Universidad de las Ciencias Informáticas

Facultad 2

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

Desarrollo de funcionalidades para la conducción de estudios en el Sistema de Gestión de Ensayos Clínicos

Autores: Mirienny Moya Chaviano

Eiler Efraín Sarmiento Ávila

Tutores: Ing. Erislán Martínez Jera

Ing. Yasmany Nuñez Broch

La Habana, junio de 2014

"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del presente trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 23 días del mes de junio del año 2014.

_____ _____

Firma del autor *Firma del autor*

Mirienny Moya Chaviano *Eiler Efraín Sarmiento Ávila*

_____

Firma del tutor

Ing. Erislán Martínez Jera

_____

Firma del tutor

Ing. Yasmay Nuñez Broch

DATOS DE CONTACTO

SÍNTESIS DEL TUTOR

Ing. Erislán Martínez Jera. Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2009. Vinculado al Departamento Sistemas Especializados en Salud (SES) del Centro de Informática Médica (CESIM) y específicamente se desempeña como jefe del proyecto Sistema de Gestión de Ensayos Clínicos.

Correo electrónico: ejera@uci.cu

Ing. Yasmany Nuñez Broch. Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2012. Recién Graduado en Adiestramiento. Se desempeña como Programador Principal del Sistema de Gestión de Ensayos Clínicos del Departamento Sistemas Especializados en Salud (SES), del Centro de Informática Médica (CESIM).

Correo electrónico: ynbroch@uci.cu

RESUMEN

El Centro de Inmunología Molecular (CIM), inaugurado el 5 de diciembre de 1994, perteneciente al Polo Científico del Oeste de La Habana, tiene como principal misión obtener y producir nuevos bio-fármacos destinados al tratamiento del cáncer y otras enfermedades crónicas no transmisibles, e introducirlos en la salud pública cubana para prolongar la expectativa de vida de los pacientes. Para dicho objetivo es necesaria la recogida de gran cantidad de datos, asegurando su legitimidad y almacenamiento confiable. Actualmente para recoger y gestionar la información el centro cuenta con el sistema Clínicas 1.0, el cual en su Módulo Enviar datos, permite la gestión de la información referente a la conducción de los ensayos clínicos. Dicho módulo presenta deficiencias en su funcionamiento, provocando existencia de información incorrecta o innecesaria y haciendo engorrosa la planificación en el cronograma específico de los sujetos, por lo que se dio la tarea de desarrollar funcionalidades para el Módulo Conducción del Sistema de Gestión de Ensayos Clínicos, eliminando las faltas identificadas.

Es por ello que partiendo de la necesidad de contar con un sistema que mejore el proceso de gestión de la información de la conducción de los ensayos clínicos, con el objetivo de centralizar, controlar y agilizar dicha conducción; se desarrolló una solución que constituye un sistema configurable y flexible ante posibles cambios, que permite el almacenamiento y gestión de los sujetos, los cronogramas específicos y las hojas CRD, sirviendo como mecanismo de apoyo a la hora de hacer las evaluaciones de los nuevos fármacos.

Palabras Claves: bio-fármacos, conducción, ensayos clínicos, legitimidad, sistema.

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La informatización de la sociedad actual deviene en un proceso ordenado e intensivo de las Tecnologías de la Información y las Comunicaciones (TIC) en la vida renovada, para satisfacer las necesidades de las esferas sociales en un esfuerzo consistente por lograr cada vez más eficiencia en los procesos y por ende mayor generación de riqueza y aumento en la calidad de vida de las naciones. En esta era se debe brindar información correcta, óptima, oportuna y rápida, estas características permiten hacer más corta la distancia y más óptimo el tiempo para la humanidad. (1)

Cuba ha identificado la necesidad de introducir en la práctica social las TIC; y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, para facilitar a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible. (2) Por esta razón está inmersa en un programa de renovación de los servicios que ofrecen sus instituciones más importantes utilizando soporte tecnológico, entre ellas el Centro de Inmunología Molecular (CIM), inaugurado el 5 de diciembre de 1994, perteneciente al Polo Científico del Oeste de La Habana. En este centro se llevan a cabo estudios de medicamentos aplicados a pacientes, teniendo como principal misión obtener y producir nuevos bio-fármacos destinados al tratamiento del cáncer y otras enfermedades crónicas no transmisibles e introducirlos en la salud pública cubana, para prolongar la expectativa de vida de los pacientes. Además, pretende hacer la actividad científica y productiva económicamente sostenible y realizar aportes importantes a la economía del país. (3)

En el CIM se encuentra instalado desde el 2012 el sistema Clínicas 1.0, desarrollado por profesores y estudiantes del Centro de Informática Médica (CESIM) de la Universidad de las Ciencias Informáticas (UCI). Los especialistas de este centro lo utilizan para la gestión de los ensayos clínicos (EC) que se desarrollan en el mismo, para lo cual cuenta con los módulos Administrar Sistema, Gestionar Estudio, Enviar Datos y Extraer Datos. El módulo “Enviar Datos”, entre sus funcionalidades permite la gestión de la información en la conducción de los ensayos clínicos, a través de la gestión de los sujetos por estudios y centros, de los cronogramas específicos y los datos clínicos registrados en cada una de las hojas del cuaderno de recogida de datos (CRD) asociadas a los mismos y planificadas por momentos de seguimiento.

En dicho sistema una vez insertados los sujetos no pueden ser modificados, borrados o deshabilitados, lo que propicia que el sistema pueda contar en algún momento con información incorrecta o innecesaria. Al no tener en cuenta los errores que puedan cometer los usuarios en la entrada de los datos del sujeto, si los mismos entran alguna información errónea deberán incluir un nuevo sujeto, causando que a la hora de exportarlos puedan existir datos repetidos, dificultando así la toma de decisiones para la evaluación de los nuevos fármacos. Se eliminaría además la posibilidad de mejorar los indicadores que se modifican con la introducción de los nuevos resultados, y los cambios que se producen en los patrones de atención médica de la enfermedad en que se evalúa el producto, pues los resultados pueden no ser válidos trayendo consigo la existencia de reacciones adversas.

Los momentos de seguimientos describen un tipo de seguimiento que sucede durante la realización del estudio y en los cuales se llenan las hojas CRD, poseen además un conjunto de características entre las que se encuentran días, plazo de llenado y tipo, que especifican como deben ser; con ellos sucede una situación similar a la gestión de los usuarios. Sus fechas de inicio y fin una vez que el sistema las planifica automáticamente al iniciar la ejecución del ensayo clínico no se pueden modificar, provocando que puedan existir momentos de seguimiento que inicien un domingo o un día que no coincida con el día real de la consulta del doctor en el hospital, generando atrasos en el cronograma específico de los pacientes y por tanto una planificación no funcional en la práctica. Esto trae como consecuencia que el posterior proceso de captura de los datos no se realice siguiendo un cronograma correcto, recogiéndose información innecesaria o se queden datos sin recopilar, y dicha recogida debe seguir un sentido lógico y constante para la obtención de resultados confiables. Además causaría confusión, pérdida de tiempo y molestias a los pacientes pues acudirán a su consulta planificada y si esta coincide con un día excepcional (dígase día feriado, domingo, etc.) no serán atendidos.

Luego de la situación analizada anteriormente se plantea como **problema a resolver** ¿Cómo mejorar la gestión de información en la conducción de ensayos clínicos del sistema Clínicas 1.0? Como **objeto de estudio** se define el proceso de conducción de ensayos clínicos y el **campo de acción** identificado es la gestión de la información de los sujetos, los cronogramas específicos y las hojas CRD en la conducción de ensayos clínicos.

INTRODUCCIÓN

El **Objetivo General** es desarrollar en el módulo Conducción del Sistema de Gestión de Ensayos Clínicos funcionalidades de forma tal que se mejore la gestión de la información de los sujetos, los cronogramas específicos y las hojas del Cuaderno de Recogida de Datos.

Para dar cumplimiento al objetivo general se trazan las siguientes **tareas de la investigación:**

1. Analizar el proceso de gestión de ensayos clínicos en el Centro de Inmunología Molecular, para profundizar en la gestión de la información en la conducción de los mismos.

2. Describir los términos y conceptos relacionados con la gestión de ensayos clínicos, para lograr un mayor entendimiento del negocio.

3. Asimilar las herramientas y tecnologías propuestas por el Centro de Informática Médica, para el desarrollo de la solución propuesta.

4. Desarrollar los artefactos correspondientes a los Flujos de Trabajo: Modelo de Negocio, Requisitos, Análisis y Diseño e Implementación según la metodología Proceso Unificado de Desarrollo.

