmany Núñez Broch

La Habana, 2014

"Año 56 de la Revolución"

Pensamiento

"Son los problemas sin resolver, no los resueltos, los que mantienen activa la mente".

Erwin Guido Kolbenheyer

Datos de contacto

Ing. Lisette Bravo León: Graduado de Ingeniería en Ciencias Informáticas en el año 2011 en la Universidad de la Ciencias Informáticas y actualmente pertenece al departamento Ingeniería de Software de la facultad 2.

Correo electrónico: **lbleon@uci.cu**

Ing. Yasmany Núñez Broch: Graduado de Ingeniería en Ciencias Informáticas en el año 2012 en la Universidad de las Ciencias Informáticas y actualmente desempeña el rol de desarrollador en el proyecto Sistema de Gestión de Ensayos Clínicos.

Correo electrónico: **ynbroch@uci.cu**

Agradecimientos

De Gleidys

A mis abuelos, Zoraida (abuela vida) y René (abuelo niño) por ser las personas más especiales en mi vida y por ayudarme siempre en todo momento brindándome su amor y cariño. A mi hermana Ariamna que quiero mucho, y siempre me ha apoyado y aconsejado en todo momento a pesar de la distancia. A mis padres, Nieves María y Armando por haberme dado una buena educación, por el apoyo incondicional que siempre me han brindado y por confiar en mí. A mi adorado novio, Gerardo por ser tan comprensible conmigo, y a sus padres y abuelos por ser tan atentos conmigo. A mi cuñado, Ismel por ayudarme a estudiar cada vez que lo necesité, y por cuidar muy bien de nosotros. A toda mi familia en general por ser personas tan especiales y buenas en mi vida. A mi compañero de tesis Juan, que juntos hemos trabajado para lograr nuestro sueño. A mis tutores, trabajadores del proyecto y al tribunal de los seminarios de tesis que nos apoyaron bastante en el desarrollo del trabajo, en especial a Claudia. A mis compañeros de aula y de apartamento por hacerme la vida mas amena en la universidad.

De Juan Miguel

Agradezco a mis amigos, compañeros y profesores de la UCI por haberme ayudado a crecer como ser humano y hombre. También a mi compañera de tesis por recordarme la frase "En la unión esta la fuerza". A mi tutores por el apoyo brindado.

Dedicatoria

De Gleidys

Le dedico mi tesis a mis abuelos que adoro mucho, a mi hermana que la quiero con el alma, a mis padres por quererme tanto y yo a ellos, a mi novio por ser la persona más especial que he conocido.

De Juan Miguel

Dedico este trabajo y los cinco años de la carrera a mis padres por constituir para mi ejemplos de amor y sacrificio. A mi hermana Rosali porque es y será lo mas bello que mis padres me han regalado en este mundo. Al resto de mi familia por estar siempre a mi lado cuando los necesite.

Resumen

El Centro de Inmunología Molecular (CIM) es una institución biotecnológica cubana dedicada a la investigación, fabricación y la comercialización de productos biofarmacéuticos, en el mercado nacional e internacional. A fin de cumplir las necesidades funcionales de este centro, ya que el sistema Clínicas 1.0 que utilizan los especialistas para efectuar los ensayos clínicos presenta algunos inconvenientes, se encuentra en desarrollo una nueva versión denominada Sistema de Gestión de Ensayos Clínicos (SIGEC). El presente trabajo de diploma está centrado en la elaboración del módulo Extracción de dicho sistema, con el objetivo de mejorar la gestión de conjuntos de datos y trazas del mismo.

Para el desarrollo de las funcionalidades se han empleado diversas herramientas y tecnologías: Eclipse 3.4.2 como entorno de desarrollo integrado, Java como lenguaje de programación, PgAdmin III para la administración del gestor de base de datos PostgreSQL 8.4. La herramienta de análisis y diseño utilizada, Visual Paradigm 8.0 apoyándose en el lenguaje de modelado UML 2.1, y la metodología de desarrollo de software, Proceso Unificado de Desarrollo (RUP). Finalmente como patrón de arquitectura de software, Modelo Vista Controlador (MVC).

Las funcionalidades propuestas permiten garantizar de forma segura y eficiente la gestión de conjuntos de datos y trazas en el SIGEC, facilitando este trabajo a los especialistas del CIM y de esta forma contribuir a la producción de fármacos en Cuba para combatir disímiles enfermedades.

Palabras claves: *Clínicas 1.0, conjuntos de datos, extracción, herramientas, Sistema de Gestión de Ensayos Clínicos (SIGEC), traza.*

Índice

Índice de Figuras

Índice de Tablas

Introducción

Desde tiempos remotos de la historia humana existe el uso de diversas sustancias químicas desde el punto de vista terapéutico, muy pocas de ellas tenían realmente la acción deseada, como no fuera la derivada del efecto placebo[1] originado por el consumo de tales preparados. A finales del siglo XIX y principios del siglo XX la manufactura de fármacos se convirtió en un negocio de gran importancia económica. Esta importancia se derivó básicamente de la producción en masa y del acceso de grupos poblacionales cada vez mayores; apareciendo un mercado que se hizo cada vez más exigente con relación al producto buscado; en otras palabras, se comenzó a requerir una mayor eficacia para los fármacos, conjuntamente con una mayor seguridad en su uso. (1)

Así pues, desde la década de los cincuenta, se fue instituyendo a nivel mundial la obligación de comprobar, tanto la eficacia como la seguridad de las drogas antes de su comercialización, surgiendo así uno de los fenómenos más importantes en la medicina y en la investigación clínica: el ensayo clínico controlado, como metodología portadora de la evidencia necesaria para la toma de decisiones en la práctica médica, para evaluar diferentes procederes terapéuticos. (2)

Un ensayo clínico es un estudio que permite a los médicos determinar si un nuevo tratamiento, medicamento o dispositivo contribuirá a prevenir, detectar o tratar una enfermedad. Ayudan a descubrir si estos nuevos tratamientos son inocuos[2] y si son mejores que los actuales (3), comprobando la eficacia de dichos medicamentos mediante la realización de pruebas con pacientes reales, los cuales son monitoreados a lo largo de todo el proceso de estudio.

Cuba cuenta con diferentes centros promotores de ensayos clínicos entre los cuales se encuentra el Centro Nacional Coordinador de Ensayos Clínicos (CENCEC) que tiene como objetivo realizar la evaluación clínica a productos de la Industria Biotecnológica, Médico Farmacéutica y de Equipos Médicos, así como a nuevas tecnologías a solicitud del Sistema Nacional de Salud (4). El Centro de Ingeniería Genética y Biotecnología de Cuba (CIGB) que es una institución de desarrollo dinámico que le ha permitido alcanzar un alto nivel en la investigación, desarrollo, producción y comercialización de productos

[1] Placebo: sustancia que carece de acción curativa pero produce un efecto terapéutico si el enfermo la toma convencido de que es un medicamento realmente eficaz.
[2] Inocuo: que no hace daño físico o moral.

biológicos, obtenidos a través de los métodos de la biotecnología moderna (5), integrando investigación-producción aplicada a diferentes ramas de la sociedad como la medicina. También el país cuenta con el Centro de Inmunología Molecular (en lo adelante CIM) que es una institución biotecnológica dedicada a la investigación, desarrollo, fabricación y la comercialización de productos biofarmacéuticos, en el mercado nacional e internacional, para el diagnóstico y tratamiento del cáncer y otras enfermedades relacionadas con el sistema inmune (6); además cuenta con un Registro Público Cubano de Ensayos Clínicos (RPCEC) que consiste en una base de datos de ensayos clínicos en la cual cualquier promotor de ensayo clínico, nacional o extranjero, puede inscribir su ensayo de forma gratuita, siendo accesible a través de la web cumpliendo con todos los requisitos establecidos por la Plataforma de Registros Internacionales de Ensayos Clínicos (ICTRP) de la Organización Mundial de la Salud (OMS). (7)

Como resultado de la gran cantidad de información que se genera y fluye constantemente en cada uno de los procesos clínicos y administrativos de los centros antes mencionados, entran a jugar un papel fundamental las Tecnologías de la Información y las Comunicaciones (en lo adelante TIC). Estas han permitido el surgimiento de sistemas informáticos para gestionar de manera eficiente el gran cúmulo de información que se genera en un estudio, viabilizando dichos procesos.

En el citado CIM se encuentra instalado el sistema Clínicas 1.0, que permite la gestión de los ensayos clínicos que ahí se realizan. Este sistema fue desarrollado en el Centro de Informática Médica de la Facultad 2, perteneciente a la Universidad de las Ciencias Informáticas (en lo adelante UCI), que es hoy en día una de las principales instituciones desarrolladoras de software del país.

El sistema Clínicas 1.0 está formado por cuatro módulos Administrar Sistema, Gestionar Estudio, Enviar Datos y Extraer Datos. Este último permite crear y visualizar conjuntos de datos a partir de la información registrada de los sujetos en el ensayo; donde la salida de estos reportes puede ser en diversos formatos de uso extendido: SPSS, ODM XML, XLS y HTML. Un conjunto de datos se puede crear de forma dinámica indicando a partir de qué momento de seguimiento (en lo adelante MS), hoja CRD[3] y/o variable desea conformarlo el especialista. Además se puede obtener información sobre algunas de las acciones realizadas por los usuarios en el sistema a través de la visualización de trazas logrando imprimir y exportar las mismas. Las trazas se pueden buscar seleccionado los criterios “variables de las hojas CRD”

[3] CRD: Cuaderno de Recogida de Datos en el cual se recoge los valores que toman los indicadores o variables a medir de los pacientes del estudio, mediante el llenado de las hojas asociadas a estos.

o por “acciones generales” que realiza determinado usuario en el módulo Enviar Datos sobre los elementos del sistema.

El módulo Extraer Datos, aunque facilita las operaciones mencionadas anteriormente, no posibilita determinar cuáles fueron los usuarios que intervinieron en el diseño de los estudios pues solamente se muestran las trazas generadas en el módulo Enviar Datos, por lo que en caso de que se pierda alguna información a través del mismo, se corre el riesgo de no poder saber de qué usuario fue la responsabilidad comprometiendo la confidencialidad, integridad y disponibilidad de la información. También este módulo permite extraer datos clínicos a partir de la gestión de conjuntos de datos. Estos conjuntos de datos se pueden crear de forma dinámica para todos los sujetos de un estudio o de un centro, pero se hace necesario poder obtener esta información para un grupo determinado de sujetos, adquiriéndose un mejor agrupamiento u organización de los datos y una mayor flexibilidad en la adquisición de la información, facilitándole a los especialistas el posterior análisis con los mismos.

De acuerdo con la situación descrita anteriormente, se define como **problema a resolver** la siguiente interrogante: ¿Cómo mejorar la gestión de la información en el proceso de extracción de datos de ensayos clínicos del sistema Clínicas? Este enmarca como **objeto de estudio** al proceso de extracción de datos en los ensayos clínicos, quedando especificado como **campo de acción** la gestión de la información en la extracción de datos en los ensayos clínicos.

El **objetivo** de la investigación es desarrollar en el Sistema de Gestión de Ensayos Clínicos (en lo adelante SIGEC) el módulo Extracción permitiendo mejorar la gestión de la información en el proceso de extracción de datos.

Las **tareas de la investigación** planteadas para lograr el objetivo propuesto son las siguientes:

1. Analizar los sistemas que gestionan conjuntos de datos y las trazas en la extracción de datos de ensayos clínicos, a nivel internacional y nacional, estableciendo similitudes con la investigación en curso.
2. Asimilar las herramientas, tecnologías y metodología, definidas en el departamento Sistemas Especializados en Salud (SES) para el desarrollo del SIGEC.

3. Generar los artefactos requeridos correspondientes a los flujos de trabajo indicados por la metodología de desarrollo Proceso Unificado de Rational (RUP), sirviendo de base para la confección del módulo.
4. Implementar las funcionalidades necesarias para obtener un módulo que gestione la extracción de datos en correspondencia con las necesidades del cliente.

Para la realización de la investigación, según las tareas definidas anteriormente, se utilizarán diferentes métodos que permitirán recolectar información importante para darle solución a la problemática planteada; entre los que se encuentran los siguientes:

Métodos teóricos

Analítico-Sintético: permitió realizar un análisis del problema de investigación y desglosarlo en la gestión de conjuntos de datos y trazas, profundizando en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

Histórico-Lógico: se empleó para realizar un estudio sobre aplicaciones informáticas que permiten la extracción de datos de ensayos clínicos implementadas en Cuba y en el mundo, permitiendo conocer el estado de la evolución actual del fenómeno identificando posibles mejoras y alternativas de solución.

Inductivo-Deductivo: se utilizó para, luego de haber consultado la bibliografía, obtener los elementos generales que serán utilizados en la investigación, profundizando en los contenidos de la gestión de conjuntos de datos y trazas.

Modelación: este se aplicó para representar mediante diagramas y esquemas el proceso de gestión de conjuntos de datos y trazas, lo que permite un mejor entendimiento del negocio.

Sistémico: este método se empleó para describir los conceptos asociados a la gestión de conjuntos de datos y trazas como elementos fundamentales en la extracción de datos de un ensayo clínico.

Métodos empíricos

Entrevista: se efectuó con el objetivo de conocer como gestionan los conjuntos de datos y las trazas en el Clínicas 1.0 los especialistas del CIM, y la importancia que le merecen la realización de estos procesos. La misma se encontrará en el Anexo 1.

Observación: se empleó con el objetivo de tener un registro visual acerca del objeto de investigación en el sistema Clínicas 1.0 y que pueda ser de utilidad en la realización de la investigación.

Beneficios esperados

Una vez desarrollado el módulo Extracción para el SIGEG, se obtendrán los siguientes beneficios:

- Facilitará el proceso de gestión de conjuntos de datos a través de los grupos de sujetos permitiéndole a los especialistas una mejor manipulación de la información a la hora de trabajar con los mismos.
- Servirá de soporte a la toma de decisiones de los especialistas, proporcionándole reportes en diversos formatos útiles para posteriores análisis.
- Ofrecerá una mayor seguridad de la información al sistema SIGEC con la adición del control sobre los eventos en otros módulos, a través de las trazas.
- Favorecerá al desarrollo más certero de los ensayos, con una mayor validez en la investigación, lo que permitirá estimar con mayor veracidad el impacto de los mismos en la salud de los pacientes.

El documento está estructurado por cuatro capítulos:

En el **Capítulo 1: Fundamentación Teórica** se realiza un estudio del estado del arte a nivel internacional y nacional sobre los sistemas que gestionan ensayos clínicos, además de estudiar las tecnologías, metodología y herramientas que se van a emplear para la solución al problema planteado.

En el **Capítulo 2: Características del Sistema** se obtiene el modelo conceptual, los requisitos funcionales y no funcionales, y se presentan algunas descripciones de los casos de uso del sistema.

En el **Capítulo 3: Diseño del Sistema** se incluye la arquitectura y los patrones de diseño utilizados en el desarrollo del módulo; se realiza el modelo de diseño.

Finalmente en el **Capítulo 4: Implementación del Sistema** se presentan los diagramas de componentes y de despliegue, los estándares y estilos utilizados así como ejemplos de las interfaces obtenidas; además de la descripción de las clases que conforman el modelo de datos.

1 Capítulo 1: Fundamentación teórica

En el presente capítulo se abordan elementos teóricos para lograr un mayor entendimiento de la gestión de conjuntos de datos y trazas en los ensayos clínicos. Se realiza además una investigación a nivel internacional y nacional de sistemas similares, y una descripción de las tecnologías, metodología y software usados para solucionar el problema planteado anteriormente.

1.1 Conceptos asociados al campo de acción

A continuación se muestran algunos conceptos importantes relacionados con el campo de acción planteado, los cuales permiten una mejor comprensión de los aspectos a tratar en la investigación.

Variable: rasgo, característica, valor, dato que se asocia a cada sujeto de un estudio.

Sujeto: paciente insertado en un estudio, los mismos transitan por distintos estados mientras son realizadas las pruebas correspondientes.

Grupo de sujeto: es la agrupación de varios pacientes que comparten una característica en común la cual requiere de un estudio separado.

Conjunto de datos: los conjuntos de datos se forman con la selección de las variables asociadas a los pacientes incluidos en un estudio.

Gestión de conjuntos de datos: posibilita u ofrece cierta información a los especialistas para la toma de decisiones en los estudios que se realizan. Un conjunto de datos está compuesto por una agrupación de variables útiles para desarrollar técnicas de análisis estadísticos.

Traza: contiene información sobre una acción realizada sobre un paciente, MS, hoja CRD, variable o centro de estudio. Esta información puede estar compuesta por la fecha de la acción, usuario que la ejecutó, tabla sobre la que se efectuó, entre otras.

Gestión de trazas: controlar la ocurrencia de eventos o acciones en cualquier sistema informático ofreciendo, a los usuarios interesados, información sobre posible afectación de la confidencialidad, integralidad, disponibilidad de la información.

Hojas CRD: son los modelos que componen cada cuaderno de recogida de datos, los cuales contienen parámetros a medir denominados variables.

Momentos de seguimiento: son los distintos momentos que se definen en un estudio donde estarán involucrados los pacientes durante el desarrollo de un ensayo clínico y a los cuales se les asocian las hojas CRD.

1.2 Estado del arte

Los sistemas de gestión de ensayos clínicos representan un gran impacto en la optimización de los procesos para la elaboración de fármacos, con el objetivo de dar respuesta a posibles epidemias que afectan al ser humano.

1.2.1 Sistemas existentes a nivel internacional para la gestión de ensayos clínicos

A nivel internacional han sido elaboradas y diseñadas varias soluciones informáticas, las cuales realizan la extracción de datos en ensayos clínicos. A continuación se muestra la descripción de algunos de estos sistemas.

- **OpenClinica**

OpenClinica es un CTMS[4] (en inglés *Clinical Trial Management System*) líder en la industria software, para la conducción, captura y manejo de los datos de los ensayos clínicos. Permite crear tus propios estudios, diseñar cuadernos de recogida de datos electrónicos y llevar a cabo una serie de funciones de captura y administración de datos clínicos. Está diseñado para ser usado en diferentes tipos de estudios clínicos. Se construye en una arquitectura moderna usando estándares libres. (8)

[4] CTMS: Un Sistema de Gestión de Ensayo Clínicos (CTMS) es un sistema de software adaptable usado por industrias biotecnológicas, farmacéuticas e instituciones de investigaciones clínicas para administrar la gran cantidad de datos involucrados en ensayos clínicos.

Funciones principales que desarrolla: (8)

- Enviar Datos: interfaz intuitiva para el registro del sujeto, la captura de datos clínicos, la validación y administración de consultas.
- Monitorear y administrar datos: las herramientas para la limpieza y administración de datos clínicos, y el monitoreo del sitio.
- Extraer Datos: definición personalizada y obtención de conjuntos de datos clínicos en tiempo real y en una variedad de formatos.
- Construcción del estudio: herramientas de diseño para la configuración del protocolo del estudio: sitios, cuadernos de recogida de datos, verificaciones de edición/reglas, usuarios y definiciones de momentos de seguimiento en el estudio.
- Roles de usuario: designar un acceso adecuado de usuarios a nivel de sitio y de estudio.
- Administración: herramientas para la supervisión del sistema en general, la auditoría, la configuración y presentación de informes.

Otras características: (8)

- Organiza la investigación clínica por el protocolo de estudio y el sitio, cada uno con su grupo de usuarios autorizados, los sujetos, las definiciones momentos de seguimiento en el estudio, y CRD. Soporte para compartición de recursos a través de estudios de una manera segura.
- Capacidades automatizadas y manuales para la importación, exportación y de intercambio de datos con otros sistemas.
- Motor de reglas realizando una validación avanzada de los datos del estudio y definir acciones automatizadas en el sistema.
- Una infraestructura tecnológica robusta y escalable desarrollada utilizando el marco de Java J2EE (en inglés *Java Plataform Enterprise Edition*) y de potente base de de datos PostgreSQL.