5. Implementar funcionalidades que permitan la gestión de los sujetos, los cronogramas específicos y las hojas del Cuaderno de Recogida de Datos en el módulo Conducción del Sistema de Gestión de Ensayos Clínicos.

Todo el trabajo investigativo se realizó siguiendo un enfoque **Dialéctico Materialista** como método general de la ciencia, los diferentes métodos utilizados reflejan este enfoque.

Los **métodos de investigación** empleados fueron:

Métodos teóricos:

Análisis Histórico Lógico: Utilizado para el estudio de investigaciones relacionadas con el proceso de gestión de ensayos clínicos y su evolución.

Inductivo - Deductivo: Empleado para realizar un análisis detallado a los sistemas existentes que gestionan la información referente a la conducción de ensayos clínicos, para ver si podían ser reutilizados; así como para la obtención de los elementos más importantes que serán utilizados en la investigación una vez consultada la bibliografía.

Analítico - Sintético: Se realizó el estudio de los conceptos relacionados con el objeto de la investigación, con el fin de obtener una mejor claridad del proceso.

Modelación: Se realizaron los diagramas correspondientes a los Flujos de Trabajo: Modelo de Negocio, Requisitos, Análisis y Diseño e Implementación según la metodología RUP.

Sistémico: Es empleado para la descripción de los conceptos asociados a la gestión de los sujetos, el cronograma específico y las hojas CRD, como componentes de la conducción de ensayos clínicos.

Análisis documental: Se utilizó para revisar la bibliografía relacionada tanto con el objeto de estudio, para comprender sus particularidades, nexos y desarrollo; así como para el estudio de los referentes teóricos que permitieron desarrollar la propuesta.

Métodos empíricos:

Entrevista: Se realizó una entrevista al Ing. Erislán Martínez Jera, principal desarrollador del sistema Clínicas 1.0 para lograr un mayor entendimiento del negocio.

Observación: Permitió la obtención de información detallada referente al objeto de investigación en el sistema Clínicas 1.0, para el esclarecimiento de la problemática.

Beneficios esperados:

1. Al permitir la modificación de los momentos de seguimientos y las hojas CRD en el cronograma específico después de iniciada la conducción del ensayo clínico, logra una correcta planificación evitándose la recogida de información incorrecta o innecesaria.
2. Permite realizar una gestión flexible de la información referente a la conducción de ensayos clínicos, facilitándole a los especialistas del centro la toma de decisiones.

INTRODUCCIÓN

El documento se encuentra divido en 4 capítulos, cuyo contenido se describe a continuación:

Capítulo 1: Fundamentación teórica

Se analizan los distintos sistemas que gestionan ensayos clínicos en el ámbito nacional e internacional. Además, se describe la metodología de desarrollo, las herramientas y los lenguajes de programación a utilizar en la implementación de las funcionalidades.

Capítulo 2: Descripción del sistema

Se elabora el Modelo conceptual y se explican los términos relacionados con él, permitiendo obtener una base para la obtención de los requerimientos funcionales y no funcionales. Se realiza además la propuesta de solución al problema de investigación, así como el Diagrama de casos de usos y la descripción de los mismos.

Capítulo 3: Diseño del Sistema.

Se describen los aspectos relacionados con el diseño de la solución propuesta, los patrones de diseño a utilizar en el desarrollo de la aplicación, se presentan los diagramas de clases de los casos de uso del sistema utilizando estereotipos web. Conformándose, finalmente, el Modelo de Diseño, el cual constituye el punto de partida para la implementación del sistema.

Capítulo 4: Implementación.

Se especifican los estándares utilizados, el Diagrama de componentes, y algunas de las interfaces del sistema. Se presenta el Modelo de datos describiendo las tablas que lo conforman y la propuesta de despliegue.

Finalmente se presentan las Conclusiones, Recomendaciones, Bibliografía y Glosario de Términos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Una de las primeras actividades que realiza todo investigador, es un análisis de los elementos teóricos que van a servir como base a la problemática; así como a otras investigaciones para esclarecer las ideas respecto al tema de interés, y con ello poder afinarlo, delimitarlo y enfocarlo desde la perspectiva de los objetivos planteados en este trabajo. Permite la caracterización de cada uno de ellos de forma tal que se logre comprender el entorno de la investigación y la teoría que la sustenta. En este capítulo se describen además las metodologías, tecnologías y herramientas de software propuestas a utilizar en el desarrollo del sistema, teniendo en cuenta su importancia para obtener un producto de calidad.

1.1. Conceptos básicos asociados al dominio del problema

El desarrollo de un fármaco engloba su seguridad, eficacia, formulación y fabricación. Normalmente, se realizan estudios de investigación que prueban el funcionamiento de los nuevos enfoques clínicos en las personas, denominados **ensayos clínicos**.
Cada estudio responde preguntas científicas e intenta encontrar mejores formas de prevenir, explorar, diagnosticar o tratar una enfermedad, además de percibir las reacciones adversas y los efectos del uso a largo plazo de un medicamento nuevo en seres humanos. Se puede comparar un tratamiento nuevo con uno que ya se encuentra disponible.

En la realización de un ensayo clínico intervienen:

1. Promotor, investigador principal, monitor.

2. Sujetos del ensayo clínico.

3. Producto objeto de ensayo clínico.

- ✓ Promotor: Es la persona física o jurídica que tiene interés en su realización. Se responsabiliza del ensayo, incluyendo su organización, comienzo y financiación.
- ✓ Investigador principal: Es quién dirige la realización práctica del ensayo. Solamente podrá actuar como investigador un profesional sanitario suficientemente calificado para evaluar la respuesta a la

sustancia o medicamento objeto de estudio, con experiencia en investigación y en el área clínica del ensayo propuesto y con reconocidos criterios de ética e integridad profesional.

- ✓ Monitor: Es el profesional capacitado con la necesaria competencia clínica elegido por el promotor, que se encarga del seguimiento directo de la realización del ensayo.
- ✓ Sujetos del ensayo clínico: Es la persona sana o enferma que participa en el mismo, después de haber otorgado libremente su consentimiento informado.
- ✓ Producto objeto de ensayo clínico: Es el resultado final que se obtiene una vez concluido el ensayo clínico.(4)

Ventajas de los ensayos clínicos:

1. Son experimentos controlados: El investigador diseña un protocolo de investigación en el que define mecanismos de control que operarán antes y durante el desarrollo de la fase experimental con el objeto de cautelar la seguridad del sujeto de experimentación.
2. Rigor para establecer causa: Es capaz de comprobar hipótesis causales.
3. Prueba de efectividad, eficacia y equivalencia: El diseño experimental permite caracterizar la naturaleza profiláctica o terapéutica de diferentes intervenciones médicas.
4. Examina efectos adversos: El desarrollo de un estudio experimental permite conocer y cuantificar la aparición de efectos colaterales indeseados a consecuencia de la intervención en estudio.

Desventajas:

1. La experimentación en seres humanos confiere a los ensayos clínicos un alto grado de complejidad.
2. La naturaleza de los estudios clínicos experimentales exige el uso de productos biológicos, farmacológicos o procedimientos terapéuticos y de control y monitoreo no exentos de costo.(5)

Conducción de Ensayos Clínicos:

La Conducción de los ensayos clínicos está conformada por tres procesos diferentes, estos son:

Gestión de los sujetos: En este proceso se insertan los sujetos a los cuales se les realizarán los estudios, además se introducen sus datos personales, se ubican en un grupo según sus características, enfermedades o por tratamientos.

Gestión de los datos asociados al estudio de los sujetos: Se gestionan los datos de los sujetos, se interrumpe el tratamiento asignado a cada uno, entre otras funcionalidades.

Monitorización de los datos asociados al estudio de los sujetos: Permite la realización de auditorías a los datos arrojados por el sistema, después de concluir los estudios realizados a los sujetos.

1.2. Sistemas de Gestión de ensayos clínicos a nivel internacional

Un Sistema de Gestión de Ensayos Clínicos (CTMS), es un software personalizable utilizado por las industrias biotecnológicas, farmacéuticas e instituciones de investigación clínica para gestionar las grandes cantidades de datos que participan en la operación de un ensayo clínico . En la actualidad existen diversos CTMS que gestionan la planificación, preparación, desempeño y presentación de informes de los ensayos clínicos, con énfasis en mantener información de contacto actualizada para los participantes y los plazos de seguimiento e hitos como los de aprobación de los reguladores o la emisión de informes de avance.

A nivel nacional e internacional han sido diseñadas y desarrolladas varias soluciones informáticas, con el fin de gestionar una u otra información de los ensayos clínicos. A continuación se muestra una breve reseña de algunos de estos sistemas.