❖ **IMPACT CTMS**

Es una solución ampliamente usada por las mejores compañías farmacéuticas para planificar, administrar y rastrear cada aspecto de ensayos clínicos. IMPACT CTMS se ha desarrollado con la máxima flexibilidad y capacidad de configuración en mente. La excepcional amplitud de funcionalidades es muy adaptable a

los flujos de trabajo y terminologías específicas del patrocinador. Con IMPACT CTMS, los clientes no tienen que modificar sus prácticas de trabajo. El sistema es capaz de gestionar los ensayos de cualquier complejidad y soporta organizaciones de cualquier tamaño y alcance global. Contiene todos los datos de los ensayos operativos y permite que la información consistente y precisa sea compartida a través de toda la empresa, lo que facilita la toma de decisiones oportuna e informada. (9)

Módulos que presenta: (9)

- Progreso: planificación y seguimiento de la información sobre todos los proyectos y ensayos, a través de los países y lugares donde se realizan, los detalles de los pacientes y su progreso a través del ensayo.
- Mi Sitio: soporte para las actividades de monitoreo en línea y fuera de línea, y la recogida de datos durante las visitas de los supervisores de campo.
- Investigador: herramienta de selección y gestión de investigador avanzada. Proporciona un almacén de datos centralizado de información sobre los investigadores, accesibles a través de una herramienta de búsqueda avanzada que posibilita a los usuarios identificar rápidamente los sitios de investigación más relevantes y de alta calidad.
- Seguimiento de Documentos Reglamentarios: permite el seguimiento de los documentos reglamentarios en todo el ciclo de vida del estudio, asegura que todos los documentos estén completos en el momento oportuno, reduce retrasos de puesta en marcha del estudio y aumenta el cumplimiento normativo.
- Seguimiento de Costos Clínicos: la planificación, la presupuestación, la proyección y el seguimiento de los costos asociados a los ensayos clínicos.
- Seguimiento de Suministros Clínicos: predicción, despacho y seguimiento de suministros.

Características principales: (9)

- Con todas las funciones CTMS que posee proporciona planificación, monitoreo, administración financiera, seguimiento de suministros en ensayos clínicos y funcionalidades para gestión de investigadores.

- Permite el despliegue rápido y flexible, sin los altos costos iniciales o inversiones en infraestructura en curso.
- Despliegue global incluyendo el lenguaje internacional y soporte de moneda.
- Configurable y adaptable para un flujo de trabajo especifico del cliente.

❖ **OnCore**

Enterprise OnCore es un comprensible CTMS basado en web que ofrece administración segura del ciclo de vida de un ensayo clínico y participantes en el estudio, además permite la captura de datos electrónicos y realización de reportes. Este ofrece registros de pacientes e integración con otros sistemas empresariales. Además proporciona los siguientes beneficios: (10)

- Integración con otros sistemas empresariales líderes para reducir entrada de datos duplicados e incrementar la calidad de los mismos.
- Provee una vista del calendario de cuando los practicantes en la investigación están programados para visitas.
- Soporte para reclutamiento de participantes en el estudio a través de la web pública.
- Almacena y proporciona acceso a los datos registrados sobre pacientes.
- Acceso a reportes del estado financiero de la conducción de la investigación.
- Provee acceso a información actualizada del estudio a los proveedores de salud de los pacientes a través de la Web.
- Provee envío de información, comunicación, acciones y monitoreo seguro de datos para varios cuerpos regulatorios incluyendo al Comité de Revisión Científica FDA (en inglés *Food and Drug Administration*).

1.2.2 Sistemas existentes a nivel nacional para la gestión de ensayos clínicos

A nivel nacional se encuentra el sistema Clínicas 1.0 el cual está desplegado en el CIM y es utilizado por los especialistas que trabajan en dicho centro.

❖ **Clínicas 1.0**

Es un sistema desarrollado en la UCI, abarca las funcionalidades de OpenClinica y además brinda otras como: la realización de un cronograma de ejecución general para los ensayos y la generación del cronograma específico para cada paciente, la validación de las variables de los ensayos clínicos, permitiendo la disminución y detección de errores en los datos, el reporte de las trazas de las acciones realizadas en el sistema; así como las restricciones de las direcciones IP[5] (en inglés *Internet Protocol*) desde las cuales se puede acceder al mismo, entre otras. (11)

Clínicas 1.0 cuenta con cuatro módulos que se interrelacionan entre sí creando dependencias entre ellos para lograr un buen funcionamiento. Administrar Sistema es el módulo en que el administrador del sistema gestiona los usuarios que tendrán acceso a los módulos restantes del mismo. Gestionar Estudio es el que se encarga de gestionar los estudios y centros asociados al ensayo clínico, así como la creación de los CRD, las reglas de validación y los cronogramas. En Enviar Datos se procede a la ejecución de los ensayos clínicos una vez seleccionados los estudios y centros, con la inserción de los sujetos y el llenado de las hojas CRD. Finalmente cuenta con el módulo Extraer Datos en el cual se gestionan los conjuntos de datos y las trazas del módulo Enviar Datos.

Otras particularidades de interés:

- Permite la gestión de varios ensayos clínicos.
- Liberado por el Centro Nacional de Calidad de Software en diciembre de 2011.
- Se han diseñado estudios reales, y los mismos se conducen desde varias provincias del país con resultados satisfactorios.

Tecnología en la cual está implementado:

- J2EE.
- Eclipse (versión 3.4.2).
- Apache Tomcat (versión 5.5.27).
- PostgreSQL 8.2.9.1.
- Subversion 1.6.

[5] IP: Internet Protocol es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

1.2.3 Análisis de los sistemas encontrados

Después de realizado un análisis de los sistemas existentes a nivel internacional, se encontraron una serie de características presentes en estos que no se adaptan a las necesidades del CIM y también inconvenientes que representan un cambio en el paradigma de conducir ensayos clínicos en Cuba. Algunos de estos sistemas conllevan a un gasto superior en adquisición debido a su integración con otros, para poder cumplir con las exigencias de los centros promotores de ensayos clínicos cubanos; cuando es posible en un único sistema realizar la gestión eficiente de un ensayo clínico, ejemplo: el sistema IMPACT CTMS para poder realizar la captura de datos clínicos, realizar reportes y exportar conjuntos de datos se apoya en el sistema Datalabs EDC[6] (en inglés *Electronic Data Capture*) el cual es un SaaS[7] (en inglés *Software As a Service*). En los términos y condiciones a cumplir de estos sistemas se establecen limitaciones en cuanto a uso para Cuba y otros países, debido a la política trazada por los EEUU de no comercializar con países incluidos en su lista negra.

El sistema Enterprise OnCore contiene un módulo para la captura de datos electrónicos y auditorías de los eventos ocurridos en un ensayo. Soporta además reclutamiento de participantes en el estudio a través de la web, procedimiento que no se tiene en cuenta en los ensayos clínicos cubanos producto a la inexistencia de una infraestructura y seguridad en la transferencia de información que abarque o facilite esta característica. También presenta el inconveniente de que el acceso a su información es limitada debido a que es necesario ser autorizados para acceder a la misma, siendo imposible indagar en el estudio del sistema.

El sistema OpenClinica presenta el módulo Extraer Datos y Administración, encargados de la gestión de conjuntos de datos para todos los pacientes de un estudio y la auditoría de las acciones realizadas en el sistema respectivamente. El primero presenta el inconveniente de que no gestiona los conjunto de datos para todos los grupos de sujetos pertenecientes a un estudio por lo que no se logra un mejor agrupamiento de la información a obtener de un estudio por el cliente; y el segundo no lleva un control de acciones, como los momentos de seguimiento creados y eliminados, realizados por un determinado

[6]EDC: Un sistema de captura de datos electrónicos es un sistema informático diseñado para la recogida de datos clínicos en formato electrónico para su uso principalmente en ensayos clínicos humanos.

[7] SaaS: El software como servicio es un modelo de entrega de software en el que el software y los datos asociados están alojados en el centro de la nube de los proveedores de software independientes o proveedores de servicios de aplicaciones.

usuario del sistema en el módulo Construcción del estudio, posibilitando el desconocimiento de la responsabilidad de un usuario ante una pérdida de información.

En el ámbito nacional se profundizó en el sistema Clínicas 1.0, donde la tecnología empleada para su desarrollo es parcialmente obsoleta, debido a que no se utilizó un marco de trabajo que logre cohesión entre tecnologías del lado de cliente con las del servidor, imposibilitando la integración con herramientas útiles que permiten el empleo de un conjunto de librerías para lograr un enriquecimiento de las vistas y consecuentemente ofrecer un producto más amigable para el cliente. Los marcos de trabajo proveen a los desarrolladores una capa de abstracción y un grupo de funcionalidades para llevar a cabo determinadas tareas; por lo que su ausencia provoca que no se adquiera una mayor facilidad de mantenimiento del software con el surgimiento de nuevos requerimientos o versiones del sistema, a petición del cliente, afectando a estos en tiempo de espera para poder hacer uso de la nueva solución. Además el no empleo de un ORM (en inglés *Object Relational Mapping*) como Hibernate conlleva a un trabajo excesivo y posibilita la ocurrencia de errores debidos a la cantidad de ficheros .xml necesarios para lograr una comunicación entre la Programación Orientada a Objetos (POO) y el modelo relacional.

Luego de haber estudiado todos estos sistemas a nivel internacional, se concluyó que ninguno de ellos soluciona el problema planteado, por lo que no serán utilizados; y en el caso del sistema a nivel nacional, Clínicas 1.0, será tomado como base para el desarrollo de la solución de la presente investigación.

1.3 Tecnologías, metodología y herramientas en los que se apoya la solución del problema

Un paso importante en el inicio del desarrollo de un software es el estudio de las tecnologías, metodología y herramientas a emplear. A continuación se presentan herramientas de código abierto a considerar, con el objetivo de minimizar los costos y optimizar los resultados. Estas tecnologías, metodología y herramientas fueron definidas en el departamento SES para el desarrollo de sus aplicaciones y se encuentran descritas en el documento SES_SIGEC_010216_6 Arquitectura Vista de Entorno de Desarrollo Tecnológico.

1.3.1 Servidor Web

Un servidor web es un programa que escucha las peticiones “http” de los usuarios realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación del lado del cliente. (12)

- **JBoss AS 4.2**

JBoss AS es el primer servidor de aplicaciones de código abierto preparado para la producción y certificado por J2EE 1.4. Ofrece una plataforma de alto rendimiento para aplicaciones de negocios. Combina una arquitectura orientada a servicios con una licencia de código abierto. Por este motivo es la plataforma más popular para desarrolladores, vendedores independientes de software y para grandes empresas. (13)

Las características destacadas de JBoss incluyen: (13)

- Producto de licencia de código abierto sin coste adicional.
- Confiable a nivel de empresa.
- Orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Provee servicios extendidos de clusterización, cacheo y persistencia.
- Permite la integración de todas las tecnologías y herramientas utilizadas por JBoss Seam.

1.3.2 Tecnologías del lado del cliente

- **XHTML**

HTML es el acrónimo por sus siglas en español del lenguaje de marcado de hipertexto. Se utiliza para crear todas las páginas web de Internet. Es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado W3C (en inglés *World Wide Web Consortium*). Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de la misma manera en cualquier navegador de cualquier sistema operativo. (14)

El Lenguaje de Marcado de Hipertexto Extensible (XHTML), es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML, ante su limitación de uso con las herramientas basadas en XML (en inglés de *eXtensible Markup Language*). Su objetivo es lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas.

- **Hojas de Estilo en Cascada (CSS 3)**

CSS (en inglés *Cascading Style Sheets*) es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación, y es imprescindible para la creación de páginas web complejas. La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (15)

Otras ventajas que brinda: (15)

- Se obtiene un mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación.
- Hace mucho más legible el código HTML al tener el código CSS aparte (siempre que no se use estilos en línea).
- Las novedades de CSS3 permiten ahorrar tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas, etc.) sin necesidad de usar un editor gráfico.

- **JavaScript 1.1**

JavaScript es un lenguaje interpretado, basado en prototipos, que se inserta en una página web HTML con el objetivo de potenciarla y hacerla más dinámica. Posee la ventaja de que su aprendizaje y uso es muy sencillo logrando la realización de tareas complejas, además puede ser ejecutado en cualquier página web sin necesidad de instalar un programa para ser visualizado. No requiere de compilación ya

que el lenguaje funciona del lado del cliente siendo los navegadores los encargados de interpretar estos códigos. (16)

- **AJAX**

AJAX son las siglas de JavaScript asíncrono y XML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML, JavaScript, CSS, DHTML (HTML dinámico), PHP/ASP (en inglés *Active Server Pages*). NET/JSP (en inglés *Java Server Pages*), XML) que posibilitan hacer páginas de internet más interactivas. La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar se puede enviar información al servidor. (17)

Ventajas al emplear AJAX: (18)

- Utiliza tecnologías ya existentes.
- Soportada por la mayoría de los navegadores modernos.
- Interactividad. El usuario no tiene que esperar hasta que lleguen los datos del servidor.
- Portabilidad (no requiere plugins[8] como Flash y Applet de Java).
- Mayor velocidad, esto debido que no hay que retornar toda la página nuevamente.
- La página se asemeja a una aplicación de escritorio.

Ajax4JSF es una librería de código abierto que se integra totalmente a la arquitectura de JSF y extiende la funcionalidad de sus etiquetas, dotándolas con tecnología AJAX de forma limpia y sin necesidad de añadir código JavaScript.

1.3.3 Tecnologías del lado del servidor

- **Java**

Es un lenguaje de programación orientado a objetos desarrollado por la empresa Sun Microsystems, la misma describe a Java como simple, distribuido, interpretado, robusto, seguro, de arquitectura neutra,

[8] Plugins: funcionalidades extras desarrolladas en java que se le proporcionan al IDE para facilitar la tarea del programador.

portable, de altas prestaciones, multitarea y dinámico. Se utiliza en una gran variedad de dispositivos móviles, como teléfonos y pequeños electrodomésticos. Presenta flexibilidad, es decir, está especialmente preparado para la reutilización del código. Es de fuente abierta y multiplataforma ya que no es el sistema operativo quien ejecuta las aplicaciones sino la Máquina Virtual de Java (por sus siglas en inglés *JVM*). (19)

- **JasperReports 3.0.1**

La biblioteca JasperReports es motor de informes de código abierto más popular del mundo. Está escrito completamente en Java y es capaz de utilizar los datos procedentes de cualquier tipo de fuente de datos y producir documentos de píxel perfecto que se pueden ver, imprimir o exportar en una variedad de formatos de documentos incluyendo HTML, PDF, Excel, Open Office y Word. (20)

1.3.4 Marcos de trabajo

- **Java Server Faces (JSF)**

Es un marco de trabajo (en inglés *framework*) de componentes Modelo-Vista-Controlador (MVC) para aplicaciones web. JSF usa un enfoque basado en componentes para el desarrollo de la web, diferente a previos marcos de trabajo como Struts. Este enfoque dio lugar a la creación de ambientes de desarrollo con estilos de arrastrar y soltar (componentes). Esta tecnología está compuesta por los siguientes elementos: (21)

- Un API (Interfaz de Programación de Aplicaciones) para representar los componentes de la interfaz de usuario y gestionar su estado; a través de la recogida de eventos, realización de validaciones y conversiones de datos en el lado del servidor; definir la navegación entre páginas.
- Dos librerías de etiquetas para paginas JSP, permitiendo la definición de los componentes dentro de estas y vincularlos a los objetos del modelo del lado del servidor.

Ventajas de JSF: (21)

- Es considerado una tecnología estándar en Java EE y tiene un largo ecosistema de usuarios y proveedores.

- Es soportado por todos los servidores de aplicación Java.
- Es completamente basado en componentes.
- Contiene un poderoso lenguaje de expresión (por sus siglas en inglés EL) que puede ser usado en páginas web.
- Contiene soporte por herramientas de interfaz gráfica de usuario en todos los líderes entornos de desarrollo integrado de Java.

Características soportadas para los componentes de interfaz gráfica de usuario:

- Validación de entrada.
- Manipulación de eventos.
- Conversión de datos entre componentes y objetos del modelo.
- Administración de creación de objetos del modelo.
- Configuración de la navegación de las páginas.

❖ **RichFaces**

RichFaces es un marco de trabajo de código abierto que añade la capacidad de AJAX en las aplicaciones JSF existentes sin recurrir a JavaScript. Aprovecha el marco de trabajo de JSF incluyendo el ciclo de vida, validación, facilidades de conversión y la gestión de recursos estáticos y dinámicos. Los componentes RichFaces son altamente personalizables, con una interfaz de gran calidad visual y pueden ser fácilmente incorporados en aplicaciones JSF. (22)

❖ **Seam**

JBossSeam es un ligero framework para Java EE 5.0. Integra tecnologías estándares Java EE con varias tecnologías no estándar pero interesantes en un consistente y unificado modelo de programación. Esas tecnologías incluyen JSF, EJB3 (en inglés *Enterprise Java Beans*), JPA (en inglés *Java Persistence API*), Hibernate, Facelets, jBPM (en inglés *Business Process Management*), JBoss Rules (Drools) y mucho más. Seam se ejecuta en casi todos los líderes servidores de aplicaciones Java incluyendo pero no limitado a JBoss AS y Tomcat. A continuación, algunas ventajas de su uso: (23)

- Colapsa la capa artificial entre EJB3 y JSF. Proporciona un enfoque coherente y basado en anotación para integrar EJB3 y JSF.
- Elimina la mayor parte del código repetitivo y configuración XML desde sus aplicaciones. Seam abre a los desarrolladores web a herramientas útiles que eran demasiado difíciles de integrar en las aplicaciones web anteriormente.
- En comparación con las aplicaciones desarrolladas en otros frameworks web, las aplicaciones desarrolladas en Seam son conceptualmente simples y requieren significantemente menos código para las mismas funcionalidades a desarrollar.
- Provee un número de etiquetas de componentes JSF y anotaciones que aumentan la "amabilidad web" y la eficiencia de páginas web de aplicaciones JSF.
- Proporciona su propio contexto de estado gestionado que permite a los frameworks integrarse profundamente vía anotaciones y lenguaje de expresiones (en lo adelante EL).
- Automáticamente mantiene actualizaciones de la base de datos en memoria y solo actualiza la base de datos al final de una conversación. Mantener en memoria cache reduce grandemente la carga de la base de datos en complejas aplicaciones.
- Ayuda a cerrar la brecha entre la programación del lado del servidor y programación web.
- Proporciona soporte AJAX vía integración ajustada con la librería AJAX4JSF ayudando a aliviar el escribir JavaScript manualmente.
- Provee mecanismos de seguridad basados en usuarios y en roles.

❖ **Facelets**

Es un framework que proporciona una alternativa a la tecnología de vista JSP y es una de las tecnologías de vista soportadas por Seam. Ofrece varias ventajas sobre JSP como tecnología de vista para nuestras aplicaciones web como son las siguientes: (24)

- Plantillas para componentes y páginas.

Es la principal ventaja sobre JSP, ya que nos posibilita un enfoque basado en plantillas para construir páginas web lográndose un diseño basado en componentes más fácil de lograr. La etiquetas <ui:

composition/>, <ui: insert />, <ui: define />, permite alcanzar alto nivel de manipulación sobre el diseño de nuestro proyecto.

- Soporte para EL.

Dentro de páginas JSF se puede hacer uso de su lenguaje de expresión pero con Facelets se consigue adicionar funcionalidades a este lenguaje de expresión. Este lenguaje utilizado por Facelets es denominado Lenguaje de Expresión Unificado.

- Rapidez en compilación.

Cuando una página JSP es accedida por primera vez es compilada a servlet consumiendo recursos en términos de memoria y tiempo lográndose solucionar a través de la precompilación. Con Facelets la precompilación no es necesaria ya que un árbol de componentes con todos los componentes JSF en la página es construido.

- Uso de XHTML para crear páginas web.
- Soporte para librería de etiquetas adicionales a las empleadas en JSF y JSTL[9] (en inglés *Java Server Pages Standard Tag Library*).
- Su uso reduce el tiempo y esfuerzo a dedicar en el desarrollo y despliegue.

1.3.5 Acceso a datos

❖ **Hibernate**

Hibernate es una herramienta de mapeo objeto/relacional (ORM) para entornos Java. El término de ORM se refiere a la técnica de mapear una representación de datos desde un modelo de objeto a un modelo de datos relacionales con un esquema basado en SQL. Hibernate se ocupa del mapeo desde las clases Java a las tablas de las bases de datos (desde los tipos de datos de Java a los tipos de datos de SQL), y además facilita la consulta y recuperación de datos. Esto permite reducir de manera importante el tiempo

[9]JSTL: es un componente de Java EE. Extiende las JSP proporcionando cuatro bibliotecas de etiquetas (Tag Libraries) con utilidades ampliamente utilizadas en el desarrollo de páginas web dinámicas.

de desarrollo si se compara con el manejo de datos de forma manual en SQL y JDBC[10] (en inglés *Java Database Conections*). (25)

- **Enterprise Java Beans (EJB3)**

La tecnología EJB es una arquitectura de componentes para la plataforma Java EE. 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.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables. (26)

- **JPA**

El API de persistencia de Java es una solución basada en estándares para persistencia. La persistencia usa un enfoque mapeo objeto-relacional para lograr un vínculo entre el modelo orientado a objetos y una base de datos relacional. La persistencia en Java consiste de tres aéreas: (23)

- El API de persistencia de Java
- Lenguaje de Consultas
- Metadatos de mapeo objeto-relacional

1.3.6 Entorno de Desarrollo Integrado (IDE)

Un IDE (acrónimo en inglés de *Integrated Development Environment*) es un entorno de programación que ha sido empaquetado como un programa informático compuesto por un conjunto de herramientas útiles para un programador. Está formado por un editor de texto, un compilador, un intérprete, un depurador y un constructor de interfaz gráfica de usuario. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

[10] 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.

- **Eclipse 3.4.2**

Eclipse es un IDE, de código abierto[11]y multiplataforma[12]. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Emplea módulos (en inglés *plugins*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Además Eclipse es uno de los IDE más utilizados por los programadores para desarrollar aplicaciones. Aunque puede usar diferentes lenguajes de programación, está pensado para el desarrollo de aplicaciones Java (27). También se le puede incorporar la herremienta JBoss Tools que es un conjunto de plugins que posibilitan el manejo de diferentes frameworks facilitando el desarrollo de aplicaciones web.

1.3.7 Sistema Gestor de Base de Datos (SGBD)

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, manipulación de datos y consulta. (28)

- **PostgreSQL 8.4**

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (en inglés *Berkeley Software Distribution*) y con su código fuente disponible libremente. Utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilo para garantizar la estabilidad del sistema. Tiene la característica que ante un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. La comunidad de usuarios de PostgreSQL es una comunidad muy activa y con un alto nivel técnico por lo que se puede encontrar gran soporte en línea. (28)

- **PgAdmin III 1.14 como herramienta para el manejo de bases de datos**

[11] Código abierto: término con el que se conoce al software distribuido y desarrollado libremente.
[12] Multiplataforma: término usado para referirse a los programas, sistemas operativos, u otra clase de software que pueden funcionar en diversas plataformas.

Es una aplicación gráfica multiplataforma para gestionar el gestor de bases de datos PostgreSQL, siendo el más completo y popular con licencia código abierto. PgAdmin es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas por lo que posee una amplia documentación y una interfaz multilingüe. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración, también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados. La conexión al servidor puede hacerse mediante conexión TCP/IP (en inglés *Transmission Control Protocol/Internet Protocol*) y puede cifrarse mediante SSL[13] (en inglés *Secure Sockets Layer*) para mayor seguridad. (29)

1.3.8 Lenguaje Unificado de Modelado (UML 2.1)

El UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para emplearse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. La especificación de UML no define un proceso estándar pero está deliberado para ser útil en un proceso de desarrollo iterativo dando apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (30)

1.3.9 Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Además puede seguir uno o varios modelos del ciclo de vida indicando qué es lo que hay que obtener a lo largo del desarrollo del proyecto. (31)

- **RUP**

RUP (en inglés *Rational Unified Process*), en primer lugar, es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos

[13] SSL: son protocolos criptográficos que proporcionan comunicaciones seguras por una red.

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, aéreas de aplicación, tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyecto. (32)

Está basado en componentes, lo cual requiere decir que el sistema software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas aumentando la calidad del producto. Otra característica es que el UML es una parte esencial de este proceso pues lo utiliza para concebir todos los esquemas de un sistema software. Además minimiza la posibilidad de que riesgos críticos pongan en peligro el éxito del proyecto, debido a que esta guiado por los riesgos. (32)

Tiene entre sus características principales las siguientes: (32)

- Dirigido por casos de uso: los casos de uso reflejan lo que el cliente necesita, lo cual se capta al modelar el negocio y se representa a través de los requerimientos. Luego los casos de uso guían el proceso de desarrollo, ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.
- Centrado en la arquitectura: la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.

El proceso puede ser descrito en dos dimensiones o ejes: (31)

- Eje horizontal: representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la figura que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se subdivide a la vez en iteraciones.
- Eje vertical: representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

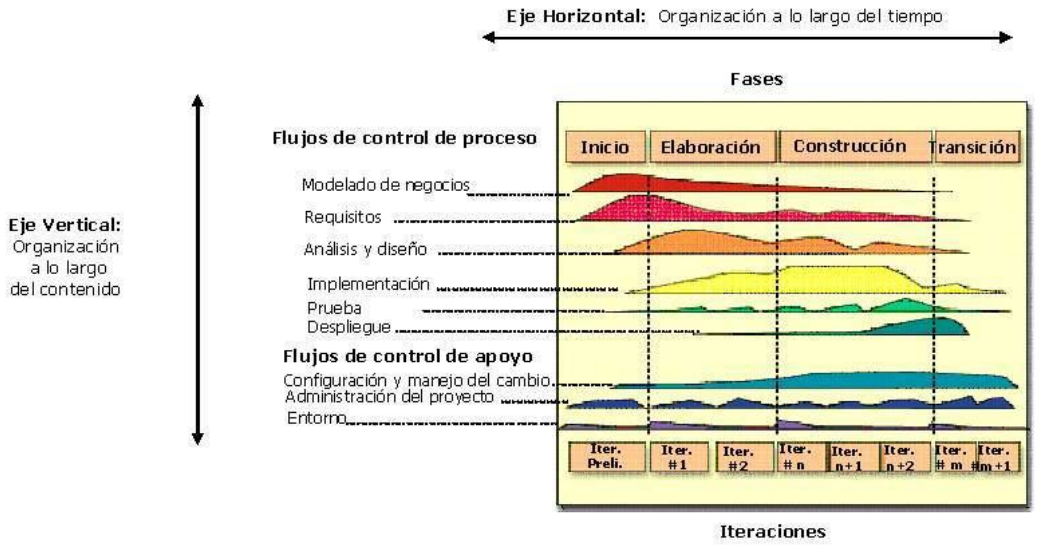

Figura 1: Estructura de RUP.

De los flujos de control de proceso enunciados por RUP solamente se van a aplicar en la presente investigación los siguientes: Modelado de negocios, Requisitos, Análisis y diseño e Implementación.

1.3.10 Herramientas CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras) son un conjunto de programas y ayudas que proporcionan asistencia a los analistas, ingenieros de software y desarrolladores, durante todas las fases del ciclo de vida del desarrollo de un software automatizando el proceso de diseño. (33)

- **Visual Paradigm 8.0**

Visual Paradigm 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. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como ingeniería inversa de bases de datos. (34)

En este capítulo, el estudio del estado del arte realizado permitió conocer sistemas existentes a nivel mundial utilizados para el proceso de extracción de datos en los ensayos clínicos. Estos sistemas no serán utilizados porque presentan funcionalidades que no cubren las expectativas a la solución del problema planteado o características no adaptables al flujo de trabajo del cliente. Además el estudio de las herramientas, metodología y tecnologías permitió la asimilación de las mismas facilitando su uso mediante el proceso de desarrollo de la solución, sirviendo como base Clínicas 1.0.

2 Capítulo 2: Características del módulo

En el presente capítulo se describen las características del módulo para facilitar un mejor entendimiento del problema. Se representa el modelo del dominio que permite tener una visión del entorno en que se está modelando; además se hace una especificación de las condiciones, capacidades y cualidades que el sistema debe tener; mostrándose en términos de requerimientos funcionales y no funcionales, así como el diagrama de casos de uso del sistema, y algunas de sus respectivas descripciones.

2.1 Propuesta de solución

Analizando los inconvenientes que posee el sistema Clínicas 1.0 descritos anteriormente y en el resto de sus módulos, los desarrolladores del proyecto determinaron realizar una nueva versión del mismo tomando como base sus funcionalidades, denominada SIGEC, que abarque la mayor parte de los procesos que se realizan en la gestión de los ensayos clínicos en el CIM. Por tanto, como solución a la situación problemática descrita en el inicio de este trabajo se propone desarrollar el módulo Extracción para el SIGEC, el cual garantizará que los especialistas del centro realicen a mayor nivel en calidad la extracción de datos clínicos. Con dicho módulo interactuarán los usuarios que según el rol y los permisos otorgados puedan gestionar las trazas y los conjuntos de datos en el sistema. Estos tendrán la posibilidad de visualizar las trazas generadas por los usuarios ante la ocurrencia de cualquier evento pudiendo exportarlas en varios formatos e imprimirlas. Además permitirá la gestión de conjuntos de datos por grupos de sujetos, lo que simplificaría el trabajo a los especialistas ofreciéndoles una mejor organización de la información a manejar presente en el estudio, también pudiéndose exportar los mismos en diferentes formatos como son SPSS, XLS, CDISC ODM xml 1.2 y HTML.

2.2 Modelo de dominio

El paso esencial de un análisis o investigación orientado a objetos es descomponer el problema en conceptos u objetos individuales: las cosas que se conocen. Un modelo conceptual es una representación de conceptos en un dominio del problema, donde se ilustra con un grupo de diagramas de estructura estática donde no se define ninguna operación, utilizando la notación UML. La designación de modelo conceptual ofrece la ventaja de subrayar fuertemente una concentración en los conceptos del dominio, no

en las entidades del software. Puede mostrar: conceptos, asociaciones entre conceptos y atributos de conceptos. (35)

A continuación se muestra el modelo de dominio del módulo Extracción, el cual aglomera los principales conceptos que se identifican en la gestión de los conjuntos de datos y trazas que tienen lugar en el mismo.

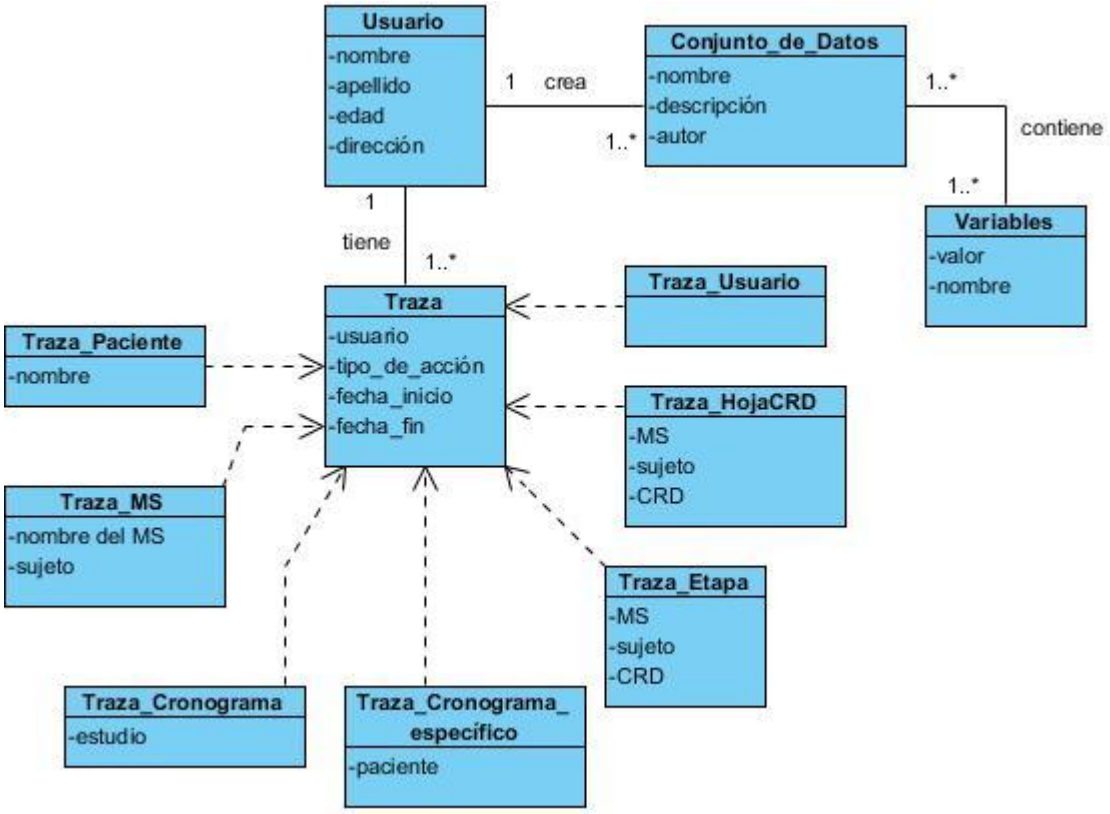

Figura 2: Modelo de dominio del módulo Extracción.

2.2.1 Definición de los conceptos del modelo de dominio

Conocer el significado de los conceptos más relevantes en el dominio del problema es de gran ayuda para comprender el modelo del dominio. Estos conceptos se explican a continuación:

Usuario: es una representación de los roles gerente de datos, investigador principal, investigador promotor, coordinador y monitor, los cuales son responsables de la extracción de datos en el sistema.

Conjunto_de_datos: representa la agrupación de la información recogida en el sistema, en dependencia de las necesidades del especialista para realizar su análisis. Están formados por las variables que seleccione el usuario, perteneciente a hojas CRD utilizadas para recoger información de un paciente en un MS determinado.

Variables: contiene información de un indicador sobre un paciente, como por ejemplo: temperatura corporal, presión arterial, entre otras.

Traza: contiene información útil sobre una acción realizada por un usuario en el sistema, es decir, sobre las acciones que ocurren durante la realización del estudio.

Traza_Paciente: muestra las trazas vinculadas a los pacientes creados, deshabilitados y eliminados en un estudio.

Traza_MS: muestra las trazas vinculadas a los momentos de seguimiento creados, modificados y eliminados de los módulos Diseño y Conducción.

Traza_Cronograma: muestra las trazas vinculadas a los cronogramas generales creados y completados en el estudio.

Traza_Cronograma_específico: muestra las trazas vinculadas a la modificación de los cronogramas por pacientes en el estudio.

Traza_Etapa: muestra las trazas vinculadas a las etapas modificadas en el estudio.

Traza_Usuario: contiene información útil de los usuarios en su momento de conexión y desconexión del sistema. La traza muestra los datos del usuario, la fecha y la hora de acceso al sistema.

Traza_HojaCRD: contiene información útil sobre una acción relacionada en una hoja CRD.

2.3 Especificación de los requisitos del módulo Extracción

Los requisitos, también conocidos como requerimientos, son las condiciones y capacidades que un software debe cumplir para satisfacer una necesidad de un cliente en específico. Los requisitos se pueden clasificar en: funcionales y no funcionales.

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de los requisitos suficientemente buena como para que pueda llegarse a un acuerdo entre el cliente y los desarrolladores sobre que debe y no debe hacer el sistema. (32)

2.3.1 Requisitos funcionales

Los requisitos funcionales representan la funcionalidad del sistema, indican que es lo que el software debe hacer, especifican cómo debe comportarse el sistema en situaciones particulares. A continuación se presentan las funcionalidades que brindará el módulo Extracción.

Tabla 1: Requisitos funcionales del módulo Extracción.

RF 1: Crear un conjunto de datos.	**RF 16:** Mostrar trazas de la funcionalidad eliminar MS del módulo Conducción.
RF 2: Mostrar listado de conjuntos de datos creados.	**RF 17:** Mostrar trazas de la funcionalidad modificar MS del módulo Diseño.
RF 3: Editar un conjunto de datos.	**RF 18:** Mostrar trazas de la funcionalidad crear MS del módulo Diseño.
RF 4: Eliminar un conjunto de datos.	**RF 19:** Mostrar trazas de la funcionalidad eliminar MS del módulo Diseño.
RF 5: Ver los datos de un conjunto de datos creado.	**RF 20:** Mostrar trazas de la funcionalidad crear cronograma general.
RF 6: Exportar un conjunto de datos en diferentes formatos.	**RF 21:** Mostrar trazas de la funcionalidad completar cronograma.
RF 7: Ver conjunto de datos en HTML.	**RF 22:** Mostrar trazas de la funcionalidad modificar cronograma específico.
RF 8: Buscar conjuntos de datos creados.	**RF 23:** Mostrar trazas de la funcionalidad modificar etapas.
RF 9: Mostrar detalles del listado de los conjuntos de datos creados.	**RF 24:** Mostrar trazas de las hojas CRD firmadas.

RF 10: Mostrar los conjuntos de datos creados por el usuario logueado en el sistema.	**RF 25:** Mostrar trazas de las hojas CRD modificadas.
RF 11: Mostrar trazas de la funcionalidad insertar paciente.	**RF 26:** Mostrar trazas de las hojas CRD iniciadas.
RF 12: Mostrar trazas de la funcionalidad deshabilitar paciente.	**RF 27:** Mostrar trazas de los momentos de conexión de los usuarios.
RF 13: Mostrar trazas de la funcionalidad eliminar paciente.	**RF 28:** Mostrar trazas de los momentos de desconexión de los usuarios.
RF 14: Mostrar trazas de la funcionalidad modificar MS del módulo Conducción.	**RF 29:** Exportar trazas.
RF 15: Mostrar trazas de la funcionalidad crear MS del módulo Conducción.	**RF 30:** Imprimir trazas.

2.3.2 Requisitos no funcionales.

Los requisitos no funcionales especifican propiedades, apariencia o interfaz externa del sistema. A continuación se describe un conjunto de requisitos no funcionales que incluirá el módulo Extracción, los cuales se encuentran en el documento: SES-SIGEC-010113_Especificación de requisitos de software módulo Extracción_v1.1.

- **Usabilidad**

RNF1: Las personas que interactuarán con el software serán médicos, clínicos, técnicos y especialistas de la salud ubicados en el CIM.

RNF2: La aplicación web tendrá un ambiente sencillo y fácil de manejar para todos los usuarios incluso para aquellos inexpertos en el trabajo con aplicaciones informáticas.

RNF3: Para su correcto funcionamiento el sistema requiere un servidor de base de datos con las siguientes características: procesador Intel Dual Core, Sistema Operativo Windows 7 o superior y Linux, memoria RAM 4GB, 500 GB de disco duro.

RNF4: Para su correcto funcionamiento el sistema requiere un servidor de aplicaciones con las siguientes características: procesador Intel Dual Core, Sistema Operativo Windows 7 o superior y Linux, memoria RAM 4GB y 120 GB de disco duro. Además, como servidor web JBoss 4.2.

RNF5: Los ordenadores clientes deberán contar con las siguientes características: procesador Intel Pentium IV o superior, Sistema Operativo Windows 7 o superior y Linux, memoria RAM 1GB y 80 GB de disco duro.

RNF6: El sistema podrá ser consultado desde los siguientes navegadores: Mozilla Firefox 3.6 o superior y Google Chrome 1.4 o superior.

RNF7: Debe estar instalada la máquina virtual de Java: java-7-openjdk para GNU Linux, o la jdk-6u3 o superior para Windows.

- **Confiabilidad**

RNF8: La aplicación tendrá un sistema de trazas que registran el flujo constante de los datos y los responsables de sus cambios.

RNF9: Se mantendrá la seguridad y el control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan.

RNF10: Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos.

- **Restricciones de diseño**

RNF11: Se usa como herramienta CASE Visual Paradigm para el modelado de los productos típicos de trabajo generados en cada fase del ciclo de vida.

RNF12: Se usa como lenguaje de programación Java.

RNF13: Se usa como Gestor de Base de Datos PostgreSQL.

- **Requisitos para la documentación y ayuda del sistema**

RNF 14: Se confeccionará un Manual de Usuario como ayuda a los usuarios finales del sistema.

- **Interfaz**

RNF15: Las páginas principales tendrán información que servirá de guía al usuario.

RNF16: Las páginas no están cargadas de imágenes y contarán con pocos colores.

RNF17: Cada rol tiene acceso a la interfaz que se corresponda con los permisos asignados.

RNF18: Se hará uso de simbología mediante íconos para indicar el estado de los elementos utilizados en el diseño. Además, los íconos contendrán funcionalidades específicas.

- **Estándares Aplicables**

RNF19: Cuenta con las pautas de diseño definidas para aplicaciones web del CESIM, permitiendo al usuario obtener de manera uniforme y organizada la información del sistema.