1.2.1. Solución Imagys para un ensayo clínico internacional en Reumatología

Las soluciones de IMAGYS han sido diseñadas específicamente por los expertos en proyección de imagen y telecomunicaciones médicas de la empresa Keosys con sede en Francia, para ayudar a ejecutar ensayos clínicos con más exactitud mientras que disminuyen la incertidumbre y el coste de desarrollo terapéutico. Lanzado en el año 2009 revoluciona el proceso de conducción de ensayos clínicos, permitiendo la recogida de datos e imágenes, lo que ayuda a manejar cada aspecto de la gerencia de datos médicos de la proyección de imagen en cada etapa de su ensayo incluyendo la disposición del estudio.

Proporciona los siguientes servicios para los ensayos clínicos:

- ✓ Gestión de proyectos.
- ✓ Protocolos específicos de diseño, desarrollo y validación de ensayos basados en módulos Imagys genéricos.
- ✓ Pruebas de validación.
- ✓ Creación y suministro de archivos administrativos y técnicos conforme a los reglamentos.
- ✓ Sistema de gestión de datos de imagen y de la infraestructura disponible. (6)

1.2.2. mPRO Clinical Trial Manager

El sistema mPRO Clinical Trial Manager desarrollado por la empresa estadounidense OBS Medical corre sobre mPRO, tomando como base la telefonía móvil los resultados son reportados electrónicamente por pacientes (ePRO), siendo una solución combinada con el diseño del estudio y el análisis de expertos, facilitando la distribución mundial de la alta conformidad y ensayos clínicos de bajo costo con tecnología familiar.

Permite la obtención de resultados y estadísticas en tiempo real, rápido despliegue y bloqueo de datos de ensayos clínicos. Los datos se introducen por los sujetos en los programas interactivos de diario en un teléfono móvil, utilizando redes de datos de teléfonos móviles seguras, la información se descarga automáticamente a un servidor seguro, y a este puede tener acceso la empresa a cargo del estudio asegurando que se identifiquen las oportunidades lo más temprana posible y la reducción de la línea de tiempo entre el inicio de prueba y bloqueo de datos de prueba. (7)

1.2.3. ARCHeS Trial Designer (ARCHeS TD)

Permite a los usuarios configurar una prueba clínica virtual que se ejecuta a través del Modelo de Arquímedes. El conjunto de datos de previsión resultante puede ser explorado a fondo en ARCHeS Outcomes Analyzer (ARCHeS OA). Ofrece una interfaz web completa con herramientas intuitivas que lleva a los usuarios paso a paso a través del proceso de creación de su prueba virtual.

Permite estimar el tamaño del ensayo clínico, identificar grupos / perfiles de pacientes, y predecir las tasas de eventos, mitigar el riesgo de fracaso clínico, realizar análisis comparativos de eficacia y coste-

efectividad y la entrada de datos online. Gestiona los estudios, los usuarios y derechos de acceso, las pantallas son de fácil uso, la entrada de datos es intuitiva y con rápida familiarización. Garantiza la normalización y uso de plantillas, la importación de datos basado en archivos ASCII, la exportación de datos a SPSS, EXCEL y para los formatos ASCII, HTML y XML así como la creación de informes y monitorización de ensayos clínicos. (8)

1.2.4. BCX Research

BCX Research es un software para la creación del cuaderno de recogida de datos (CRD) de cualquier investigación biológica o sanitaria, de un modo fácil e intuitivo, siguiendo un procedimiento simple, que facilita la creación del soporte donde posteriormente se recogerán los datos de los centros, visitas y pacientes. Esta forma de crear la estructura del estudio reduce considerablemente el tiempo de creación del cuaderno de recogida de datos (CRD), siendo más efectivo en su composición.

Existen tres versiones diferentes de esta aplicación (open, pro y premium), diferenciándose cada una de ellas en las funcionalidades y ventajas que ofrecen a la hora de crear la estructura del estudio ofreciendo flexibilidad para que el investigador seleccione la aplicación que mejor se ajuste a sus necesidades. (9)

- **BCX research open:** Esta herramienta puede apoyar la etapa de recogida de datos en estudios con muestras de tamaño reducido. Ideal para cualquier profesional que quiera realizar estudios no simultáneos individualmente.
- **BCX research pro:** Es la herramienta de apoyo ideal para diseñar la estructura del cuaderno de recogida de datos (CRD) y registrar la información que se almacenará en el mismo. Permite la gestión de paneles y la exportación filtrada de los datos almacenados en el cuaderno de recogida.
- **BCX research premium:** Permite realizar distintos estudios por un conjunto de personas simultáneamente. No incluye restricciones en cuanto a número de pacientes, visitas o campos a estudiar, pudiendo abarcar investigaciones destinadas una gran muestra de la población.

1.2.5. BCX Clinical

BCX Clinical es un software para ensayos clínicos que permite diseñar el CRD del estudio de un modo fácil e intuitivo siguiendo un procedimiento simple, que facilita la creación del soporte donde

posteriormente se recogerán los datos de los centros, visitas y pacientes. Esta forma de crear la estructura del estudio reduce considerablemente el tiempo de creación del CRD, siendo más efectivo en su composición.

Está compuesta por dos aplicaciones de escritorio: **BCX clinical studio**, que proporciona las herramientas necesarias para el diseño, en versión digital, que tendrá el cuaderno de recogida de datos (CRD), y **BCX clinical core**, que proporciona la interfaz de recogida de los datos de los individuos del estudio, facilitando además las herramientas necesarias para la validación y exportación de los datos almacenados. (10)

1.2.6. OpenClinica

Es un software de código abierto para la gestión integral de estudios clínicos a través del control total en tiempo real del desarrollo del estudio. Permite la construcción y administración de estudios, la introducción de datos en CRD electrónico, la monitorización y gestión de datos así como su presentación, validación, anotación, importación, filtrado y exportación de datos a SPSS, Excel y otros formatos. Garantiza seguridad y control a través de múltiples roles y perfiles, además de ser multiusuario, multiestudio y multicentro. (11)

Dentro de sus características técnicas se incluyen:

- Clasificación de los datos del estudio clínico por estudio, protocolo y centro (cada uno con sus usuarios autorizados, sus propios pacientes, sus definiciones de eventos del estudio y sus CRDs). Además incluye soporte para compartir recursos entre varios estudios para proporcionar una mayor eficiencia. Todo de manera segura y transparente.
- Generación dinámica de e-CRDs, basados en web, para la captura electrónica de los datos, según los parámetros clínicos y los parámetros de validación previamente especificados en la fase de puesta en marcha.
- Manejo longitudinal de la información para las visitas de los pacientes complejas y recurrentes.
- Herramientas de importación/exportación/migración de datos clínicos.
- Interfaz para consulta y obtención de datos segmentados por pacientes, tiempo y otros parámetros clínicos, con posibilidad de exportación de los datos a los formatos más comunes, como el delimitado por tabulaciones, SPSS o CDISC ODM XML.

- Permite roles de usuarios bien diferenciados y privilegios de acceso a datos y contraseñas, así como una seguridad autentificada (encriptación SSL de los datos transmitidos). También tiene capacidad intrínseca y global para auditorías y monitorización de los cambios efectuados en los registros de la base de datos.

Después de analizados los sistemas existentes a nivel internacional se determina que ninguno puede ser utilizado, pues el único de código abierto es OpenClinica, y este no permite planificar los momentos de seguimiento en un cronograma general que sea cumplido por todos los sujetos pertenecientes a un estudio y que estos se actualicen automáticamente, además de no realizar la gestión de notas del sitio.

Para la utilización de los demás sistemas analizados es necesario el pago de licencias y permisos lo que no responde a la necesidad de alcanzar una soberanía tecnológica. Los sistemas BCX Clinical y BCX Research se enfocan en la creación y diseño de las hojas CRD abarcando la etapa de la recogida de datos, sin embargo descarta la gestión de los sujetos en cuanto a su cronograma específico y los momentos de seguimiento planificados en él. El software mPRO Clinical Trial Manager a pesar de permitir la obtención de resultados y estadísticas en tiempo real y rápido despliegue, se basa en el uso de la telefonía móvil, lo que resulta un inconveniente para el país.

1.3. Sistemas de Gestión de ensayos clínicos a nivel nacional

Luego de realizada una investigación a nivel nacional sobre sistemas que garanticen la gestión de la información en la conducción de ensayos clínicos, arrojó que solo existe el Clínicas 1.0. Es un software desarrollado por estudiantes y profesores del Centro de Informática Médica de la Universidad de las Ciencias Informáticas, cuyo propósito es acelerar y automatizar la conducción de ensayos clínicos (EC) en el Centro de Inmunología Molecular. Basado en OpenClinica abarca sus funcionalidades y brinda además 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 EC, 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 desde las cuales se puede acceder al mismo, entre otras.