2.4 Definición de los casos de uso

Cada forma en que los actores usan el sistema se representa con un caso de uso. Los casos de uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores; es decir, un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia. (32)

Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo. Mediante la utilización de los casos de uso, los analistas se ven obligados a pensar en términos de quienes son los usuarios, y que necesidades u objetivos de la empresa pueden cumplir. (32)

- **Definición de los actores**

El modelo de casos de uso describe lo que hace el sistema para cada tipo de usuario. Cada uno de estos se representa mediante uno o más actores. Los actores suelen corresponderse con trabajadores en un negocio. (32)

Representación del actor que interactúa con el módulo, indicando la función que realiza:

Tabla 2: Descripción de los actores del módulo Extracción.

Actor	Justificación
Usuario	Según el rol y los permisos otorgados puede gestionar las trazas y los conjuntos de datos en el sistema.

- **Diagrama de casos de uso del módulo Extracción**

Un modelo de casos de uso del sistema es una representación gráfica que contiene actores, casos de uso y sus relaciones, describiendo las funcionalidades que genera el software modelado. El modelo de casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas. (32)

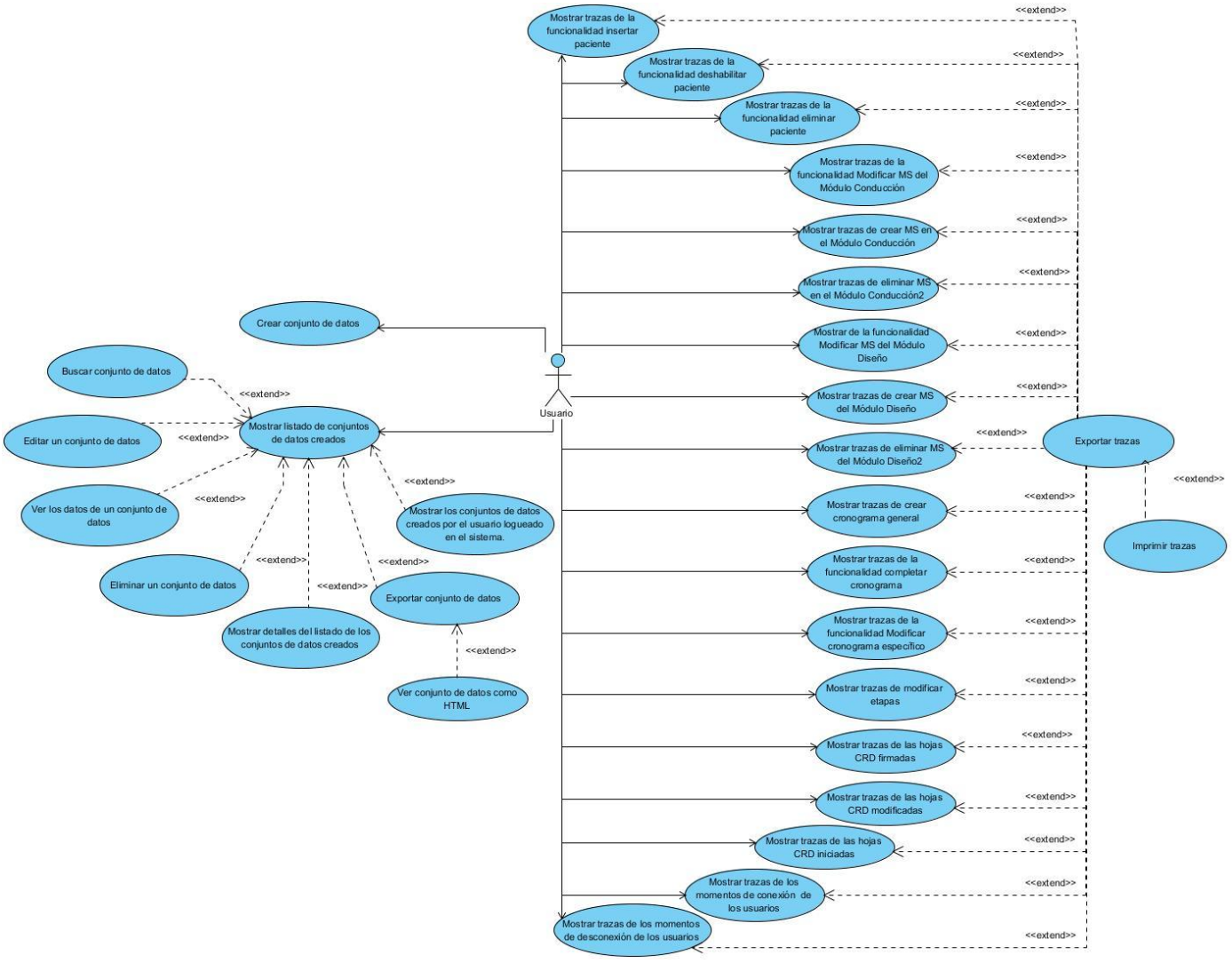

Figura 3: Diagrama de casos de uso del módulo Extracción.

- **Descripción de los casos de uso**

Seguidamente se describen textualmente algunos de los casos de uso identificados para la construcción del módulo; el resto se podrán encontrar en documento SES-SIGEC-0114_Especificación de casos de uso módulo Extracción_v1.1.

Tabla 3: Descripción del caso de uso “Ver datos de un conjunto de datos”.

Caso de uso	Ver los datos de un conjunto de datos	
Objetivo	Mostrar los datos de un conjunto de datos que ha sido creado en el estudio activo.	
Actores	Usuario: según el rol y los permisos otorgados, visualiza los datos de un conjunto de datos creado en el estudio.	
Resumen	El caso de uso inicia cuando el usuario accede a la sección Conjunto de Datos y selecciona el ícono “ver” en el listado de conjuntos de datos creados en el estudio, mostrando el sistema los datos que contiene el conjunto de datos indicado, termina el caso de uso.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El usuario ha sido logueado en el sistema. El usuario tiene los permisos para visualizar los datos de los conjuntos de datos creados en el estudio.	
Postcondiciones	Visualiza los datos de los conjuntos de datos que se han creado en el estudio.	
Referencias	RF 5	
Flujo de eventos		
Flujo básico Ver los datos de un conjunto de datos		
	Actor	**Sistema**
1.	Accede a la sección “Conjunto de Datos” del módulo Extracción.	
2.		Muestra una interfaz que contiene un listado de los conjuntos de datos que han sido creados en el estudio activo brindando la posibilidad de realizar las siguientes acciones sobre los mismos: • Ver • Editar • Eliminar • Exportar
3.	Selecciona la acción Ver.	
4.		Muestra en una interfaz con todos los atributos

		del conjunto de datos seleccionado: los datos de las hojas CRD, los datos del paciente, del MS, en fin, toda la información que conforma el conjunto de datos. Brinda además las opciones: • editar (Ver descripción del caso de uso) • exportar (Ver descripción del caso de uso)
5.		El caso de uso termina.
Relaciones	**CU Incluidos**	-
	CU Extendidos	-

Tabla 4: Descripción del caso de uso “Mostrar trazas de la funcionalidad insertar paciente”.

Caso de uso	**Mostrar trazas de la funcionalidad insertar paciente**
Objetivo	Visualizar las trazas de la funcionalidad insertar paciente.
Actores	Usuario: según el rol y los permisos otorgados, genera las trazas de la funcionalidad que requiera.
Resumen	El caso de uso inicia cuando el usuario accede a la sección Trazas, selecciona los criterios requeridos relacionados con la funcionalidad insertar paciente, el sistema muestra las trazas, terminando el caso de uso.
Complejidad	Media
Prioridad	Media
Precondiciones	El usuario ha sido logueado en el sistema. El usuario tiene los permisos para visualizar las trazas.
Postcondiciones	Visualiza las trazas de la funcionalidad insertar paciente.
Referencias	RF 11
Flujo de eventos	
Flujo básico Mostrar trazas de la funcionalidad insertar paciente	

Actor		Sistema
1.	Accede a la sección “Trazas” del módulo Extracción.	
2.		Muestra una interfaz que contiene las opciones para visualizar trazas: Diseño, Conducción, Hojas CRD y Usuarios.
3.	Selecciona la opción Conducción.	
4.		Muestra en la interfaz los campos para especificar los criterios deseados y los botones Aplicar Filtro y Limpiar Filtro. • Usuarios • MS • Sujetos • CRD • Fecha Inicio • Fecha Fin • Acción
5.	Despliega el campo Acción.	
6.		Muestra las acciones que se pueden realizar en el módulo: • Insertar Paciente • Deshabilitar Paciente • Eliminar Paciente • Crear MS • Eliminar MS • Modificar MS
7.	Introduce el criterio Insertar Paciente y demás campos que requiera. Presiona el botón Aplicar Filtro.	

8.		Muestra las trazas existentes en el sistema que cumple con los criterios especificados, en una tabla que presenta las siguientes datos en las columnas como son: • Usuario • Sujeto • MS • CRD • Versión • Valor viejo • Valor nuevo • Tipo de acción • Fecha de acción Además brinda al final de la tabla las opciones: • Exportar (Ver descripción del caso de uso) • Imprimir (Ver descripción del caso de uso) En caso contrario ver Flujos alternos.
9.		El caso de uso termina.
Flujos alternos		
1 El usuario no especifica criterio de búsqueda		
Actor		**Sistema**
		2. Muestra todas las trazas del sistema. Termina el CU.
2 La búsqueda no arroja resultados		
		2. Muestra la lista de trazas vacía con un mensaje en la parte superior informando que no

		hay datos que respondan a los criterios de búsqueda específicados. Termina el caso de uso.
3 Selecciona la opción Limpiar Filtro		
		2. Limpia todos los campos seleccionados. Termina el caso de uso.
Relaciones	**CU Incluidos**	-
	CU Extendidos	Imprimir trazas, Exportar trazas

Se realizó el modelo conceptual del módulo Extracción obteniendo una mejor comprensión de los conceptos relacionados con el dominio, identificándose los actores que interactuarán con el mismo. La definición de las características funcionales a desarrollar, especificándose en términos de requerimientos funcionales y no funcionales siendo modelados gráficamente mediante un diagrama de casos de uso y mostrándose algunas de sus especificaciones, permitió asegurar que el software se corresponderá con las necesidades del cliente. Con el desarrollo de este capítulo quedaron sentadas las bases para facilitar y dar paso al diseño del módulo.

3 Capítulo 3: Diseño del módulo

En el presente capítulo se tiene como propósito traducir los requisitos a una especificación que describe cómo implementar el sistema, realizar una representación gráfica de los diagramas de clases del diseño y la descripción de las mismas, así como fundamentar la arquitectura utilizada.

3.1 Descripción de la arquitectura

La arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. Su objetivo principal es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. (36)

3.1.1 Arquitectura

La arquitectura utilizada es cliente/servidor, que es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor, al proceso que responde a las solicitudes. Es el modelo de interacción más común entre aplicaciones en una red. (37)

Cliente: Constituirá cualquier PC cliente instalada en el CIM, la cual contendrá un navegador web encargado de interactuar con el usuario y que se comunicará con el servidor de aplicación a través del protocolo http (en inglés *Hypertext Transfer Protocol*) o https (en inglés *Hypertext Transfer Protocol Secure*).

Servidor: Será un host que cumplirá los requerimientos mínimos establecidos para servidores. Tendrá instalado un servidor de aplicación (JBoss AS) que responderá las peticiones del cliente y se comunicará a través de la familia de protocolos TCP/IP (en inglés *Transmission Control Protocol/Internet Protocol*) con el host destinado para contener el SGBD. El servidor de aplicaciones accederá a los datos ubicados en la base de dato haciendo uso del API *Java Database Connectivity* (JDBC).

3.1.2 Patrón de arquitectura

Los patrones arquitectónicos son plantillas que describen los principios estructurales globales que construyen las distintas Arquitecturas de Software viables. Plantean una organización estructural fundamental para un sistema de software, expresando un conjunto de subsistemas predefinidos, especificando responsabilidades y organizando las relaciones entre ellos. La selección de un patrón arquitectónico es además una decisión fundamental de diseño cuando se desarrolla un sistema de software. (38)

La propuesta de solución presenta una arquitectura basada en el patrón arquitectónico Modelo-Vista-Controlador (MVC), el cual separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: (39)

- Modelo: el modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista: maneja la visualización de la información a través de las interfaces de usuario.
- Controlador: interpreta las acciones del usuario, informando al modelo y/o a la vista para que cambien según resulte apropiado.

De acuerdo con la tecnología a emplear para el desarrollo del módulo, el estilo MVC resulta distribuido de la siguiente forma: la **vista** está desarrollada básicamente con JSF, usando componentes Seam de interfaz de usuario, la librería de componentes RichFaces y Ajax4JSF, y Facelets como motor de plantillas, los cuales permiten el desarrollo de interfaces amigables en un período corto de tiempo. En el **controlador** se utiliza Seam como marco de trabajo de integración entre los componentes visuales y los de acceso a datos, además de JasperReports para la obtención de reportes en diversos formatos. Finalmente en el **modelo** se usa Hibernate como herramienta de mapeo objeto relacional y como librerías para la persistencia de los datos: EJB3 y JPA. Todas estas tecnologías están soportadas sobre JEE v2, plataforma que permite la programación de aplicaciones bajo el lenguaje de programación Java y JRE, que es un conjunto de utilidades que permite la ejecución de programas Java sobre todas las plataformas soportadas.

3.2 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tiene impacto en el sistema a considerar. Este artefacto sirve de abstracción y constituye la entrada fundamental de las actividades de implementación. (32)

3.2.1 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, que codifican buenos principios y sugerencias, siendo de gran importancia la captación de estos, si se quiere diseñar eficazmente un software orientado a objeto. Entre estos se encuentran los patrones GRASP (en inglés *General Responsibility Asignment Software Patterns*) los cuales describen los principios fundamentales para asignar responsabilidades a los objetos. Para el diseño del módulo se emplean los siguientes patrones GRASP:

Experto: este patrón plantea como solución asignar una responsabilidad al experto en información dentro del dominio, es decir la clase que cuenta con la información necesaria para cumplir una determinada responsabilidad. El mismo se evidencia en todas las clases construidas, ejemplo, las clases TrazaMS, y TrazaPaciente son las encargadas de obtener la información relacionada con trazas de momentos de seguimientos y pacientes respectivamente.

```
@Name("trazaMS")
@Scope(ScopeType.CONVERSATION)
public class TrazaMS extends Trazas{

    private Long idSujeto;//Solo se utiliza para conduccion

    private String idAccion;

    private Long idMs;

    private Long idCrd;

    public TrazaMS(String idUsuario,Long idSujeto,Long idMs,Long idCrd,String tipo

        setIdUsuario(idUsuario);

        this.idSujeto=idSujeto;
        this.idMs=idMs;
        this.idCrd=idCrd;

        setTipoAccion(tipoAccion);
        setFechaInicio(fechaInicio);
        setFechaFin(fechaFin);
   public TrazaListcustomExtraccion trazaMSMD(){
        String entityId="";

        if("Todos".equals(getIdUsuario())){
            setIdUsuario("");
        }

        //Condicion de Tipo Accion
        if("Crear_MS".equals(getTipoAccion())){

            idAccion="traza.auditLogEventTypeId='44'";

        }else{
                if("Eliminar_MS".equals(getTipoAccion())){

                    idAccion="traza.auditLogEventTypeId='45'";

                }else{

                    if("Modificar_MS".equals(getTipoAccion())){

                        idAccion="(traza.auditLogEventTypeId='49' or traza.auditLog
                    }
                    .
                    .
                    .
        TrazaListcustomExtraccion trazaMSMD=new TrazaListcustomExtraccion(consulta)

        return trazaMSMD;
```

Figura 4: Patrón experto en clase TrazaMS.

Creador: guía la asignación de responsabilidades al identificar quién debe ser el responsable de la creación de nuevos objetos o clases. Este patrón se utilizó en la clase GestionarTrazas ya que la misma

contiene un método que en dependencia del tipo de traza seleccionada que se quiera visualizar, crea un objeto de una clase relacionada con trazas según el tipo de traza seleccionada.

```
@SuppressWarnings("unchecked")
@Name("gestionarTrazas")
@Scope(ScopeType.CONVERSATION)
public class GestionarTrazas {

//@In(create=true,value="trazaList")
private TrazaList trazaFiltro;//=new TrazaList(entityManager);

private List<AuditLogEvent> lista=new ArrayList<AuditLogEvent>();

private String current_query="";

private String current_order="traza.userId asc";

private String lenguaje;

@In(create=true)
LocaleSelector localeSelector;

@In(create=true,value="trazaListcustomExtraccion")
private TrazaListcustomExtraccion trazaListcustomExtraccion=new TrazaListcustomExt

@In
EntityManager entityManager;
            .
            .
            .
public void aplicarFiltro(){
            .
            .
            .
  if(trazaFiltro.getSeccion().equals("conduccion"))
  {
      if("Insertar_Paciente".equals(trazaFiltro.getAccion()) || "Deshabilitar_Pac

      TrazaPaciente objetoTrazaPaci=new TrazaPaciente(trazaFiltro.getIdUsuario(),

      getTrazaListcustomExtraccion().setEjbql(objetoTrazaPaci.trazaPacientes().get

      //setTrazaListcustomExtraccion(objetoTrazaPaci.trazaPacientes());

      getTrazaListcustomExtraccion().setFirstResult(0);

      current_query=getTrazaListcustomExtraccion().getEjbql();

      }else{

          if("Mod_Cronograma_Esp".equals(trazaFiltro.getAccion())){

              TrazaCronogramaEs objetoTrazaCronoEs=new TrazaCronogramaEs(trazaFil
```

Figura 5: Patrón creador en clase GestionarTrazas.

Bajo acoplamiento: este patrón especifica que las clases deben presentar la menor dependencia posible entre ellas, ayudando esto a que si existiese una modificación en una clase, repercuta lo menos posible en

otras, además que sean fáciles de entender por separado y de reutilizar. El empleo de este patrón se evidencia en las diferentes clases que son utilizadas para la obtención de las trazas, pues existe cierta independencia en las mismas ya que cada una se encarga de adquirir información de un tipo de traza en específico.

```
@Name("trazaMS")
@Scope(ScopeType.CONVERSATION)
public class TrazaMS extends Trazas{

    private Long idSujeto;//Solo se utiliza para conduccion

    private String idAccion;

    private Long idMs;

    private Long idCrd;


    public TrazaMS(String idUsuario,Long idSujeto,Long idMs,Long idCrd,String tipo

        setIdUsuario(idUsuario);

        this.idSujeto=idSujeto;
        this.idMs=idMs;
        this.idCrd=idCrd;

        setTipoAccion(tipoAccion);
        setFechaInicio(fechaInicio);
        setFechaFin(fechaFin);
```

```
@SuppressWarnings("unchecked")
@Name("trazaPaciente")
@Scope(ScopeType.CONVERSATION)
public class TrazaPaciente extends Trazas {

    private Long idSujeto;

    private int idAccion;

    private Long idMs;


    public TrazaPaciente(String idUsuario,Long idSujeto,Long idMs,Long idCrd

    setIdUsuario(idUsuario);

    this.idSujeto=idSujeto;
    this.idMs=idMs;
    this.idCrd=idCrd;

    setTipoAccion(tipoAccion);
    setFechaInicio(fechaInicio);
    setFechaFin(fechaFin);
```

Figura 6: Patrón bajo acoplamiento en clases TrazaMS y TrazaPaciente.

Alta cohesión: plantea asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen

un trabajo grande. Este patrón se manifiesta en la clase CrearDataset debido a que presenta información coherente y relacionada con la misma. Esta clase contiene una lista de las hojas CRD pertenecientes al estudio activo, así como un objeto de conjunto de datos que será modificado en todo el proceso de creación de un conjunto de datos.

```
@SuppressWarnings("unchecked")
@Name("crearDataset")
@Scope(ScopeType.CONVERSATION)
public class CrearDataset {

    private Dataset conjunto = new Dataset();

    // Lista de Variables referenciadas
    private List<VariableHojaArbol> listaVariableHoja = new ArrayList<VariableHoj

    // Lista de hojas asociadas al grupo elegido
    private List<HojaCRDArbol> listaHojasArbol = new ArrayList<HojaCRDArbol>();

    private Date datestart;
    private Date dateend;

    private String listaSeleccionados = "";
    private List<Long> listaSeleccionados2=new ArrayList<Long>();
    private String itemsSeleccionadosMostrar="";

    private TreeNodeImpl<String> arbol=new TreeNodeImpl<String>();
    private int contArbol=0;

    private int cont = 0;
    private String seccionSeleccionada = "-";
```

Figura 7: Patrón alta cohesión en clase CrearDataset.

Controlador: es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Se evidencia en la clase GestionarTrazas debido a que interactúa con la interfaz de listar trazas y envía la información a la clase encargada de controlar los eventos según el tipo de traza a visualizar seleccionado en la interfaz mencionada anteriormente.

```
@SuppressWarnings("unchecked")
@Name("gestionarTrazas")
@Scope(ScopeType.CONVERSATION)
public class GestionarTrazas {

//@In(create=true,value="trazaList")
private TrazaList trazaFiltro;//=new TrazaList(entityManager);

private List<AuditLogEvent> lista=new ArrayList<AuditLogEvent>();

private String current_query="";

private String current_order="traza.userId asc";

private String lenguaje;

@In(create=true)
LocaleSelector localeSelector;

@In(create=true,value="trazaListcustomExtraccion")
private TrazaListcustomExtraccion trazaListcustomExtraccion=new TrazaListcustomExt

@In
EntityManager entityManager;
```

Figura 8: Patrón controlador en GestionarTrazas.

3.2.2 Diagrama de clases del diseño

Los diagramas de clases se realizan por cada caso de uso y muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Seguidamente se muestran algunos elementos que componen cada uno de estas representaciones y se ilustran dos ejemplos de los diagramas de clases del diseño confeccionados con la descripción de las clases que los conforman, el resto se podrán encontrar en el documento: SES - SIGEC - 010215_MDI_Extracción_v1.1.

Tabla 5: Estereotipos web para las clases de diseño.

 **JSF(Servlet)**	 ***.xhtml**	 **form**
Representa la clase que tiene código que se ejecuta en el servidor, la cual da respuesta a	La página cliente *.xhtml es la interfaz a través de la cual interactúan los usuarios con el	A través del formulario se envían datos del cliente al servidor.

las peticiones generadas en la vista.	sistema.	

La estructura general de los diagramas de clases del diseño estará compuesta por páginas clientes que son construidas por páginas servidoras y que a su vez contienen formularios que muestran y envían al servidor los datos para ser procesados los pedidos. Estas páginas clientes utilizan un conjunto de librerías basadas en el marco de trabajo JSF que permiten su construcción, además de que incluyen un conjunto de validaciones JavaScript que le proporcionan dinamismo a las mismas.

El controlador de JSF (servlet) invoca al controlador de SEAM (servlet) el cual enruta las peticiones para cada bean (paginas servidoras). Los bean invocan a funciones en la clase controladora que pueden modificar las entidades para darle respuesta a la petición solicitada, creando instancias de ellas para persistir la información a través de la interfaz Entity Manager contenida en el motor de persistencia JPA, el cual es utilizado por el marco de trabajo: EJB, facilitando el desarrollo de la lógica de negocio. Las entidades también emplean el ORM Hibernate para realizar el mapeo objeto relacional desde las clases Java a las tablas de las bases de dato, facilitando las consultas y recuperación de datos.

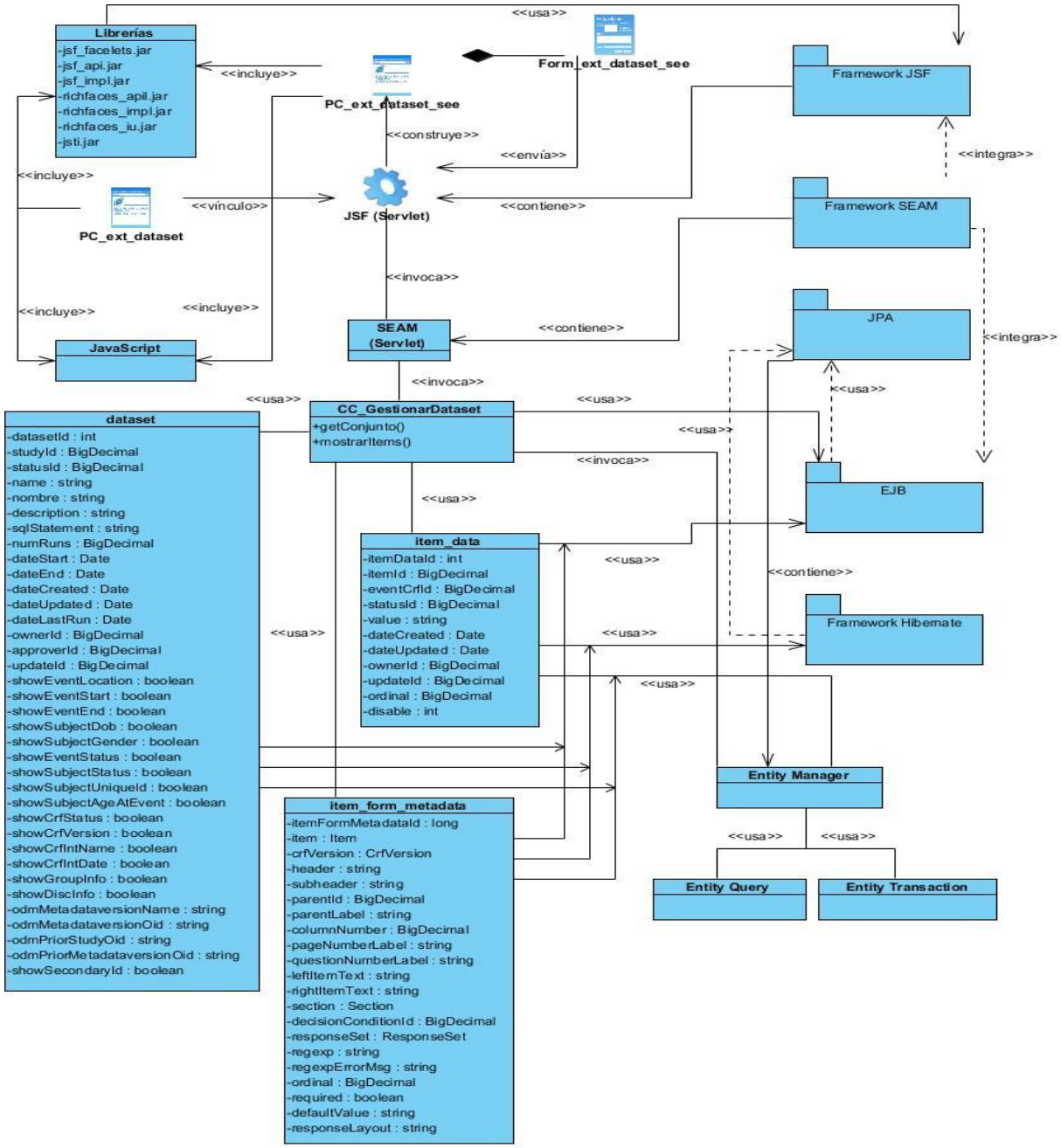

Figura 9: Diagrama de clases del diseño del caso de uso ¨Ver datos de un conjunto de datos¨.

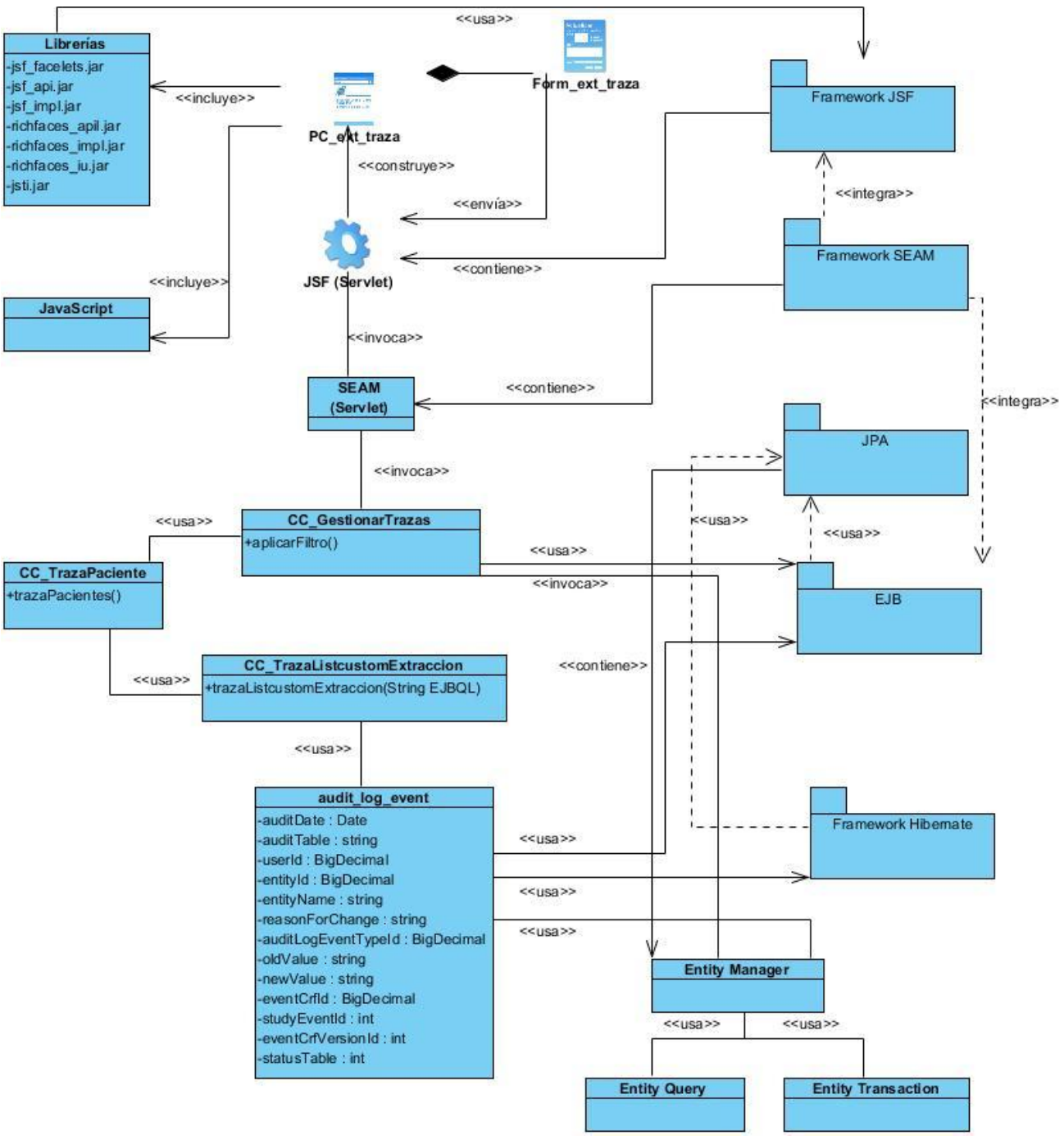

Figura 10: Diagrama de clases del diseño del caso de uso ¨Mostrar trazas de la funcionalidad insertar paciente¨.

3.2.3 Descripción de las clases.

Seguidamente se presentan las descripciones de las clases que contienen los diagramas de clases del diseño presentados anteriormente.

- **Vistas**

Tabla 6: Descripción de la vista ver conjunto de datos.

Nombre: ext_dataset_see	
Tipo de clase: Vista	
Descripción:	Permite ver los datos que conforman los conjuntos de datos creados en el sistema.

Tabla 7: Descripción de la vista gestionar conjunto de datos.

Nombre: ext_dataset	
Tipo de clase: Vista	
Descripción:	Permite visualizar las trazas de las acciones realizadas en los diferentes módulos del sistema.

Tabla 8: Descripción de la vista gestionar trazas.

Nombre: ext_traza	
Tipo de clase: Vista	
Descripción:	Muestra una interfaz que contiene los diferentes criterios de selección posibles para visualizar las trazas realizadas en el sistema, listándose las mismas.

- **Controladoras**

Tabla 9: Descripción de la clase GestionarDataset.

Nombre: GestionarDataset	
Tipo de clase: Controladora	
Atributo	**Tipo**
conjunto	Dataset
listaVariableHoja	List<VariableHojaArbol>
ver	Integer
trazaFiltro	TrazaList
Para cada responsabilidad:	
Nombre:	getConjunto ()
Descripción:	Retorna una instancia de la entidad Dataset.
Nombre:	mostrarItems ()
Descripción:	Devuelve la lista de variables de un conjunto seleccionado.
Nombre:	verMas ()
Descripción:	Permite visualizar más información del listado de conjunto de datos.
Nombre:	mostrarMisCD ()
Descripción:	Muestra el listado de conjuntos de datos creados por el usuario autenticado.
Nombre:	deleteDataset ()
Descripción:	Elimina el conjunto de datos seleccionado.

Tabla 10: Descripción de la clase TrazaPaciente.

Nombre: TrazaPaciente	
Tipo de clase: Controladora	
Atributo	**Tipo**
usuario	String
tipo_accion	String

fecha_inicio	Timestamp
fecha_fin	Timestamp
id_paciente	Long
id_ms	Long
id_crd	Long
id_accion	Long
Para cada responsabilidad:	
Nombre:	trazaPacientes ()
Descripción:	Retorna una lista de pacientes que se han introducido en un estudio teniendo en cuenta los criterios de búsqueda: usuarios, id_sujeto, id_ms, id_crd, fecha inicio y fin seleccionados en el formulario de trazas.

Tabla 11: Descripción de la clase GestionarTrazas.

Nombre: GestionarTrazas	
Tipo de clase: Controladora	
Atributo	**Tipo**
traza_filtro	TrazaList
traza_list_custom	TrazaListcustom_extraccion
Para cada responsabilidad:	
Nombre:	aplicarFiltro ()
Descripción:	Método encargado de invocar a la clase de Trazas relacionada con el tipo de acción seleccionada en la interfaz de trazas.

Tabla 12: Descripción de la clase TrazaListcustomExtraccion.

Nombre: TrazaListcustomExtraccion

Tipo de clase: Controladora	
Atributo	**Tipo**
EJBQL	String
pagina	Integer
Para cada responsabilidad:	
Nombre:	trazaListcustomExtraccion (String EJBQL)
Descripción:	Constructor de la clase TrazaListcustom_extraccion que permite la obtención de una lista de trazas a través de consulta que recibe por parámetro.

Con la realización de este capítulo se obtuvo el modelo de diseño de cada caso de uso, se generaron los diagramas de clases del diseño y la descripción de cada uno de estos lográndose, de esta forma, transformar los requisitos funcionales a un lenguaje de diseño que sirviendo de guía para la implementación del módulo Extracción. Igualmente se plasmó como patrón arquitectónico a utilizar: el MVC, permitiendo dividir la lógica de negocio en las capas de presentación, lógica del negocio y modelado, haciendo la solución más escalable.

4 Capítulo 4: Implementación

En este capítulo se define cómo desarrollar la arquitectura prevista durante el diseño y se implementan las clases definidas en el capítulo anterior en términos de componentes. Se construye el Modelo de datos y se describen cada una de las tablas que lo forman. También se modelan los diagramas de despliegue y de componentes, quedando conformado el modelo de implementación del sistema, dando una visión de cómo quedará construida y distribuida la aplicación. Además, se proporciona una detallada explicación de los estándares de diseño, codificación, así como el tratamiento de errores utilizados.

4.1 Modelo de datos

En la informática, un modelo de datos se centra en el planeamiento del desarrollo de aplicaciones y la decisión de cómo se almacenarán los datos y cómo se accederá a ellos. Los modelos de datos determinan la estructura de la información, con el objetivo de mejorar la comunicación y la precisión en aplicaciones que usan e intercambian datos. (40)

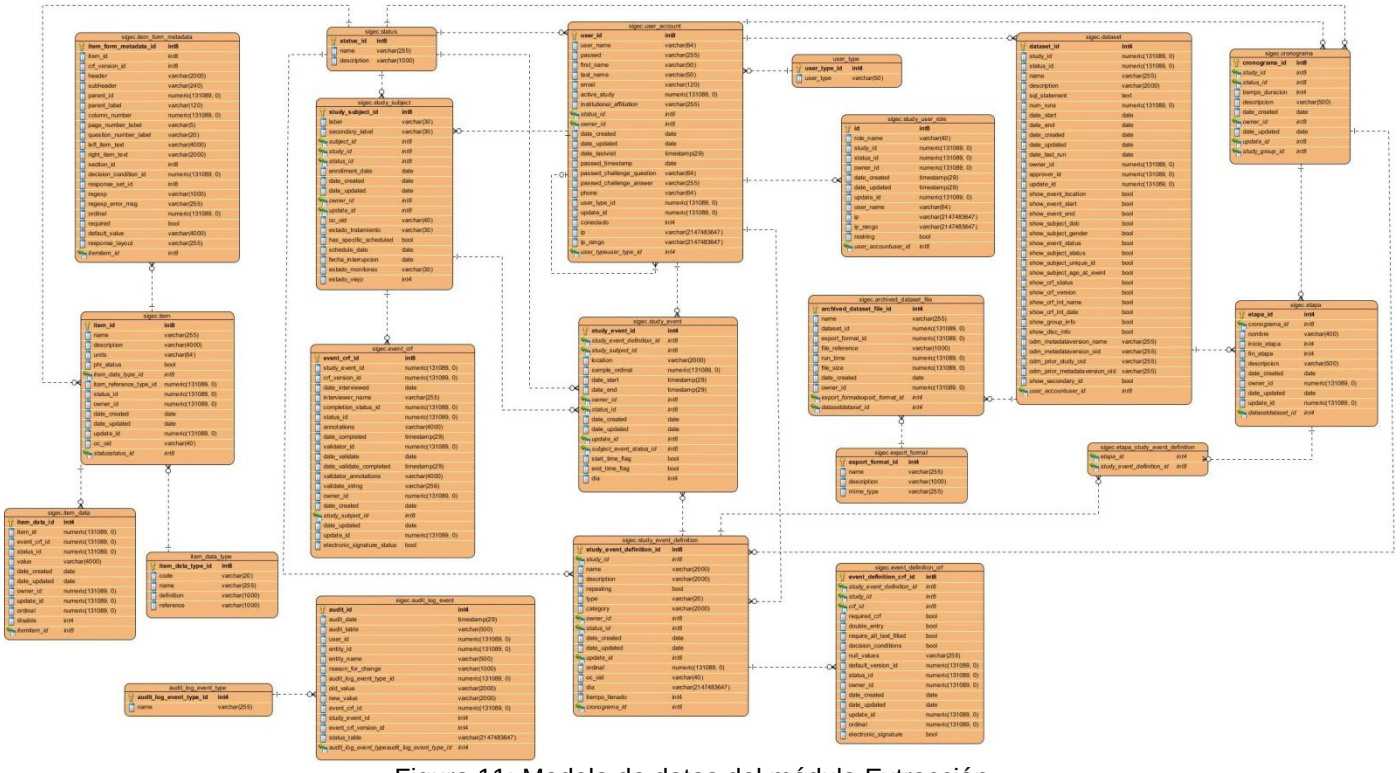

Figura 11: Modelo de datos del módulo Extracción.

A continuación se muestran algunas de las descripciones de las tablas del modelo de datos, las restantes descripciones se encontrarán los anexos.

Tabla 13: Descripción de la tabla dataset.