Está formado por cuatro módulos:

- El módulo Administrar Empresa: permite gestionar los permisos y roles de los usuarios que accederán al sistema, así como la administración de direcciones IP desde las cuales se accederá al sistema.
- El módulo Gestionar Estudio: permite la creación de EC, así como la confección de CRD y la creación del cronograma general de ejecución de cada EC. Además permite realizar la validación de las variables de un CRD a través del establecimiento de reglas de validación y derivación; permitiendo la disminución y detección de los errores que pueden cometerse al entrar los datos recogidos de los pacientes en el ensayo.
- El módulo Enviar Datos: permite generar el cronograma de ejecución específico para cada paciente dentro de un ensayo, así como el llenado de las hojas CRD. Además permite realizar el monitoreo de los datos de los pacientes de un sitio y la creación de notas o discrepancias asociadas a cada uno de los datos monitoreados.
- El módulo Extraer Datos: permite realizar reportes a partir de los datos almacenados por pacientes en el ensayo y realizar un reporte de las trazas de auditoría asociado fundamentalmente a las variables de los modelos y a las acciones en general que realiza determinado usuario sobre los elementos del sistema. Garantiza la salida de información en diversos formatos de uso extendido: SPSS, EXCEL y HTML.(12)

El sistema Clínicas 1.0 a pesar de reducir los costos en cuanto a transportación y material de oficina, ya que se ocupa menos espacio físico al eliminar los CRD impresos, facilitar la comunicación entre los centros que participen en la conducción del ensayo, estandarizar la información y dar rápida respuesta a los problemas, presenta deficiencias en el módulo Enviar Datos. Un ejemplo de estos problemas son que una vez que se insertan los sujetos estos no pueden ser modificados, borrados o deshabilitados, similar ocurre con los momentos de seguimientos, lo que propicia que el sistema pueda contar en algún momento con información incorrecta o innecesaria, y la planificación sea un proceso engorroso.

Además para su desarrollo no se utilizó un marco de trabajo que brindara un enlace entre las tecnologías del lado de cliente con las del servidor, imposibilitando el empleo de librerías para lograr un mayor enriquecimiento de las vistas y así ofrecer un producto más amigable para el cliente. También, es necesario gran cantidad de ficheros .xml para lograr una comunicación entre la Programación Orientada a Objetos (POO) y el modelo relacional ya que no fue empleada ninguna herramienta de mapeo objeto-relacional. El soporte de este software, según sus características, se hace costoso en cuanto a tiempo, lo

que imposibilitaría la adición de nuevas funcionalidades. Por lo que surge la necesidad de implementar una nueva y mejorada versión del módulo tomando como base el sistema Clínicas 1.0, que se encargue de subsanar las deficiencias encontradas.

Para el desarrollo de la solución propuesta, fue de vital importancia realizar un estudio del arte, analizando los sistemas existentes tanto en el ámbito nacional como internacional, y con ello obtener un punto de partida para conseguir los objetivos trazados.

1.4. Técnicas, tecnologías, metodologías y software, en las que se apoya la solución del problema

Para el desarrollo del sistema se utilizaron tecnologías y herramientas definidas en el departamento Sistemas Especializados en Salud (SES) del Centro de Informática Médica (CESIM) perteneciente a la UCI.

1.4.1. Metodología de desarrollo de software

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y eficiencia en el proceso de generación de software. En la actualidad existen diversas metodologías que son utilizadas a la hora de crear los proyectos en dependencia a las características y la envergadura del mismo.

Rational Unified Process (RUP): Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Utiliza UML para definir los modelos de software y se divide en 4 fases: Inicio, Elaboración, Construcción y Transición.

En esta metodología se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería, de los cuales se desarrollaran las actividades correspondientes a los flujos de Diseño y Pruebas. Los tres últimos constituyen flujos de

apoyo. RUP tiene tres características esenciales: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. (13)

1.4.2. Tecnologías

Tecnologías es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes y servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de las personas. (14)

JBoss Seam: Es un framework moderno que integra la capa de presentación Java Server Faces (JSF) con la capa de negocios y persistencia Enterprise Java Bean (EJB). Una característica importante es que se pueden hacer validaciones en los POJOs (Plain Object Java). Mientras que en los frameworks tradicionales, todo el estado es administrado básicamente en la sesión HTTP, Seam provee una mayor granularidad de contextos de estado.

Algunos de los frameworks con los que se integra son:

- ✓ Java Server Faces (JSF): Para el desarrollo de la interfaz de usuario, en este sentido Seam contiene la librería RichFaces/Ajax4JSf, que reduce enormemente el esfuerzo de los programadores con los componentes web.
- ✓ Hibernate: Para el mapeo y el acceso a la base de datos. Una de las principales ventajas del uso de Seam es que permite el control de sus componentes mediante anotaciones, lo cual reduce la cantidad de archivos XML de configuración.

JBoss Tools: Es un conjunto de herramientas para Eclipse que permiten el manejo de diferentes frameworks que facilitan el desarrollo de aplicaciones. Jboss Tools dispone de plugins que proporcionan soporte en Eclipse para Hibernate, JBoss AS, Drools, JBPM, JSF, (X)HTML, Seam, Smooks, JBoss ESB o JBoss Portal, entre otros. (15)

Jboss tools es una colección de plugins que se le añaden al IDE, los cuales le incorporan una serie de funcionalidades, además de poder ser añadidos a diferentes servidores de aplicación. Posee un editor gráfico para la configuración de archivos Seam y soporta la realización de pruebas de integración de Seam desde el Eclipse.

Dentro de los plugins están:

- ✓ Seam Tools: Incluye soporte para la integración de los componentes del framework Seam.
- ✓ RichFaces VE: Editor visual para componentes HTML, JSF y RichFaces.
- ✓ Hibernate Tools: Sirve de apoyo para la utilización de los componentes del framework Hibernate y para el mapeo con la base de datos.

Enterprise Java Bean (EJB): Es una arquitectura de componentes de servidor que simplifica el proceso de construcción de aplicaciones de componentes empresariales distribuidos en Java. Con su utilización es posible escribir aplicaciones escalables, fiables y seguras sin escribir código de infraestructura. La existencia de infraestructura permite un desarrollo más rápido de la parte servidora

Ajax4jsf: Es una librería open source que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript. Mediante este framework podemos variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. En definitiva Ajax4jsf permite dotar a las aplicaciones JSF de contenido mucho más profesional con muy poco esfuerzo. (16)

Java Server Faces (JSF): Es un framework orientado a la interfaz gráfica de usuario (GUI), facilitando el desarrollo de éstas, y que sin embargo, realiza una separación entre comportamiento y presentación, además de proporcionar su propio servlet como controlador, implementando así los principios del patrón de diseño Modelo-Vista-Controlador (MVC), lo que da como resultado un desarrollo más simple y una aplicación mejor estructurada.

RichFaces: Es una librería de componentes visuales para JSF. Posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF. Se integra perfectamente en el ciclo de vida de JSF, provee varias librerías de componentes como son Core Ajax y UI, así como administración avanzada de recursos como imágenes, código JavaScript y Hojas de Estilo en Cascada (CSS). (17)

Es posible crear interfaces de usuario de manera rápida y eficiente, basado en componentes que están listos para usar y son altamente configurables. Es, además, un proyecto que fue desarrollado con una arquitectura abierta para que fuera compatible con la mayor cantidad de entornos.

Facelets: Es un framework para plantillas (templates) centrado en la tecnología JSF (Java Server Faces), por lo cual se integran de manera muy fácil. JSP (Java Server Pages) y JSF no se complementan naturalmente, cuando se usan juntos ambos escriben output al response, pero lo hacen de una manera diferente: JSP crea output ni bien encuentra código JSP (es decir procesa los elementos de la página de arriba a abajo), mientras que JSF dicta su propio re-rendering (ya que su ciclo de vida está dividido en fases marcadas). Facelets llena este vacío entre JSP y JSF, siendo una tecnología centrada en crear árboles de componentes y estar relacionado con el complejo ciclo de vida JSF.