Nombre: dataset		
Descripción: Almacena conjuntos de datos.		
Atributo	**Tipo**	**Descripción**
datasetId	Integer	Identificador del conjunto de datos.
studyId	BigDecimal	Identificador del estudio al cual pertenece el conjunto de datos.
statusId	BigDecimal	Estado del conjunto de datos.
name	String	Nombre del conjunto de datos.
description	String	Descripción del conjunto de datos.
sqlStatement	String	Almacena las variables que conforman el conjunto de datos.
numRuns	BigDecimal	
dateStart	Date	Fecha de inicio del conjunto de datos.
dateEnd	Date	Fecha final del conjunto de datos.
dateCreated	Date	Fecha de creación del conjunto de datos.
dateUpdated	Date	Fecha de actualización del conjunto de datos.
dateLastRun	Date	Fecha de la última ejecución del conjunto de datos.
ownerId	BigDecimal	Identificador del usuario que creó el conjunto de datos.

approverId	BigDecimal	Identificador del usuario que aprobó el conjunto de datos.
updateId	BigDecimal	Identificador del usuario que actualizó el conjunto de datos.
showEventLocation	Boolean	Incluir localización de los ms que conforman el conjunto de datos.
showEventStart	Boolean	Incluir fecha inicio de los ms que conforman el conjunto de datos.
showEventEnd	Boolean	Incluir fecha final de los ms que conforman el conjunto de datos.
showSubjectDob	Boolean	Incluir fecha de nacimiento de los sujetos que forman parte del conjunto de datos.
showSubjectGender	Boolean	Incluir género de los sujetos que forman parte del conjunto de datos.
showEventStatus	Boolean	Incluir estado de los ms que conforman el conjunto de datos.
showSubjectStatus	Boolean	Incluir estado de los sujetos que forman parte del conjunto de datos.
showSubjectUniqueId	Boolean	Incluir identificador de los sujetos que forman parte del conjunto de datos.
showSubjectAgeAtEvent	Boolean	Incluir edad de los sujetos en el MS.
showCrfStatus	Boolean	Incluir estado del CRD.
showCrfVersion	Boolean	Incluir versión del CRD.
showCrfIntName	Boolean	Incluir nombre del entrevistador

		que registró la información en el CRD.
showCrfIntDate	Boolean	Incluir la fecha en que se recogieron los datos en el CRD.

Tabla 14: Descripción de la tabla item_data.

Nombre: item_data		
Descripción: Entidad que contiene los valores asociados a las “variables” de una “hoja de CRD”.		
Atributo	**Tipo**	**Descripción**
itemDataId	Integer	Identificador de la tabla.
itemId	BigDecimal	Identificador de la variable.
eventCrfId	BigDecimal	Identificador de la tabla que contiene el MS y CRD a los cuales pertenece la variable.
statusId	BigDecimal	Estado de la variable en el MS.
value	String	Valor de la variable.
dateCreated	Date	Fecha de creación de los datos introducidos en la variable.
dateUpdated	Date	Fecha de actualización de los datos introducidos en la variable.
ownerId	BigDecimal	Identificador del usuario que insertó la información en la variable.
updateId	BigDecimal	Identificador del usuario que actualizó la información en la variable.

Tabla 15: Descripción de la tabla audit_log_event.

Nombre: audit_log_event		
Descripción: Contiene las trazas de los eventos ocurridos sobre varias tablas de la base de datos.		
Atributo	**Tipo**	**Descripción**
auditDate	Date	Fecha en que ocurrió el evento.
auditTable	String	Tabla donde ocurrió el evento.
userId	BigDecimal	Usuario que generó el evento.
entityId	BigDecimal	Identificador de la tupla de la tabla donde se genero el evento.
entityName	String	Nombre de la entidad.
reasonForChange	String	Razón por la que ocurrió el cambio.
auditLogEventTypeId	String	Identificador del tipo de evento.
oldValue	String	Valor antiguo de la variable.
newValue	String	Nuevo valor de la variable.
eventCrfId	BigDecimal	Identificador de la tupla de la tabla MS-CRD donde ocurrió el evento.
studyEventId	Integer	Identificador del MS donde ocurrió el evento.
eventCrfVersionId	Integer	Identificador de la versión del CRD en la tabla MS-CRD donde ocurrió el evento.
statusTable	Integer	Estado de la tabla.

4.2 Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se puede encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la

relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Un diagrama de implementación muestra las dependencias entre las partes de código del sistema (diagramas de componentes) y la estructura del sistema en ejecución (diagrama de despliegue). (41)

4.2.1 Diagrama de despliegue

Es un diagrama que se utiliza para modelar el hardware empleado en las implementaciones de sistemas y las relaciones entre sus componentes. Describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. (42)

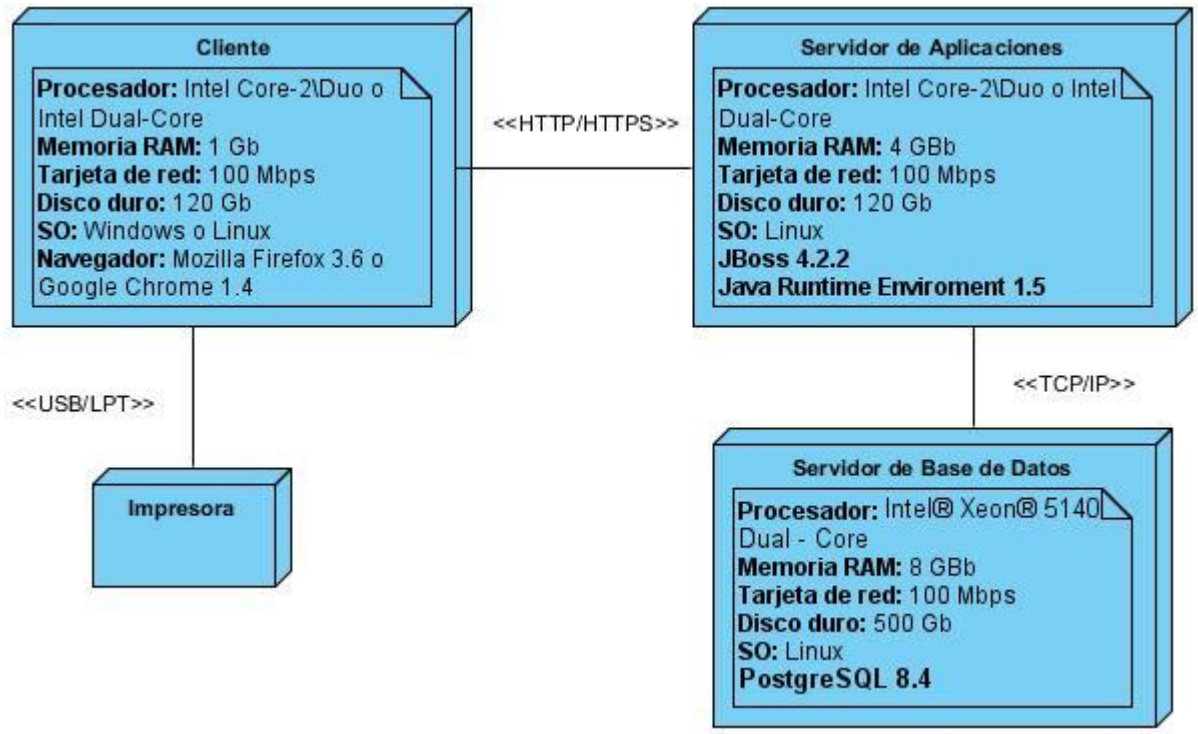

Figura 12: Diagrama de despliegue.

4.2.2 Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Algunos estereotipos estándar de componentes presentes, en los diagramas que se muestran posteriormente son: <<library>> es una librería estática o dinámica, <<table>> es una tabla de base de datos. (32)

Los distintos componentes pueden agruparse en paquetes según un criterio lógico y con vista a simplificar la implementación. Estos paquetes son estereotipados como <<subsistemas>>. Cada subsistema puede contener componentes y otros subsistemas. A continuación se presentan los diagramas de componentes del módulo Extracción, conformado por los subsistemas de implementación de la vista, modelo y controlador.

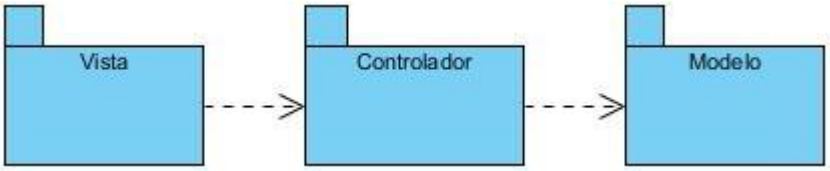

Figura 13: Subsistema de implementación.

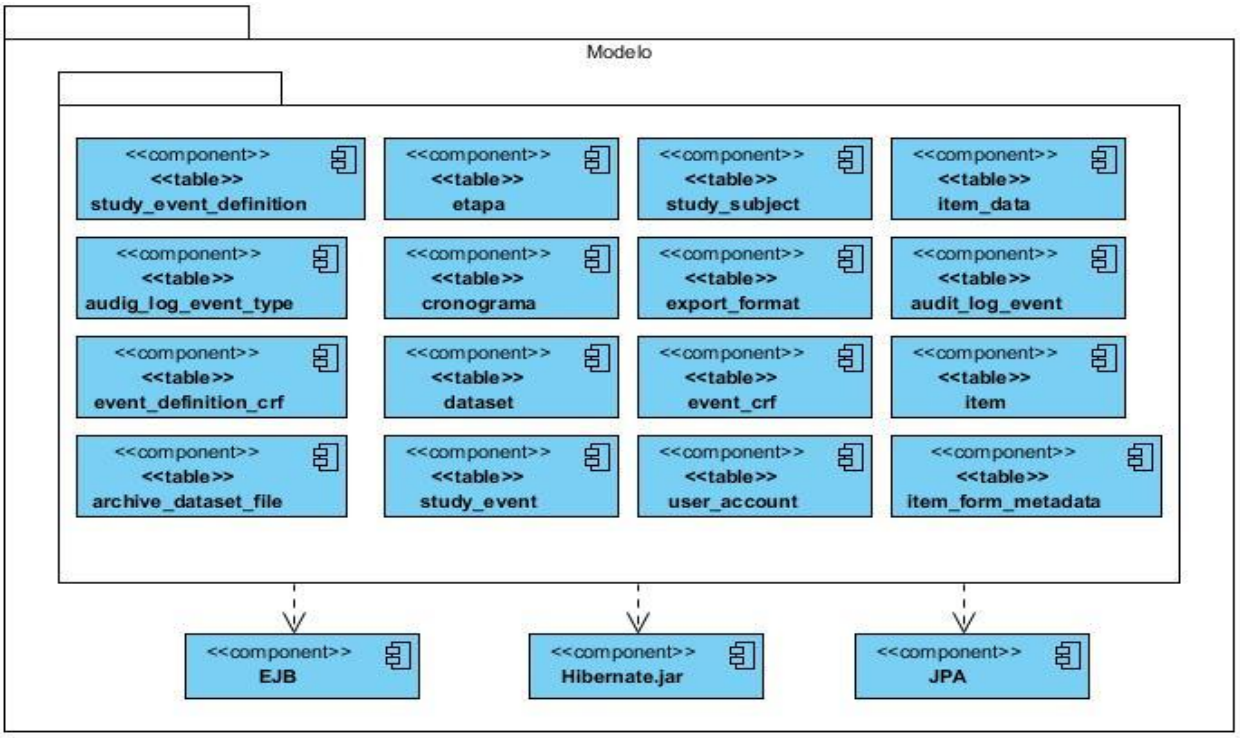

Figura 14: Diagrama de componentes del subsistema: Modelo.

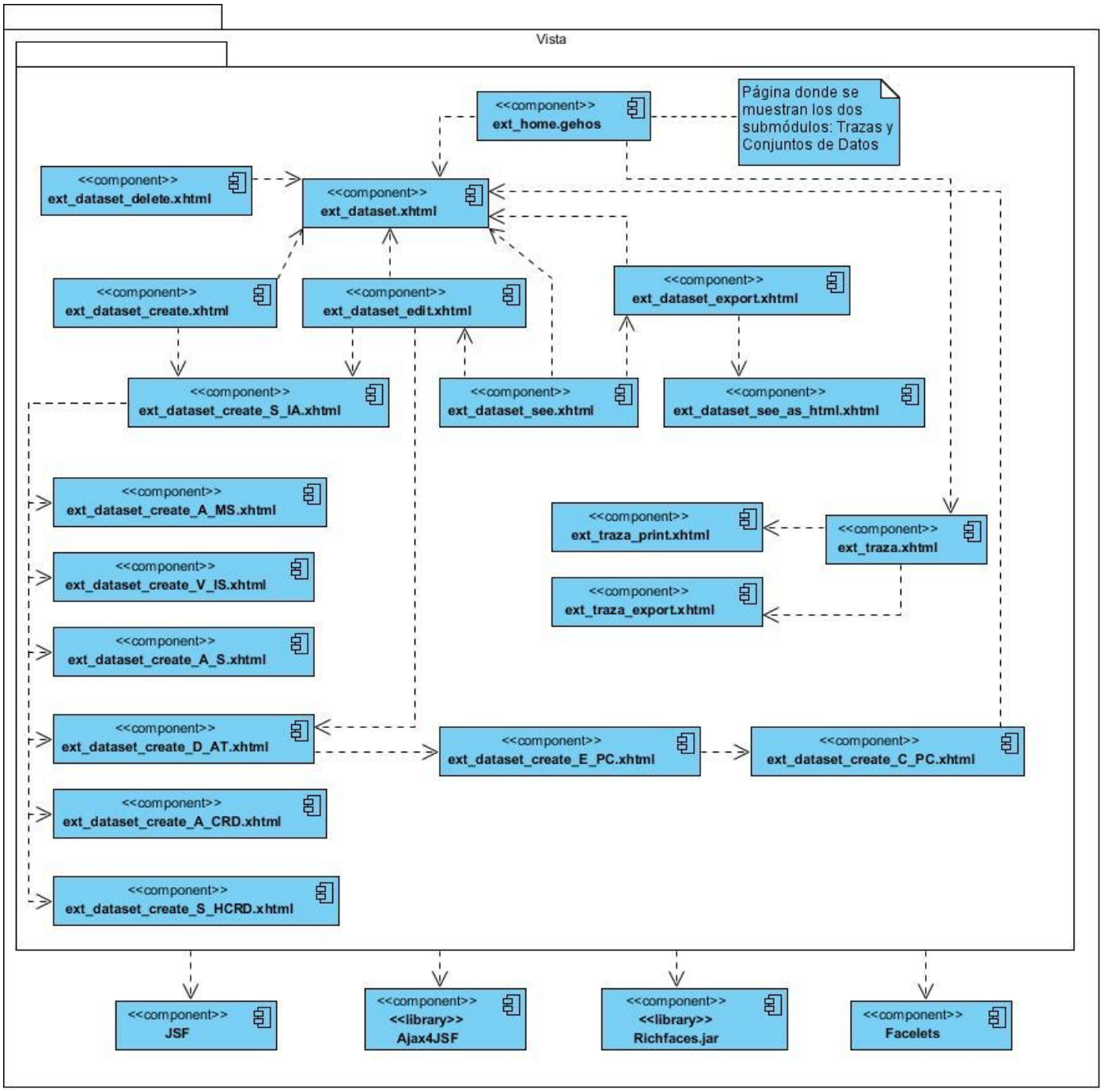

Figura 15: Diagrama de componentes del subsistema: Vista.

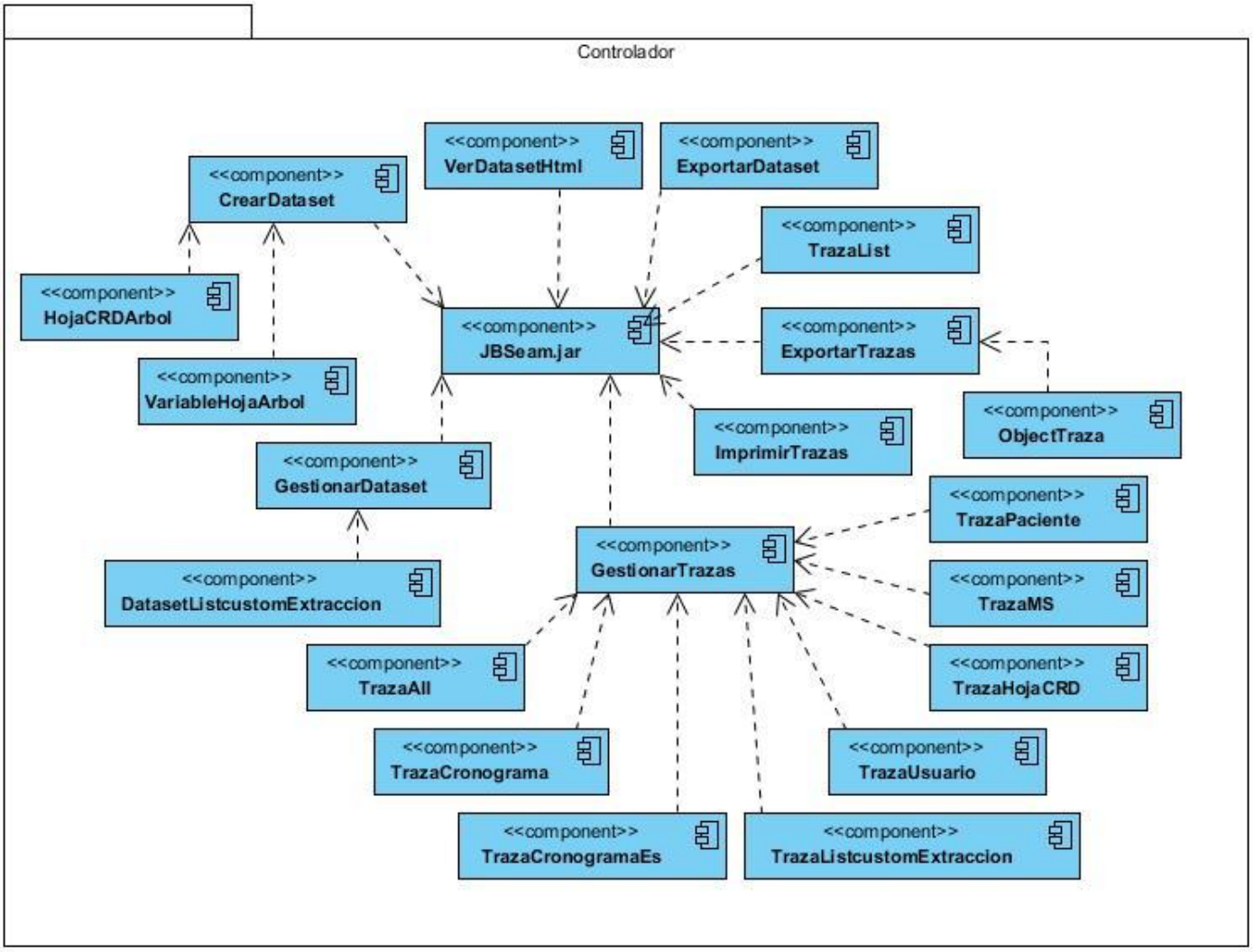

Figura 16: Diagrama de componentes del subsistema: Controlador.

4.3 Tratamiento de errores

En las funcionalidades desarrolladas para la solución propuesta, se tiene en cuenta el tratamiento de todos los posibles errores que puedan ocurrir, para garantizar así la integridad y confiabilidad de los datos. Se muestran mensajes de advertencia que son de fácil comprensión para el usuario, pudiéndose observar cuando se define incorrectamente el rango de fecha de matrícula de los sujetos en el estudio a incluir en el conjunto de datos. En las consultas realizadas a la base de datos para la visualización de información

relacionada con trazas y realización de reportes se hace uso de la sentencia try-catch evitando comportamientos inesperados en el proceso de adquisición de los datos. Se realizan validaciones en el código para la consistencia de los datos entrados por el usuario. Además para evitar la creación de una nueva versión del sistema en cada idioma se utiliza las capacidades de internacionalización que brinda JSF, el cual permite definir en archivos o ficheros de propiedades los mensajes de respuesta a cada una de las excepciones en los idiomas que soporta el sistema.

4.4 Seguridad

Se puede entender la seguridad como la necesidad de proteger. En una red se deben proteger todos los equipos que posibilitan el proceso de la comunicación, las personas que producen, acceden y distribuyen los datos y finalmente la información que es considerada como uno de los activos más importantes de las organizaciones. Para mantener segura la información que viaja a través de la red esta debe cumplir con tres requisitos: (43)

- Integridad: Requiere que los recursos sean modificados por quienes están autorizados y que los métodos y los procesamientos de la información sean salvaguardados en su totalidad y con exactitud.
- Confidencialidad: Se debe garantizar que la información sea accesible solo por quienes están autorizados para su lectura, cambios, impresión y formas de revelación.
- Disponibilidad: Se requiere que la información esté disponible en el momento exacto para quienes están autorizados a acceder a ella.