Asynchronous JavaScript and XML (AJAX): Es una técnica de desarrollo web que genera aplicaciones web interactivas combinando: XHTML y CSS para la presentación de información, Document Object Model (DOM) para visualizar dinámicamente e interactuar con la información presentada, XML, XSLT para intercambiar y manipular datos, XMLHttpRequest para recuperar datos asíncronamente y Javascript como nexo de unión de todas estas tecnologías. Permite que las aplicaciones sean más interactivas, reduce el tamaño de la información intercambiada y actualiza porciones de la página en vez de la página completa, lo que significa aumentar la velocidad y usabilidad en las aplicaciones. (18)

Hibernate: Herramienta que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación mediante archivos declarativos XML o anotaciones en los beans (componente que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java) de las entidades que permiten establecer estas relaciones. Es software libre, distribuido bajo los términos de las licencias GNU y Licencia Pública General Reducida (LGPL por sus siglas en inglés, Lesser General Public License). (19)

Hojas de estilo en cascada (por sus siglas en inglés CSS): Se utilizan para lograr una apariencia uniforme, haciendo que los códigos HTML sean más fáciles de leer ya que los estilos se definen por separado, y con ello que las páginas se carguen más rápido ya que hay menos cantidad de HTML en cada página. (20)

Java Persistence API (JPA): Proporciona un modelo de persistencia basado en POJO's para mapear bases de datos relacionales en Java. El Java Persistence API fue desarrollado por el grupo de expertos de EJB 3.0 como parte de JSR 220(Java Specification Requests), aunque su uso no se limita a los

componentes software EJB. También puede utilizarse directamente en aplicaciones Web y aplicaciones clientes; incluso fuera de la plataforma Java EE.
En su definición, se han combinado ideas y conceptos de los principales frameworks de persistencia como Hibernate, Toplink y JDO, y de las versiones anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA. (21)

Plataforma de programación Java Enterprise Edition (Java EE): Son un conjunto de especificaciones que facilitan el desarrollo y despliegue de aplicaciones empresariales multi-capa. Java EE ofrece un conjunto de especificaciones y técnicas que proporcionan soluciones completas, seguras, estables y escalables para el desarrollo, despliegue y gestión de aplicaciones de múltiples niveles de funcionalidad basadas en servidores. Se reduce el costo y complejidad de desarrollo, lo cual resulta en servicios que se pueden desplegar y extender fácilmente. (22)

Java Runtime Environment (JRE): Se corresponde con un conjunto de utilidades que permite la ejecución de programas java sobre todas las plataformas soportadas.

1.4.3. Lenguaje de Modelado

Unified Modeling Language (UML): es un lenguaje que se utiliza para especificar, visualizar, construir y documentar los artefactos de sistemas intensivos de software. UML es gratuito, accesible a todos, y conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelamiento de sistemas grandes y complejos. (23)

1.4.4. Lenguaje de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.(24)

Java: "Un lenguaje simple. Orientado al objeto, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, multihilos y dinámico". Basado en el lenguaje C++ pero donde se eliminan muchas de las características OOP que se utilizan esporádicamente y que creaban frecuentes

problemas a los programadores. Esta eliminación de causas de error y problemas de mantenimiento facilita y reduce el coste del desarrollo de software. (25)

1.4.5. Herramientas

El manejo de la información en la actualidad exige el desarrollo de un conjunto de habilidades que permitan definirla, analizarla y aprovecharla; así como también el dominio de las herramientas informáticas para su presentación y comunicación con mayor rapidez, reduciendo el esfuerzo.

Las herramientas informáticas (tools, en inglés), son programas, aplicaciones o simplemente instrucciones usadas para efectuar otras tareas de modo más sencillo. (26) Para su uso es necesario tener conocimiento de sus elementos, objetos que manejan, así como sus operaciones básicas, para de esta manera saber que se puede hacer con ellas.

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. El software de modelado UML contribuye a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (27)

Este trae incluidos los objetos más recientes de UML, diagramas de casos de uso, diagramas de componentes, diagramas de clase, reversa instantánea para Java, C++, XML y XML Schema, brinda soporte para Rational Rose, integración con MicrosoftVisio, y posibilita generar reportes y documentación en HTML/PDF. Se integra con IDE’s como NetBeans (de Sun Microsystems), JDeveloper (de Oracle), Eclipse (de IBM), JBuilder (de Borland).

Eclipse: Es un entorno de desarrollo integrado, Integrated Development Enviroments (IDE), cedido por IBM. Su misión consiste en evitar tareas repetitivas, facilitar la escritura de código correcto, disminuir el tiempo de depuración e incrementar la productividad del desarrollador.

JBossServer: Es el servidor de aplicaciones de código abierto más utilizado actualmente en todo el mundo. Este servidor de aplicaciones se encuentra certificado J2EE y soporta sistemas de gran

complejidad y alta concurrencia. Al estar basado en Java, puede ser utilizado en cualquier sistema operativo que lo soporte. Implementa todo el paquete de servicios de J2EE.

Las características destacadas de JBoss incluyen:

- ✓ Producto de licencia de código abierto sin coste adicional
- ✓ Confiable a nivel de empresa.
- ✓ Orientado a arquitectura de servicios.
- ✓ Flexibilidad consistente.
- ✓ Servicios del middleware para cualquier objeto de Java.
- ✓ Ayuda profesional 24x7 de la fuente.
- ✓ Soporte completo para JMX.

Es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, puede ser descargado, utilizado, instalado, y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores de software independientes, así como para grandes empresas.

pgAdmin III: Es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Facilita enormemente la administración e incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, entre otras. (28)

1.4.6. Sistema Gestor de Bases De Datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: DataBase Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras

de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. (29)

PostgreSQL: Es un poderoso SGBD de código abierto que ha ganado una fuerte reputación debido a su rentabilidad, ingeniería de datos, exactitud y fácil administración. Soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. y permite la creación de tipos propios. Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.Posee interfaces de programación para lenguajes como C, C++, Java, Perl y Python así como una documentación excepcional. (30)

Con el estudio de los principales términos y conceptos relacionados con la gestión de ensayos clínicos se logra una mayor comprensión del negocio. Se identificaron las principales deficiencias que presentan los sistemas homólogos en el ámbito tanto nacional como internacional, facilitando la definición de las nuevas funcionalidades a implementar, así como la descripción de las tecnologías y herramientas para ello partiendo de sus características y ventajas.

CAPÍTULO 2: DESCRIPCIÓN DEL SISTEMA

Para el desarrollo de un sistema es necesario realizar un análisis de sus características, para comprender el dominio del problema a resolver. En ello juega un papel importante el modelo conceptual, el cual sirve de base para la realización de los diagramas de clases y la obtención de los requerimientos funcionales y no funcionales con mayor calidad. Luego es necesario elaborar el diagrama de casos de uso, y la descripción detallada de cada uno de ellos, ya que en él se exponen las funcionalidades con las que contará el sistema.

2.1. Propuesta de solución

Luego de analizados los inconvenientes que posee el sistema Clínicas 1.0, el grupo de desarrollo decidió realizar una nueva versión denominada SIGEC (Sistema de Gestión de Ensayos Clínicos), el cual hará uso del ORM Hibernate, así como de tecnologías para el enriquecimiento de las vistas de los usuarios, ofreciendo un producto más amigable al cliente. Por tanto, para dar solución al problema de investigación planteado, se propone incorporar a este sistema en el módulo Conducción funcionalidades que permitan la gestión de los sujetos, los cronogramas específicos y las hojas CRD asociadas a ellos, abarcando las funcionalidades presentes en el Módulo Enviar Datos del Clínicas 1.0 además de otras como: Eliminar sujeto, Modificar sujeto, Deshabilitar y Habilitar sujeto, Modificar Momento de Seguimiento.

2.2. Modelo conceptual

Explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción del problema. También conocido como modelo de dominio.

Para la realización del modelo conceptual se especifican los **conceptos** de mayor importancia que sirven de base a la solución propuesta, partiendo de la información obtenida en la entrevista realizada al Ing. Erislán Martínez Jera, principal desarrollador del sistema Clínicas 1.0, la cual puede ser consultada en el Anexo 1.

CIC: Coordinador de la Investigación Clínica.

Cronograma Específico: Contiene una planificación de todos los momentos de seguimiento, definidos para un sujeto a partir de una fecha indicada por el CIC y la planificación de cada momento del estudio en el cronograma general.

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

Hoja del Cuaderno de Recogida de Datos (CRD): Modelo o Formulario que agrupa un conjunto de datos del sujeto de interés para el estudio.

Sujeto: Persona asociada a un estudio.

Estado: Representa las fases por las que pasan todos los conceptos identificados como los estudios, cronogramas y reglas, estos pueden ser:

- En el caso del estudio (Diseño o Ejecución).
- En el caso del cronograma (Completado o Iniciado).
- En el caso de las Reglas (Completado o Iniciado).

Investigador Principal: Es el responsable máximo de toda la información clínica recogida en su sitio de investigación y además es el encargado en el sistema de insertar y retirar su firma de las hojas de CRD completadas. Puede modificar las hojas y gestionar notas de Monitoreo, de sitio y generales.

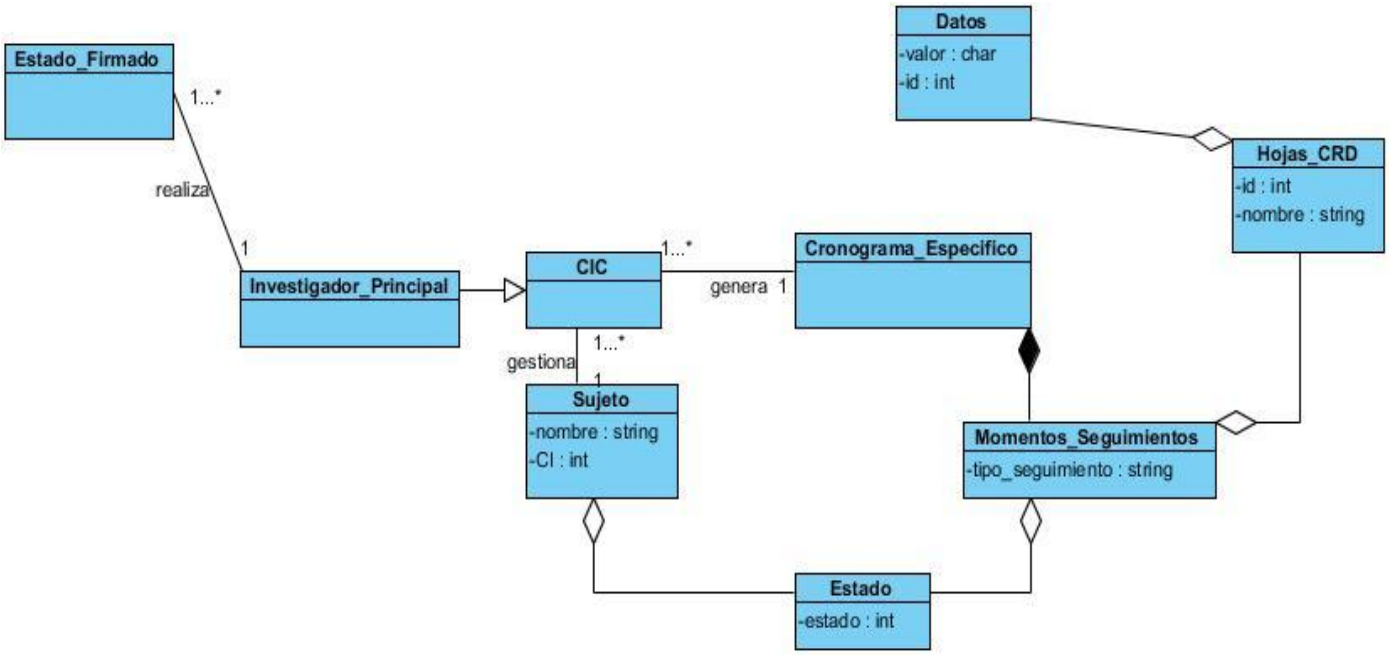

Figura 1. Modelo conceptual del Módulo Conducción

2.3. Requerimientos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares.(31)

- ✓ RF 1 Insertar un sujeto
- ✓ RF 2 Mostrar datos del sujeto
- ✓ RF 3 Modificar los datos de un sujeto
- ✓ RF 4 Desactivar sujeto
- ✓ RF 5 Restaurar sujeto
- ✓ RF 6 Establecer estado de inclusión del sujeto
- ✓ RF 7 Cambiar sujeto de centro
- ✓ RF 8 Interrumpir sujeto
- ✓ RF 9 Eliminar sujeto
- ✓ RF 10 Visualizar listado de sujetos
- ✓ RF 11 Guardar datos en una hoja CRD
- ✓ RF 12 Mostrar datos de una hoja CRD

- ✓ RF 13 Modificar datos de una hoja CRD
- ✓ RF 14 Firmar hoja CRD
- ✓ RF 15 Eliminar firma de hoja CRD
- ✓ RF 16 Visualizar listado de hojas CRD
- ✓ RF 17 Generar cronograma específico
- ✓ RF 18 Visualizar cronograma específico de un sujeto
- ✓ RF 19 Modificar automáticamente el cronograma a partir de una fecha
- ✓ RF 20 Adicionar MS no programado en el cronograma específico
- ✓ RF 21 Eliminar MS no programado en el cronograma específico
- ✓ RF 22 Visualizar MS
- ✓ RF 23 Visualizar cantidad de MS por sujetos
- ✓ RF 24 Modificar MS en el cronograma específico
- ✓ RF 25 Mostrar estado de monitoreo de los sujetos
- ✓ RF 26 Imprimir MS por sujetos
- ✓ RF 27 Imprimir cronograma específico

2.4. Requerimientos no funcionales

Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como son la fiabilidad, el soporte y la capacidad de almacenamiento. Imponen restricciones en el diseño o la implementación o Estándares de Calidad. Son propiedades o cualidades que el producto debe tener. (31)

- **Usabilidad**
  - ✓ RNF 1: La aplicación web tendrá un ambiente sencillo y fácil de manejar para los usuarios.
  - ✓ RNF 2: 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 o Sistemas Operativos Unix, Memoria RAM 4GB, 500 GB de Disco Duro.
  - ✓ RNF 3: 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, o Sistemas Operativos Unix, Memoria RAM 4GB y 120 GB de Disco Duro. Además, como servidor web JBoss4.2.2.

- ✓ RNF 4: Las PC Clientes deberán contar con las siguientes características: Procesador Intel Pentium IV o superior, Sistema Operativo Windows 7 o superior o Sistemas Operativos Unix, Memoria RAM 1GB y 80 GB de disco duro.
- ✓ RNF 5: El sistema podrá ser consultado desde los siguientes navegadores: Mozilla Firefox 3.6 o superior y Google Chrome 1.4 o superior.
- ✓ RNF 6: Debe estar instalada la máquina virtual de Java: java-7-openjdk para GNU Linux, o la jdk-6u3 o superior para Windows.

- **Confiabilidad**
  - ✓ RNF 7: La aplicación tendrá un sistema de trazas que registran el flujo constante de los datos y los responsables de sus cambios.
  - ✓ RNF 8: 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.
  - ✓ RNF 9: Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos.

- **Interfaz**
  - ✓ RNF 10: Las páginas principales tendrán información que servirá de guía al usuario.
  - ✓ RNF 11: Las páginas no estarán cargadas de imágenes.
  - ✓ RNF 12: Cada rol tiene acceso a la interfaz que se corresponda con los permisos asignados.
  - ✓ RNF 13: Se hará uso de simbología mediante íconos para indicar el estado de los elementos utilizados en el diseño. Además, los íconos contendrán funcionalidades específicas.

- **Restricciones de diseño**
  - ✓ RNF 14: Se usa como herramienta CASE Visual Paradigm para el modelado de los productos típicos de trabajo generados en cada fase del ciclo de vida.
  - ✓ RNF 15: Se usa como lenguaje de programación Java.
  - ✓ RNF 16: Se usa como Gestor de Base de Datos PostgreSQL.

- **Estándares aplicables**
  - ✓ RNF 17: 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.

2.5. Diagrama de Casos de Uso

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de *casos de uso* y los actores participantes en el proceso. Sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente; y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen *qué* es lo que debe hacer el sistema, pero no *cómo*. (32)

Actor del sistema

Un actor es una entidad externa al sistema que se modela y que puede interactuar con él. Puede ser una persona o un grupo de personas homogéneas, otro sistema, o una máquina. Por lo tanto, al identificarlos, estamos comenzando a delimitar el sistema y a definir su alcance. (33)

Actor	Objetivos
Usuario	Realizar todas las acciones en el módulo una vez que se haya registrado en el sistema.