Para garantizar la integridad de la información en el módulo de Extracción y evitar diferentes ataques como inyecciones HQL (en inglés *Hibernate Query Language*) se hace uso del ORM Hibernate, el cual ofrece la posibilidad de realizar consultas parametrizadas. La confidencialidad y disponibilidad de la información será responsabilidad del módulo Administración del SIGEC, el cual implementará una jerarquía de accesos para los diferentes usuarios del sistema, los cuales en dependencia del rol que desempeñe dentro este, tendrá los privilegios sobre las posibles acciones a realizar. Además sólo podrán acceder a la aplicación a través de direcciones IP específicas y bien controladas, y todo cambio o modificación en el sistema debe ser atribuible a un usuario particular según su autenticación, auditando todas las acciones realizadas.

4.5 Estrategias de codificación. Estándares y estilos a utilizar

Al comenzar un proyecto de software, es de gran importancia el establecimiento de un estándar de codificación para asegurar que todos los programadores trabajen de forma coordinada, siguiendo un conjunto de reglas para la escritura del código fuente. El uso del mismo permite una mejor comunicación entre los programadores facilitándoles el entendimiento del código, además crea condiciones para la reusabilidad y el mantenimiento de los sistemas. También la definición de pautas para el diseño garantiza la uniformidad del producto en todos sus módulos, ofreciendo una aplicación más atractiva e intuitiva para el usuario.

Se identificaron como estrategias de codificación a emplear para la implementación del módulo Extracción las que se explican a continuación.

Estrategias de codificación

El inicio y fin de los bloques de código deben ser de dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción.

Los comentarios de implementación se ubicarán al inicio de la clase o función especificando el objetivo de la misma, delimitados por /*...*/, y //.

Para los nombres de las funciones se debe utilizar verbos que denoten la acción que realiza la función. Ejemplo: imprimirTrazas (). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set. Ejemplo: getConjunto (). Además se aplicará el estilo camelCase en los nombres de las clases, atributos, funciones cumpliendo que la primera letra del identificador esté en minúscula y la primera letra de las siguientes palabras, si contiene, concatenadas en mayúscula como los ejemplos presentados anteriormente.

Los nombres de las tablas deben escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizará underscoard para separarlo. (Ejemplo: dataset, audig_log_event).

Estándares y estilos

En el documento Pautas_Diseño_AplicacionesWEB_CESIM se encuentran los estándares y estilos definidos por el CESIM, los cuales serán aplicados al módulo Extracción.

En este capítulo quedaron conformados el modelo de datos y las descripciones de las tablas que componen al mismo permitiendo describir los elementos que intervienen en el dominio problema y la forma en que se relacionan entre sí. La elaboración del diagrama de componentes posibilitó una mejor comprensión de la solución a implementar y el diagrama de despliegue, visualizar las dependencias entre los diferentes elementos que lo componen. Además, se detallan los estándares de diseño, codificación, tratamiento de errores, y cómo se garantiza la seguridad en el módulo, aspectos importantes que contribuyen a lograr una solución con la calidad requerida. Finalmente se implementaron todas las clases y métodos necesarios, logrando la obtención del módulo Extracción el cual cumple con los requerimientos del cliente.

Conclusiones generales

Con la presente investigación se logró el desarrollo las tareas propuestas dando cumplimento al objetivo general planteado, por lo que se pueden arribar a las siguientes conclusiones:

- Los sistemas informáticos relacionados con la extracción de datos en ensayos clínicos analizados a nivel internacional, demostraron que no cumplen con todas las necesidades del CIM, no adaptándose a las características del Sistema Nacional de Salud cubano; ratificándose el desarrollo del módulo Extracción para el Sistema de Gestión de Ensayos Clínicos.
- El estudio realizado al software Clínicas 1.0 a nivel nacional, sirvió de base para el desarrollo del módulo Extracción, arrojando como resultado, un conjunto de funcionalidades que sustentaron el desarrollo de la solución propuesta.
- El identificarse los principales conceptos asociados al dominio de la investigación, así como las relaciones entre estos, posibilitó una mejor comprensión del mismo y constituyó el punto de partida para una adecuada selección de los requerimientos funcionales del módulo a desarrollar.
- Se generó la documentación necesaria de la investigación mediante la confección de los artefactos requeridos correspondientes a los flujos de trabajo indicados por la metodología de desarrollo de software RUP, utilizándose el Lenguaje Unificado de Modelado (UML), y Visual Paradigm for UML 8.0 como herramienta CASE; sirviendo de guía al desarrollador para a la implementación.
- Fueron implementadas las funcionalidades propuestas utilizándose como herramientas y tecnologías las definidas en el departamento SES para el desarrollo de sus aplicaciones como son: el gestor de base de datos PostgreSQL, el lenguaje de programación Java, el servidor de aplicaciones JBoos AS, el framework JBoss Seam y el IDE Eclipse.
- Se aplicaron las pautas de diseño a la solución, definidas en el centro CESIM, garantizando la uniformidad y homogeneidad en el sistema SIGEC, y el uso de patrones de diseño incorporando buenas prácticas al resultado de la investigación.

Recomendaciones

Al concluir el presente trabajo de diploma, considerando alcanzados los objetivos trazados en el mismo, se recomienda:

- Incorporar al submódulo de gestión de trazas la posibilidad de visualizar las trazas de los pacientes habilitados, los cronogramas específicos creados, valores de las variables actualizadas, eliminadas y las acciones realizadas sobre los conjuntos de datos.
- Permitir a través de las trazas la recuperación de información, teniendo en cuenta que una de las políticas de trabajo en la base de datos del SIGEC es la no eliminación de los datos.

Referencias bibliográficas

1. **Ilustre Colegio de Médicos de Madrid.** S.E.F.C. Sociedad Española de Farmacología Clínica. [En línea] [Citado el: 23 de octubre de 2013.] http://www.se-fc.org/gestor/index.php/ensayos-clinicos/informacion-general.

2. **Pascual López, María Amparo, y otros.** Scielo. [En línea] 2011. [Citado el: 20 de febrero de 2014.] http://scielo.sld.cu/pdf/far/v45n1/far02111.pdf.

3. **Texas Heart Institute.** Texas Heart Institute. [En línea] agosto de 2012. [Citado el: 21 de octubre de 2013.] http://www.texasheartinstitute.org/HIC/Topics_Esp/FAQ/clinical_trials_span.cfm.

4. **Lic. Yoselyn Díaz Rodríguez. Departamento de Fuentes de Información. Infomed-Centro Nacional de Información de Ciencias Médicas, Ministerio de Salud Pública .** Directorio de Instituciones en Salud. [En línea] 7 de junio de 2012. [Citado el: 20 de noviembre de 2013.] http://dirinstituciones.sld.cu/index.php?P=FullRecord&ID=321(Infomed-Directorio de Instituciones).

5. **Centro de Ingeniería Genética y Biotecnología.** Centro de Ingeniería Genética y Biotecnología. [En línea] 25 de junio de 2012. [Citado el: 20 de noviembre de 2013.] http://www.cigb.edu.cu/.

6. **UEB Web CITMATEL.** Centro de Inmunología Molecular. [En línea] 2013. [Citado el: 20 de noviembre de 2013.] http://www.cim.co.cu/cim.php.

7. **Facultad de Ciencias Médicas de Cienfuegos.** Revista de Ciencias Médicas de Cienfuegos. [En línea] 2013. [Citado el: 20 de noviembre de 2013.] http://www.medisur.sld.cu/index.php/medisur/article/view/2316/1151..

8. **OpenClinica, LLC.** OpenClinica: Open Source for Clinical Research. [En línea] 2014. [Citado el: 10 de febrero de 2014.] https://www.openclinica.com/product-features.

9. **PAREXEL International Corporation.** Perceptive Informatics. [En línea] 2014. [Citado el: 10 de febrero de 2014.] http://www.parexel.com/solutions/informatics/study-management-monitoring/impact-ctms/.

10. **Regents of the University of Minnesota.** Clinical and Translational Science Institute. [En línea] 4 de septiembre de 2013. [Citado el: 10 de febrero de 2014.] http://www.ctsi.umn.edu/biomedical-informatics/clinical-trial-management-system.

11. **Rodríguez García, Lucía, y otros.** Revista Cubana de Informática Médica. [En línea] 25 de diciembre de 2011. [Citado el: 17 de enero de 2014.] http://www.rcim.sld.cu/revista_24/articulo_pdf/alasclinicas.pdf.

12. **Ecured.** Enciclopedia Cubana. [En línea] [Citado el: 18 de febrero de 2014.] http://www.ecured.cu/index.php/Servidor_web.

13. **Juntao Yuan, Michael y Heute, Thomas.** *JBoss Seam: Simplicity and Power Beyond Java EE.* s.l. : Prentice Hall, 2007. ISBN: 9780137129393.

14. **Pérez, Javier Eguiluz.** Libros Web. [En línea] 14 de marzo de 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/xhtml/capitulo_1.html..

15. —. Libros Web. [En línea] 2013. [Citado el: 12 de febrero de 2014.] http://librosweb.es/css/capitulo_1.html.

16. **JavaScript Ya.** JavaScript Ya. [En línea] [Citado el: 18 de febrero de 2014.] http://www.javascriptya.com.ar/temarios/descripcion.php?cod=2.

17. **Ajax Ya.** Ajax Ya. [En línea] [Citado el: 12 de febrero de 2014.] http://www.ajaxya.com.ar/.

18. **Pérez, Javier Eguiluz.** Libros Web. [En línea] 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/ajax/capitulo_1.html.

19. **Definicion De.** Definicion De. [En línea] 2014. [Citado el: 10 de febrero de 2014.] http://definicion.de/java/.

20. **Jaspersoft Community.** Jaspersoft Community. *Open Source Java Reporting Library.* [En línea] 2014. [Citado el: 13 de Mayo de 2014.] http://community.jaspersoft.com/project/jasperreports-library.

21. **Tong, Kent Ka Iok.** *Beginning JSF™ 2 APIs and JBoss® Seam.* s.l. : Apress, 2009. ISBN: 978-1-4302-1923-1.

22. **JBoss.org.** JBoss.org. [En línea] 2013. [Citado el: 13 de febrero de 2014.] http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/pdf/richfaces_reference.pdf..

23. **Salter, David.** *Seam 2.x Web Development.* Olton Birmingham : Packt Publishing, 2009. ISBN: 978-1-847195-92-0.

24. **Oracle and/or its affiliates.** Oracle Documentation. [En línea] enero de 2013. [Citado el: 17 de febrero de 2014.] http://docs.oracle.com/javaee/6/tutorial/doc/gijtu.html. 821-1841-16.

25. **Linwood, Jeff y Minter, Dave.** *Beginning Hibernate. Second Edition.* New York : Apress, 2010. ISBN: 978-1-4302-2851-6.

26. **Yuan, Michael Juntao, Orshalick, Jacob y Heute, Thomas.** *Seam Framework. Experience the Evolution of JavaTM EE. Second Edition.* Boston : Prentice Hall, 2009. ISBN: 978-0-13-712939-3.

27. **López, Carlos Llorente.** Archivo Abierto Institucional de la Universidad Rey Juan Carlos. [En línea] 18 de junio de 2010. [Citado el: 13 de febrero de 2014.] http://eciencia.urjc.es/bitstream/10115/4111/1/pfcCarlosLlorenteLopez.pdf.

28. **PostgreSQL.** PostgreSQL. [En línea] 2013. [Citado el: 10 de febrero de 2014.] http://www.postgresql.org.es/comunidad.

29. **Guia Ubuntu.** Guia Ubuntu. [En línea] 2008. [Citado el: 13 de febrero de 2014.] http://www.guia-ubuntu.com/index.php/PgAdmin_III.

30. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje unificado de modelado. Manual de Referencia.* s.l. : Eddison Wesley, 2007. ISBN: 9788478290871.

31. **Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia.** Entorno Virtual de Aprendizaje. [En línea] [Citado el: 17 de febrero de 2014.] http://eva2.uci.cu/mod/folder/view.php?id=902.

32. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de desarrollo de software.* España : Eddison Wesley, 2000. ISBN: 84-7829-036-2.

33. **Sommerville, Ian.** *Ingeniería del Software.* Madrid : Adisson Wesley, 2005. ISBN: 84-7929-074-5.

34. **Quintana Rondón, Yoandri, Camejo Domínguez, Lianet y Díaz Berenguer, Abel.** Investigación | FIUBA. Facultad de ingeniería Universidad de Buenos Aires. [En línea] 2011. [Citado el: 10 de febrero de 2014.] http://laboratorios.fi.uba.ar/lie/Revista/Articulos/080815/A3mar2011.pdf.

35. **Larman, Craig.** *UML Y PATRONES Introducción al analisis y diseño orientado a objetos. .* México : PRENTICE HALL, 1999. ISBN: 970-1 7-0261-1.

36. **Casanova, Josep.** Desarrolloweb.com. [En línea] 9 de Septiembre de 2004. [Citado el: 22 de Mayo de 2014.] http://www.desarrolloweb.com/articulos/1622.php.

37. **Fletes Gudiño, Pedro, Hernández Barbosa, Oscar Daniel y Benavides Delgado, J. Reyes.** Instituto Tecnológico de Colima. *Tutorial de fundamentos de bases de datos.* [En línea] 2007. [Citado el: 20 de Mayo de 2014.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u1_6.htm.

38. **Sandra Almeira, Adriana, Perez Cavenago, Vanina y Beatriz Rosanigo, Zulema.** Departamento de Informática Sede Trelew (DIT). *Arquitectura de Software: Estilos y Patrones.* [En línea] Marzo de 2007. [Citado el: 18 de Mayo de 2014.] http://www.roa.unp.edu.ar:8080/graduate/bitstream/123456789/203/1/Tesina%20Arquitectura%20de%20Soft.pdf.

39. **Reynoso, Carlos y Kiccillof, Nicolás.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 2004. [Citado el: 24 de marzo de 2014.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp.

40. **Definición de.** Definición de modelo de datos - Qué es, Significado y Concepto. [En línea] [Citado el: 21 de Abril de 2014.] http://definicion.de/modelo-de-datos/.

41. **Hernandez, Leovigilda.** Modelo de implementación. [En línea] 1 de Junio de 2013. [Citado el: 20 de Abril de 2014.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

42. **Ecured.** Enciclopedia Cubana . [En línea] Enero de 2012. [Citado el: 18 de Abril de 2014.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

43. *LA SEGURIDAD EN LAS REDES DE COMUNICACIONES.* **Alirio González, Jaime y Vanegas, Carlos Alberto.** 1, Bogotá : s.n., 2006, Vol. 3. ISSN: 1794-211X.

Bibliografía

Ajax Ya. Ajax Ya. [En línea] [Citado el: 12 de febrero de 2014.] http://www.ajaxya.com.ar/.

Casanova, Josep. Desarrolloweb.com. [En línea] 9 de Septiembre de 2004. [Citado el: 22 de Mayo de 2014.] http://www.desarrolloweb.com/articulos/1622.php.

Caules, Cecilio Álvarez. Arquitectura Java. [En línea] 2 de agosto de 2013. [Citado el: 18 de febrero de 2014.] http://www.arquitecturajava.com/introduccion-a-ejb-3-1-i/.

Centro de Ingeniería Genética y Biotecnología. Centro de Ingeniería Genética y Biotecnología. [En línea] 25 de junio de 2012. [Citado el: 20 de noviembre de 2013.] http://www.cigb.edu.cu/.

Definicion De. Definicion De. [En línea] 2014. [Citado el: 10 de febrero de 2014.] http://definicion.de/java/.

Definición de. Definición de modelo de datos - Qué es, Significado y Concepto. [En línea] [Citado el: 21 de Abril de 2014.] http://definicion.de/modelo-de-datos/.

Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. Entorno Virtual de Aprendizaje. [En línea] [Citado el: 17 de febrero de 2014.] http://eva2.uci.cu/mod/folder/view.php?id=902.

Ecured. Enciclopedia Cubana. [En línea] [Citado el: 18 de febrero de 2014.] http://www.ecured.cu/index.php/Servidor_web.

Ecured. Enciclopedia Cubana . [En línea] Enero de 2012. [Citado el: 18 de Abril de 2014.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

Facultad de Ciencias Médicas de Cienfuegos. Revista de Ciencias Médicas de Cienfuegos. [En línea] 2013. [Citado el: 20 de noviembre de 2013.] http://www.medisur.sld.cu/index.php/medisur/article/view/2316/1151.

Fletes Gudiño, Pedro, Hernández Barbosa, Oscar Daniel y Benavides Delgado, J. Reyes. Instituto Tecnológico de Colima. *Tutorial de fundamentos de bases de datos.* [En línea] 2007. [Citado el: 20 de Mayo de 2014.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u1_6.htm.

Guia Ubuntu. Guia Ubuntu. [En línea] 2008. [Citado el: 13 de febrero de 2014.] http://www.guia-ubuntu.com/index.php/PgAdmin_III.

Hernández León, Rolando Alfredo y Coello González, Sayda. *El Proceso de Investigación Científica.* Ciudad de La Habana : Editorial Universitaria, 2011. ISBN: 978-959-16-1307-3.

Hernandez, Leovigilda. Modelo de implementación. [En línea] 1 de Junio de 2013. [Citado el: 20 de Abril de 2014.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar. *Metodología de la Investigación.* s.l. : McGraw Hill, 2006. 970-10-5753-8.

Ilustre Colegio de Médicos de Madrid. S.E.F.C. Sociedad Española de Farmacología Clínica. [En línea] [Citado el: 23 de octubre de 2013.] http://www.se-fc.org/gestor/index.php/ensayos-clinicos/informacion-general.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de desarrollo de software.* España : Eddison Wesley, 2000. ISBN: 84-7829-036-2.

Jaspersoft Community. Jaspersoft Community. *Open Source Java Reporting Library.* [En línea] 2014. [Citado el: 13 de Mayo de 2014.] http://community.jaspersoft.com/project/jasperreports-library.

JavaScript Ya. JavaScript Ya. [En línea] [Citado el: 18 de febrero de 2014.] http://www.javascriptya.com.ar/temarios/descripcion.php?cod=2.

JBoss.org. JBoss.org. [En línea] 2013. [Citado el: 13 de febrero de 2014.] http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/pdf/richfaces_reference.pdf.

Juntao Yuan, Michael y Heute, Thomas. *JBoss Seam: Simplicity and Power Beyond Java EE.* s.l. : Prentice Hall, 2007. ISBN: 9780137129393.

Larman, Craig. *UML Y PATRONES Introducción al analisis y diseño orientado a objetos. .* México : PRENTICE HALL, 1999. ISBN: 970-1 7-0261-1.

La seguridaden las redes de comunicaciones. Alirio González, Jaime y Vanegas, Carlos Alberto. 1, Bogotá : s.n., 2006, Vol. 3. ISSN: 1794-211X.

Libros Web. [En línea] 14 de marzo de 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/xhtml/capitulo_1/breve_historia_de_html.html..

Libros Web. [En línea] 2013. [Citado el: 12 de febrero de 2014.] http://librosweb.es/css/capitulo_1.html.

Lic. Yoselyn Díaz Rodríguez. Departamento de Fuentes de Información. Infomed-Centro Nacional de Información de Ciencias Médicas, Ministerio de Salud Pública . Directorio de Instituciones en Salud. [En línea] 7 de junio de 2012. [Citado el: 20 de noviembre de 2013.] http://dirinstituciones.sld.cu/index.php?P=FullRecord&ID=321(Infomed-Directorio de Instituciones).

Linwood, Jeff y Minter, Dave. *Beginning Hibernate. Second Edition.* New York : Apress, 2010. ISBN: 978-1-4302-2851-6.

López, Carlos Llorente. Archivo Abierto Institucional de la Universidad Rey Juan Carlos. [En línea] 18 de junio de 2010. [Citado el: 13 de febrero de 2014.] http://eciencia.urjc.es/bitstream/10115/4111/1/pfcCarlosLlorenteLopez.pdf.

Martínez Jera, Erislán, Rodríguez García, Lucía y Hernández Ramírez, Martha Denia. SLD159 “ALASCLÍNICAS”: Sistema de Gestión de Ensayos Clínicos. [En línea] 3-7 de diciembre de 2012. www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/download/329/227.

OpenClinica, LLC. OpenClinica: Open Source for Clinical Research. [En línea] 2014. [Citado el: 10 de febrero de 2014.] https://www.openclinica.com/product-features.

Oracle and/or its affiliates. Oracle Documentation. [En línea] enero de 2013. [Citado el: 17 de febrero de 2014.] http://docs.oracle.com/javaee/6/tutorial/doc/gijtu.html. 821-1841-16.

Oracle. Enterprise JavaBeans Technology. [En línea] [Citado el: 18 de febrero de 2014.] http://www.oracle.com/technetwork/java/javaee/ejb/index.html.