Tabla 1. Descripción del actor del sistema

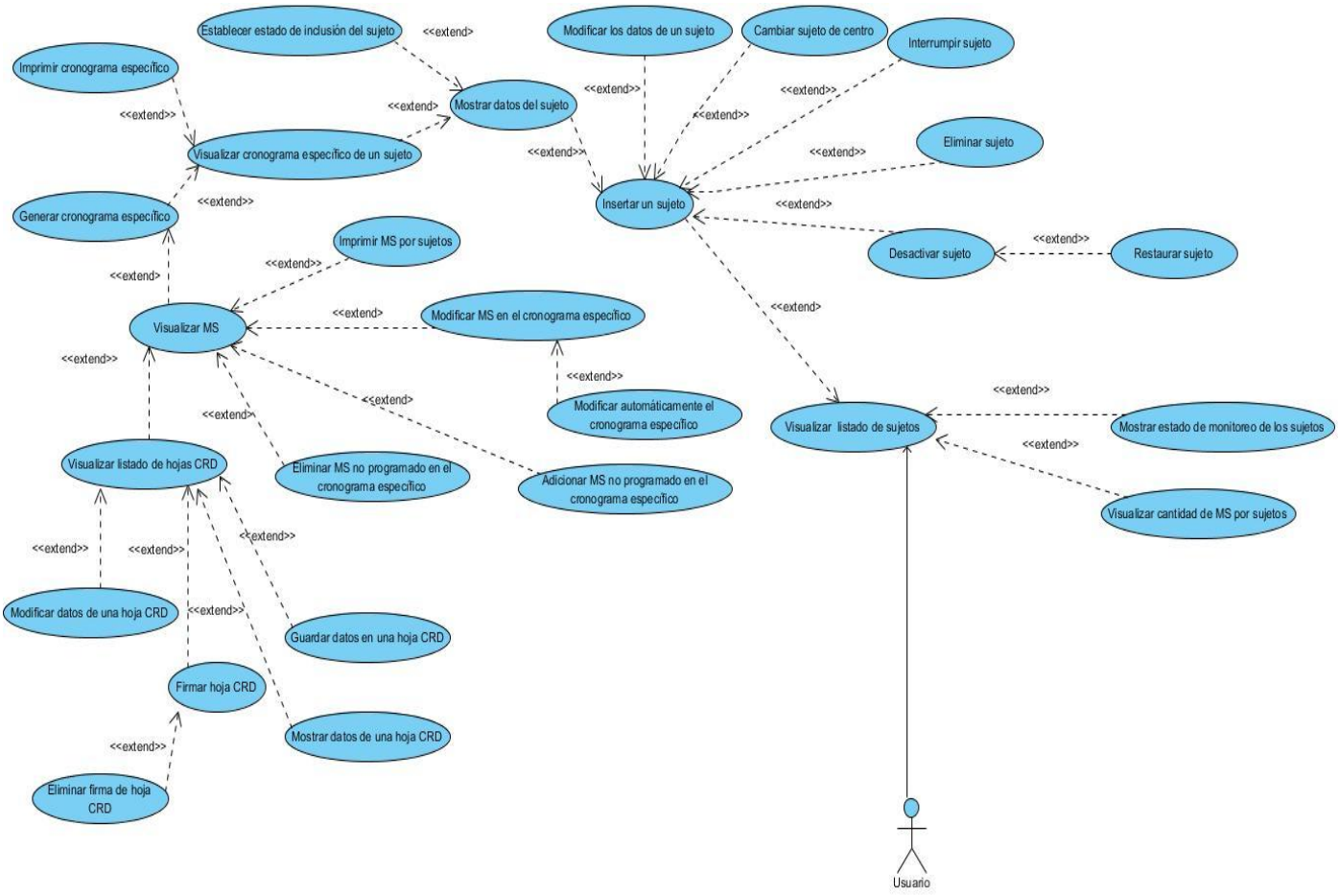

Figura 2. Diagrama de casos de uso del Módulo Conducción

2.6. Especificación de CU

A continuación se muestran las especificaciones de los casos de uso Modificar los datos de un sujeto e Insertar sujeto. Las restantes especificaciones de casos de uso, se pueden consultar en el Anexo 2.

Objetivo	Permitir al usuario modificar los datos de un sujeto.
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario indica la opción “Modificar sujeto” en un sujeto del listado de sujetos. El sistema muestra una

	interfaz con los datos del sujeto dando la posibilidad de modificarlos. El usuario introduce los datos que desea cambiar y el sistema guarda los cambios ocurridos.
Complejidad	Baja
Prioridad	Media
Precondiciones	Debe haber al menos un sujeto en un estudio.
Postcondiciones	Quedan modificados los datos de un sujeto.

Flujo de eventos

Flujo básico:

	Actor	**Sistema**
1.	Indica la opción “Modificar sujeto” en un sujeto del listado de sujetos.	
2.		Muestra una interfaz con los datos del sujeto y brinda la posibilidad de modificarlos. Los datos son: ✓ ID de la persona ✓ OID ✓ Fecha de nacimiento ✓ Sexo ✓ Fecha de inclusión
3.	Introduce los datos que desea modificar.	
4.		Verifica que los datos están correctos. La fecha de nacimiento debe ser menor que la fecha de

		inclusión del sujeto.
5.		Modifica los datos del sujeto.
6.		Finaliza el caso de uso.
Flujos alternos		
Nº Evento:4		
	Actor	**Sistema**
1		Detecta que la fecha de nacimiento no es menor que la fecha de inclusión del sujeto. Muestra un mensaje informando que no se pueden realizar los cambios por estos motivos.
2		Finaliza el caso de uso.

Relaciones	**CU Incluidos**	
	CU Extendidos	

Tabla 2. Especificación CU Modificar los datos de un sujeto

Objetivo	Permitir al usuario insertar un sujeto.
Actores	Usuario
Resumen	El caso de uso inicia cuando se decide insertar un nuevo sujeto, el sistema muestra los datos a llenar y cuando estos han sido llenados, los guarda.
Complejidad	Baja

Prioridad	Media	
Precondiciones	Debe haber al menos un estudio activo	
Postcondiciones	Queda insertado el sujeto.	
Flujo de eventos		
Flujo básico:		
	Actor	**Sistema**
1.	Indica insertar un sujeto en el estudio.	
2.		Muestra una interfaz con los datos: identificador del sujeto en el estudio, fecha de inclusión, sexo y fecha de nacimiento, que deben de ser introducidos por el usuario.
3.	Introduce los datos requeridos e indica guardarlos.	
4.		Revisa los datos introducidos, si son válidos añade el sujeto en el sistema y le asigna como estado de tratamiento del sujeto el de “Evaluación.
5.		Finaliza el caso de uso.
Flujos alternos		

<table>
<tr><td colspan="4">Nº Evento:3</td></tr>
<tr><td></td><td colspan="2">Actor</td><td>Sistema</td></tr>
<tr><td>1.</td><td colspan="2">Cancela la opción de insertar un sujeto.</td><td></td></tr>
<tr><td>2.</td><td colspan="2"></td><td>Re direcciona al listado de sujetos.</td></tr>
<tr><td>3.</td><td colspan="2"></td><td>Finaliza el caso de uso.</td></tr>
<tr><td rowspan="2">Relaciones</td><td>CU Incluidos</td><td colspan="2"></td></tr>
<tr><td>CU Extendidos</td><td colspan="2">Modificar los datos de un sujeto

Mostrar los datos de un sujeto

Cambiar sujeto de centro

Interrumpir sujeto

Eliminar sujeto

Desactivar sujeto</td></tr>
</table>

Tabla 3. Especificación CU Insertar sujeto

Con la definición de los requerimientos funcionales y no funcionales se logra alcanzar un mejor entendimiento de las características con las que deberá contar el sistema, para satisfacer las necesidades del cliente, los cuales son modelados a partir del diagrama de casos de uso. Además, la creación del modelo conceptual favoreció a que se enmarcaran los conceptos asociados al dominio del problema, y por ende se comprendiera cómo se realiza la conducción de ensayos clínicos.

CAPÍTULO 3: DISEÑO DEL SISTEMA

El diseño es una etapa de suma importancia en la realización del software, pues es donde se moldea lo que se quiere construir, por lo cual es necesario modelar los artefactos que contribuyan a la implementación del mismo. En el presente capítulo se muestran los principales diagramas de clases, el diagrama de paquetes, y se describe la arquitectura del sistema.

3.1. Descripción de la arquitectura

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. (34) La solución propuesta presenta la arquitectura Cliente-Servidor.

Arquitectura Cliente-Servidor

La arquitectura Cliente–Servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Es la combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos a compartir. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, Módem, etc. Con el uso de dicha arquitectura se facilita la integración entre sistemas diferentes, se comparte información y se favorece el uso de interfaces gráficas interactivas. Además su estructura inherentemente modular facilita la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

3.1.1. Patrón arquitectónico: Modelo Vista Controlador (MVC)

El patrón arquitectónico Modelo Vista Controlador es un estilo que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo lo constituyen las clases que representan los datos y permiten interactuar con el

Sistema de Gestión de Bases de Datos, a través de una lógica definida que es regulada por los controladores como responsables de recibir los eventos.