PAREXEL International Corporation. Perceptive Informatics. [En línea] 2014. [Citado el: 10 de febrero de 2014.] http://www.parexel.com/solutions/informatics/study-management-monitoring/impact-ctms/.

Pascual López, María Amparo, y otros. Scielo. [En línea] 2011. [Citado el: 20 de febrero de 2014.] http://scielo.sld.cu/pdf/far/v45n1/far02111.pdf.

Pérez, Javier Eguiluz. Libros Web. [En línea] 14 de marzo de 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/xhtml/capitulo_1.html..

Pérez, Javier Eguiluz. Libros Web. [En línea] 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/ajax/capitulo_1.html.

PostgreSQL. PostgreSQL. [En línea] 2013. [Citado el: 10 de febrero de 2014.] http://www.postgresql.org.es/comunidad.

Quintana Rondón, Yoandri, Camejo Domínguez, Lianet y Díaz Berenguer, Abel. Investigación | FIUBA. Facultad de ingeniería Universidad de Buenos Aires. [En línea] 2011. [Citado el: 10 de febrero de 2014.] http://laboratorios.fi.uba.ar/lie/Revista/Articulos/080815/A3mar2011.pdf.

Regents of the University of Minnesota. Clinical and Translational Science Institute. [En línea] 4 de septiembre de 2013. [Citado el: 10 de febrero de 2014.] http://www.ctsi.umn.edu/biomedical-informatics/clinical-trial-management-system.

Reynoso, Carlos y Kiccillof, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 2004. [Citado el: 24 de marzo de 2014.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp.

Rodríguez García, Lucía, y otros. Revista Cubana de Informática Médica. [En línea] 25 de diciembre de 2011. [Citado el: 17 de enero de 2014.] http://www.rcim.sld.cu/revista_24/articulo_pdf/alasclinicas.pdf.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El lenguaje unificado de modelado. Manual de Referencia.* s.l. : Eddison Wesley, 2007. ISBN: 9788478290871.

Salter, David. *Seam 2.x Web Development.* Olton Birmingham : Packt Publishing, 2009. ISBN: 978-1-847195-92-0.

Sandra Almeira, Adriana, Perez Cavenago, Vanina y Beatriz Rosanigo, Zulema. Departamento de Informática Sede Trelew (DIT). *Arquitectura de Software: Estilos y Patrones.* [En línea] Marzo de 2007. [Citado el: 18 de Mayo de 2014.] http://www.roa.unp.edu.ar:8080/graduate/bitstream/123456789/203/1/Tesina%20Arquitectura%20de%20Soft.pdf.

Sommerville, Ian. *Ingeniería del Software.* Madrid : Adisson Wesley, 2005. ISBN: 84-7929-074-5.

Suárez Batista, Anisbert, y otros. Serie Científica de la Universidad de las Ciencias Informáticas (SC-UCI). [En línea] 2010. [Citado el: 28 de marzo de 2014.] http://publicaciones.uci.cu/index.php/SC.

Texas Heart Institute. Texas Heart Institute. [En línea] agosto de 2012. [Citado el: 21 de octubre de 2013.] http://www.texasheartinstitute.org/HIC/Topics_Esp/FAQ/clinical_trials_span.cfm.

Tong, Kent Ka Iok. *Beginning JSF™ 2 APIs and JBoss® Seam.* s.l. : Apress, 2009. ISBN: 978-1-4302-1923-1.

UEB Web CITMATEL. Centro de Inmunología Molecular. [En línea] 2013. [Citado el: 20 de noviembre de 2013.] http://www.cim.co.cu/cim.php.

Universidad de las Ciencias Informáticas. Entorno Productivo. [En línea] 2011. [Citado el: 28 de marzo de 2014.] http://www.uci.cu/?q=entorno-productivo.

Anexos

Anexo 1: Entrevista efectuada al cliente

Entrevista realizada a Patricia Lorenzo-Luaces Álvarez, jefa del departamento de Buenas Prácticas Clínicas en el Centro de Inmunología Molecular.

¿Qué objetivo cumple la realización de la extracción de datos para el desarrollo de sus estudios?

El objetivo es poder hacer análisis con ellos, que estos análisis lo hacemos a través de software estadísticos, que normalmente la entrada de los datos en los software estadísticos es en formato texto, excel, o spss.

¿Cómo funciona actualmente la extracción de datos en el centro?

Actualmente la extracción de datos se realiza en el módulo exportar datos, uno va seleccionando las variables de los distintos modelos, es decir, el especialista va formando el conjunto de datos que quiere exportar. Conformado ya su fichero entonces das en la opción exportar, y hasta ahora lo que más hemos hecho es exportar en spss, con este formato obtenemos dos ficheros, uno que es de sintaxis del spss ~~el cual leo la sintaxis~~ y la sintaxis lo que hace es leer el otro fichero que es un txt que contiene toda la información seleccionada, para cuando uno corre la sintaxis ya el convierte ese fichero txt a un fichero de datos leíble con el cual nosotros podemos hacer análisis.

¿Antes de usar ALAS Clínicas 1.0 como realizaban este proceso de extracción de datos?

Era manual, la entrabamos en Epidata que es un software de manejo de datos como Excel, lo que más específico para nuestro trabajo, que permite comparación de dos bases de datos, una validación de entrada de datos dobles. Este software nos permite, porque aun lo usamos con estudios que todavía no han concluido y que comenzaron antes de adquirir el ALAS Clínicas 1.0, exportar también la información en diferentes formatos.

¿Que logran ustedes al agrupar los conjuntos de datos por grupos de sujetos?

A veces uno necesita hacer análisis con subgrupos específicos de pacientes por ejemplo todos los femeninos o nos interesa excluir del análisis a pacientes que no cumplieron con el tratamiento o que fueron mal incluidos por alguna razón, o por ejemplo usar solo los sujetos que recibieron más de seis administraciones del producto de investigación y los que tuvieron menos de seis no los vamos a tener en cuenta porque no cumplieron con la etapa de inducción y tienen menos probabilidad de ser beneficiados con el tratamiento.

¿Cuales sistemas, aparte de ALAS Clínicas ustedes emplean para todo lo que tiene que ver con el desarrollo de los ensayos clínicos?

Para la entrada de datos el Epidata que todavía se emplea porque los ensayos empezaron hace seis años atrás antes de adquirir ALAS Clínicas 1.0, también para el análisis estadístico se utilizan el Stata, SPSS y también R y Winbugs que son software libres muy bien aceptados en la comunidad científica.

¿Cuáles son los objetivos estratégicos de la organización?

El Centro de Inmunología Molecular tiene como misión la investigación, desarrollo, fabricación en gran escala y la comercialización de productos biofarmacéuticos, en el mercado nacional e internacional, en especial anticuerpos monoclonales, para el diagnóstico y tratamiento del cáncer y otras enfermedades relacionadas con el sistema inmune.

Y su visión es lograr impacto a nivel mundial en el tratamiento del cáncer y otras autoinmunes con productos novedosos. Desarrollar un crecimiento sostenible de las exportaciones, accediendo a los mercados de los países industrializados. Convertirnos en una empresa Biotecnológica de referencia internacional desarrollando una cultura empresarial de excelencia sobre la base de la innovación, diversificación, y la consagración al trabajo.

Anexo 2: Descripciones de las tablas que conforman el modelo de datos

Tabla 16: Descripción de la tabla study_event.

Nombre: study_event		
Descripción: Entidad que contiene los “momentos de seguimiento” asociados a un paciente.		
Atributo	**Tipo**	**Descripción**
study_event_id	Integer	Identificador del MS en conducción.
study_event_definition_id	Integer	Identificador del MS en diseño.
study_subject_id	Integer	Identificador del sujeto.
owner_id	Integer	Identificador del sujeto o usuario que insertó o modificó la hoja

		CRD.
update_id	Integer	Identificador para cada vez que se modifique una hoja CRD.
date_start	Timestamp	Fecha de inicio del MS.
date_end	Timestamp	Fecha fin del MS.
date_created	Date	Fecha de creación de la hoja CRD.
date_updated	Date	Fecha de actualización de la hoja CRD.
dia	Integer	Día en que se realiza el MS.

Tabla 17: Descripción de la tabla study_event_definition.

Nombre: study_event_definition		
Descripción: Entidad que contiene las “definiciones de momentos de seguimientos”, es decir, contiene todos los momentos de seguimientos que se le pueden asociar a un paciente.		
Atributo	**Tipo**	**Descripción**
study_event_definition_id	Integer	Identificador del MS.
study_id	Integer	Estudio activo en el que se realizo el MS.
status_id	Integer	Estado del MS.
name	Varchar	Nombre del MS.
description	Varchar	Descripción del MS.
type	Varchar	Tipo de MS.
date_created	Date	Fecha de creación de la hoja CRD.
date_updated	Date	Fecha de actualización de la hoja CRD.

ordinal	Numeric	Valor de posición del MS en el listado de MS.
oc_oid	Varchar	Identificador del MS.
dia	Varchar	Listado de días donde se efectúa el MS.
tiempo_llenado	Integer	Tiempo de llenado del MS.

Tabla 18: Descripción de la tabla event_crf.

Nombre: event_crf		
Descripción: Entidad que contiene las “hojas de CRD” asociadas a un “MS”.		
Atributo	**Tipo**	**Descripción**
event_crf_id	Integer	Identificador de la tupla en la tabla event_crf.
study_event_id	Numérico	Identificador del MS asociado a la hoja.
crf_version	Numérico	Identificador de la versión de la hoja asociado al MS.
date_interviewed	Date	Fecha de la entrevista.
interviewed_name	Varchar	Nombre del entrevistador.
status_id	Numérico	Identificador del estado.
owner_id	Numérico	Identificador del usuario que creó la tupla.
date_created	Date	Fecha de creación de la tupla.
study_subject_id	Integer	Identificador del sujeto en el estudio.
date_updated	Date	Fecha de actualización de la tupla.

updated_id	Numérico	Identificador del usuario que actualizo la tupla.

Tabla 19: Descripción de la tabla event_definition_crf.

Nombre: event_definition_crf		
Descripción: Entidad que asocia las entidades “study_event_definition” con “crf” donde se asignan varios “CRFs” a varias “definiciones de momentos de seguimiento”.		
Atributo	**Tipo**	**Descripción**
event_definition_crf_id	Integer	Identificador de la hoja CRD asociada al MS.
study_event_definition_id	Integer	Identificador del MS.
study_id	Integer	Estudio activo en el que se introdujo la hoja en el MS.
crf_id	Integer	Identificador de la hoja CRD asociada.
default_version_id	Numérico	Versión de la hoja CRD.
status_id	Numérico	Estado de la hoja CRD asociada al MS.
owner_id	Numérico	Sujeto o usuario activo.
date_created date	Date	Fecha de creación.
date_updated	Date	Fecha de actualización.
updated_id	Numérico	Identificador del usuario que actualizó la tupla.

Tabla 20: Descripción de la tabla user_account.

Nombre: user_account

Descripción: En esta entidad se insertan los datos de los usuarios después de creados, además cuando se muestra el listado de usuarios, se cargan los datos a mostrar de dicha tabla, así como otras funcionalidades con respecto a los usuarios.

Atributo	Tipo	Descripción
user_id	Integer	Representa el identificador único del usuario, el cual es generado automáticamente.
user_name	Varchar	Nombre del usuario.
passwd	Varchar	Representa la contraseña asignada al usuario.
first_name	Varchar	Indica el primer nombre del usuario.
last_name	Varchar	Indica los apellidos del usuario.
email	Varchar	Contiene el correo electrónico del usuario.
active_study	Numérico	Representa el estudio activo al que pertenece el usuario.
institutional_affiliation	Varchar	Institución a la que pertenece el usuario.
status_id	Integer	Indica el identificador de la entidad estado.
date_created	Date	Fecha de creación del usuario.
date_updated	Date	Fecha de actualización del usuario.
date_lastvisit	Date	Fecha de la última visita del usuario.
passwd_timestamp	Date	Período de duración de la

		contraseña asignada al usuario.
passwd_challenge_question	Varchar	Representa la pregunta en caso de olvidada la contraseña de un usuario en específico.
passwd_challenge_answer	Varchar	Contiene la nueva contraseña introducida por el usuario.
phone	Varchar	Indica el teléfono del usuario.
update_id	Numérico	Indica el identificador del rol que actualizo los datos de un usuario en específico.
conectado	Integer	Representa si es usuario está habilitado o deshabilitado.
ip	Varchar	Indica las direcciones IP asignadas al usuario.
ip_rango	Varchar	Indica el rango IP asignado al usuario.

Tabla 21: Descripción de la tabla study_user_role.

Nombre: study_user_role		
Descripción: Esta entidad generada contiene la relación entre los estudios, usuarios y roles del sistema informático.		
Atributo	**Tipo**	**Descripción**
id	Integer	Representa el identificador único de esta entidad, el cual es generado automáticamente.
role_name	Varchar	Nombre del rol en el sistema.
study_id	Numérico	Indica el identificador del estudio.

status_id	Numérico	Indica el identificador de la entidad estado.
date_created	Date	Representa la fecha de confección de la entidad study_user_role.
date_updated	Date	Representa la fecha de actualización de la entidad study_user_role.
user_name	Varchar	Representa el nombre del usuario.
ip	Varchar	Representa las direcciones IP de la entidad study_user_role.
ip_rango	Varchar	Representa el rango de IP de la entidad study_user_role.

Tabla 22: Descripción de la tabla user_type.

Nombre: user_type		
Descripción: Esta entidad contiene los diferentes tipos de usuarios que acceden al sistema.		
Atributo	**Tipo**	**Descripción**
user_type_id	Integer	Representa el identificador del tipo de usuario.
user_type	Varchar	Indica la descripción del tipo de usuario.

Tabla 23: Descripción de la tabla audig_log_event_type.

Nombre: audig_log_event_type

Descripción: Tabla que contiene el tipo de evento ocurrido en el sistema.		
Atributo	**Tipo**	**Descripción**
audig_log_event_type_id	Integer	Identificador del tipo de evento.
name	Varchar	Nombre del evento.

Tabla 24: Descripción de la tabla status.

Nombre: status		
Descripción: Entidad que contiene los tipos de "estados" definidos para "pacientes", "momentos de seguimientos" y "hojas de CRD".		
Atributo	**Tipo**	**Descripción**
status_id	Integer	Representa el identificador del estado.
name	Varchar	Indica el nombre del estado.
description	Varchar	Descripción del estado.

Tabla 25: Descripción de la tabla cronograma.

Nombre: cronograma		
Descripción: Esta entidad maneja la información referente a los cronogramas.		
Atributo	**Tipo**	**Descripción**
cronograma_id	Integer	Identificador del cronograma.
study_id	Integer	Estudio activo en el que se realizo el cronograma.
status_id	Integer	Estado del cronograma.
tiempo_duracion	Integer	Tiempo que se estima durará el

		cronograma.
description	Varchar	Descripción del cronograma.
date_created	Date	Fecha de creación del cronograma.
date_updated	Date	Fecha de actualización del cronograma.
owner_id	Integer	Sujeto o usuario que inserto o modifico el cronograma.
study_group_id	Integer	Grupo de sujeto al cual pertenece dicho cronograma.
update_id	Integer	Identificador para cada vez que se modifique un cronograma determinado.

Tabla 26: Descripción de la tabla study_subject.

Nombre: study_subject		
Descripción: Entidad que contiene todos los “sujetos” que se deciden estudiar en un ensayo y pasan a ser “pacientes”.		
Atributo	**Tipo**	**Descripción**
study_subject_id	Integer	Identificador del sujeto en el estudio.
subject_id	Integer	Identificador del sujeto.
study_id	Integer	Identificador del estudio.
status_id	Integer	Estado de entidad.
enrollment_date	Date	Fecha de matriculado el sujeto.
date_created	Date	Fecha de creación del sujeto en el estudio.

date_updated	Date	Fecha de última actualización del sujeto en el estudio.
owner_id	Integer	Identificador del usuario que creó el sujeto en el estudio.
updated_id	Integer	Identificador del usuario que actualizó el sujeto en el estudio.
oc_oid	Varchar	Identificador único del sujeto en el estudio.
estado_tratamiento	Varchar	Estado del tratamiento.
has_specific_scheduled	Boolean	Especifica si el sujeto tiene cronograma específico.
schedule_date	Date	Fecha programada.
fecha_interrupcion	Date	Fecha Interrupción.
estado_monitoreo	Varchar	Estado de monitoreo.

Tabla 27: Descripción de la tabla etapa.

Nombre: etapa		
Descripción: Entidad que contiene los nombres, días de inicio y fin de las “etapas” asociadas a un “cronograma general”.		
Atributo	**Tipo**	**Descripción**
etapa_id	Integer	Identificador de la etapa.
cronograma_id	Integer	Identificador del cronograma al cual pertenece la etapa.
nombre	Varchar	Nombre de la etapa.
inicio_etapa	Integer	Inicio de la etapa.
fin_etapa	Integer	Fin de la etapa.
descripcion	Varchar	Descripción de la etapa.

date_created	Date	Fecha de creada la etapa.
date_updated	Date	Fecha de actualizada la etapa.
updated_id	Numérico	Identificador del usuario que realizo la última actualización de la etapa.
owner_id	Numérico	Identificador del usuario que creó la etapa.

Tabla 28: Descripción de la tabla etapa_study_event_definition.

Nombre: etapa_study_event_definition		
Descripción: Entidad que asocia las “definiciones de momentos de seguimiento” existente a una “etapa” definida en el “cronograma general”.		
Atributo	**Tipo**	**Descripción**
etapa_id	Integer	Identificador de la etapa.
study_event_definition_id	Integer	Identificador de el MS.

Tabla 29: Descripción de la tabla item.

Nombre: item		
Descripción: Entidad que contiene las “variables” existentes en una “hoja de CRD”.		
Atributo	**Tipo**	**Descripción**
item_id	Integer	Identificador de la variable.
name	Varchar	Nombre de la variable.
description	Varchar	Descripción de la variable.
units	Varchar	Unidad de la variable.

item_data_type_id	Integer	Identificador del tipo de dato de la variable.
status_id	Numérico	Identificador del estado de la variable.
owner_id	Numérico	Identificador del usuario que creó la variable.
date_created	Date	Fecha de creada la variable.
date_updated	Date	Fecha de actualizada la variable.
updated_id	Numérico	Identificador del usuario que realizo la última actualización de la variable.
oc_oid	Varchar	Identificador de la variable en el estudio.

Tabla 30: Descripción de la tabla item_data_type.

Nombre: item_data_type		
Descripción: Entidad que contiene la descripción del tipo de dato que puede estar asociado a una variable en una “hoja de CRD”.		
Atributo	**Tipo**	**Descripción**
item_data_type_id	Integer	Identificador del tipo de dato.
code	Varchar	Código del tipo de dato.
name	Varchar	Nombre del tipo de dato.
definition	Varchar	Definición del tipo de dato.
reference	Varchar	Referencia del tipo de dato.

Tabla 31: Descripción de la tabla archive_dataset_file.

Nombre: archive_dataset_file		
Descripción: Entidad que contiene los datos sobres los ficheros de los conjuntos de datos que se han exportado.		
Atributo	**Tipo**	**Descripción**
archive_dataset_file_id	Integer	Identificador del fichero que se ha exportado.
name	Varchar	Nombre del fichero que se ha exportado.
dataset_id	Numérico	Identificador del conjunto de datos que se haya exportado.
export_format_id	Numérico	Identificador del tipo de formato al cual se haya exportado el fichero.
file_reference	Varchar	Dirección en el sistema de archivo donde se encuentra ubicado el fichero.
run_time	Numérico	Tiempo que duro el proceso de exportar el conjunto de datos.
file_size	Numérico	Tamaño del fichero de conjunto de datos.
date_created	Date	Fecha de creado el fichero de conjunto de datos
owner_id	Numérico	Identificador del usuario que creó el fichero de conjunto de datos.

Tabla 32: Descripción de la tabla export_format.

Nombre: export_format

Descripción: Entidad que contiene el tipo de formato al cual puede ser exportado un fichero de conjunto de datos.		
Atributo	**Tipo**	**Descripción**
export_format _id	Integer	Identificador del tipo de formato.
description	Varchar	Descripción del tipo de formato.
name	Varchar	Nombre del tipo de formato.
mime_type	Varchar	