Descripción

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. No tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario.
- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

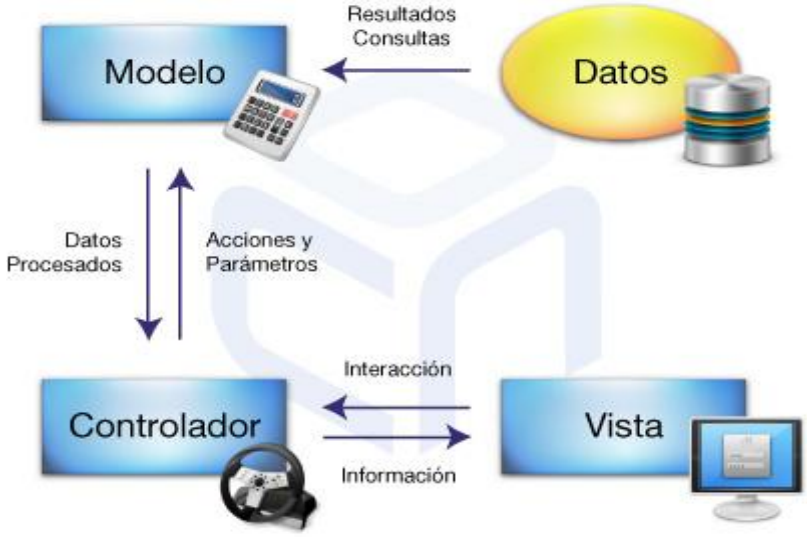

Figura 3. Representación del patrón MVC

Ventajas de utilizar MVC

Una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc. sin alterar la lógica de negocio. La separación de capas como presentación, lógica de negocio, acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

• Agregar nuevas vistas.

• Agregar nuevas formas de recolectar las órdenes del usuario (interpretar sus modelos mentales).

• Modificar los objetos de negocios bien sea para mejorar el comportamiento o para migrar a otra tecnología.

• Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones solo se deben hacer en un solo lugar y no en varios como sucedería si se tuviera una mezcla de presentación e implementación de la lógica del negocio.

• Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se paralice. Adicionalmente el patrón MVC propende a la especialización de cada rol del equipo, por tanto en cada liberación de una nueva versión se verán los resultados. (35)

3.1.2. Patrones de diseño

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.” En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. (38)

Ventajas del uso de patrones de diseño

Proporcionan una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. Así la incorporación de un nuevo programador, no requerirá conocimiento de lo realizado anteriormente por otro. Permiten tener una estructura de código común a

todos los proyectos que implemente una funcionalidad genérica. La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software. (39)

En el desarrollo de la solución se utilizan los patrones para la asignación de responsabilidades (General Responsibility Assignment Software Patterns) más conocidos como GRASP, entre ellos el patrón experto, creador, alta cohesión, bajo acoplamiento y el controlador, con el objetivo de describir los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Experto

- ✓ Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?
- ✓ Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- ✓ Explicación: Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Da origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican a la cosa real que representa, por lo que ofrece una analogía con el mundo real.
- ✓ Beneficios: Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener.

Este patrón se evidencia en la clase controladora Gestionarsujeto la cual contiene la información necesaria para el trabajo con los sujetos.

Creador

- ✓ Problema: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?
- ✓ Solución: Asignarle a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos:
    - B agrega los objetos A.
    - B contiene los objetos A.

- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán trasmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).
- B es un creador de los objetos A.
- Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

✓ Explicación: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

✓ Beneficios: Brinda un soporte a un bajo acoplamiento, lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

Se evidencia en la clase CrearSujeto_conduccion la cual es la encargada de crear instancias u objeto de la clase Subject.

Bajo Acoplamiento

✓ Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente, ya que presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

✓ Solución: Asignar una responsabilidad para mantener bajo acoplamiento.

✓ Explicación: El bajo acoplamiento es un principio que se debe tener siempre en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los

cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

- ✓ Beneficios: Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

Este patrón es evidenciado,por ejemplo, en las clases VerSujeto_conduccion y ModificarSujeto_conduccion.

Alta Cohesión

- ✓ Problema: ¿Cómo mantener la complejidad dentro de límites manejables? En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:
  - Son difíciles de comprender.
  - Son difíciles de reutilizar.
  - Son difíciles de conservar.
  - Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

- ✓ Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta.
- ✓ Explicación: El patrón Alta Cohesión es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande.

- ✓ Beneficios: Con el uso de este patrón mejoran la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en funcionalidad. A menudo se genera un bajo acoplamiento. La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Este patrón es evidenciado en todas las clases que contiene el sistema, pues se poseen responsabilidades que tienen relación con ellas.

Controlador

- ✓ Problema: ¿Quién debería encargarse de atender un evento del sistema? Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema.

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

- ✓ Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:
  - El “sistema” global (controlador de fachada).
  - La empresa u organización global (controlador de fachada).
  - Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
  - Un manejador artificial de todos los eventos del sistema de un caso de uso (controlador de casos de uso).
- ✓ Explicación: La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. Otros medios de entrada son los mensajes externos o las señales procedentes de sensores como sucede en los sistemas de control de procesos. En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La misma clase

controladora debería utilizarse con todos los eventos sistémicos de un caso de uso, de modo que se pueda conservar la información referente al estado del caso.

- ✓ Beneficios: Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Al delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio favorece la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras. (40)

El uso de este patrón es evidenciado en las clases controladoras creadas para cada caso de uso.

Diseño

El diseño tiene un papel importante en el desarrollo de un software, ya que permite que ingenieros de software produzcan modelos distintos que moldean el plano de la solución a ser implementada. Visto como un proceso, el diseño de software es la actividad de ciclo de vida de ingeniería de software en la que los requerimientos de software son analizados para causar una descripción de la estructura interna del software que servirá como base para su construcción. (36)

3.2. Modelo del diseño

Es una abstracción del Modelo de implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a la implementación. Representa a los casos de uso en el dominio de la solución. Puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo. (37)

3.2.1. Definición de elementos del diseño

El sistema se modela utilizando las clases UML estereotipadas “Server Page”, “Client Page” y “Form” empleadas para el código servidor, código cliente y formularios respectivamente.

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

Tabla 4. Clases de diseño estereotipadas

3.2.2. Diagramas de clases de diseño

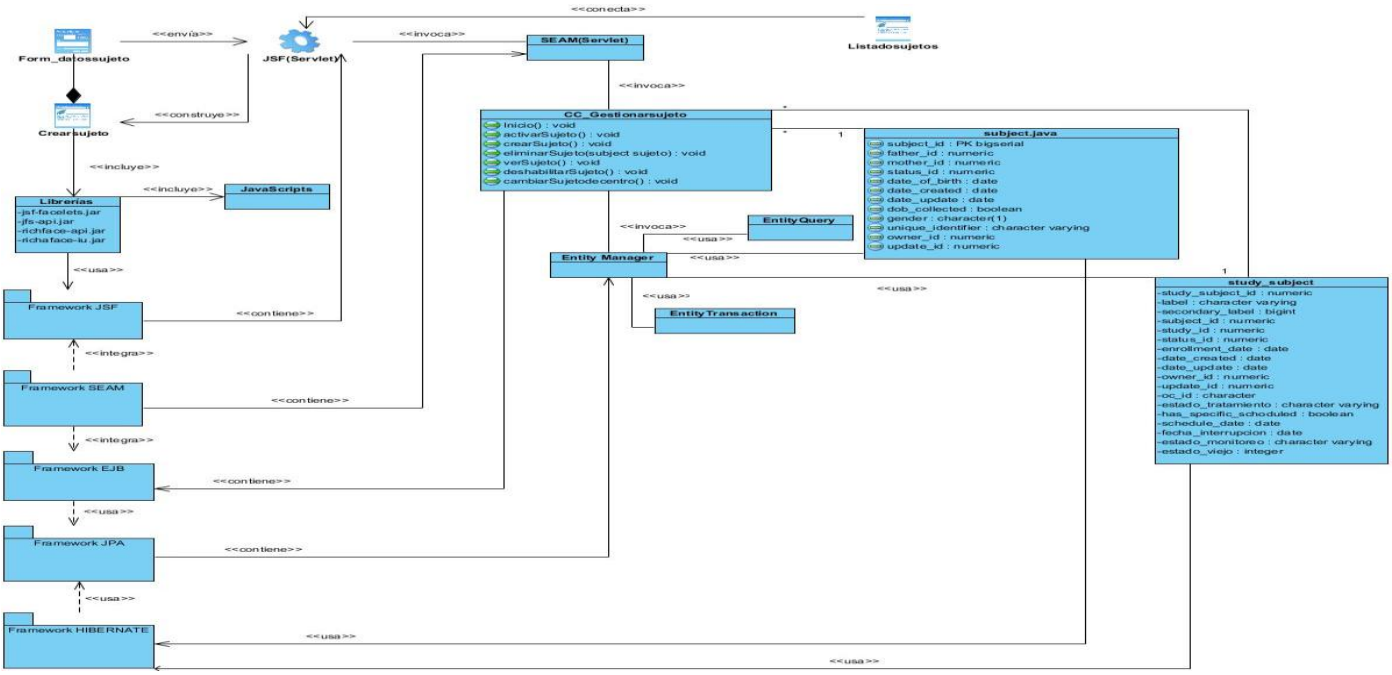

Figura 4. Modelo de clases de diseño del CU Insertar sujeto

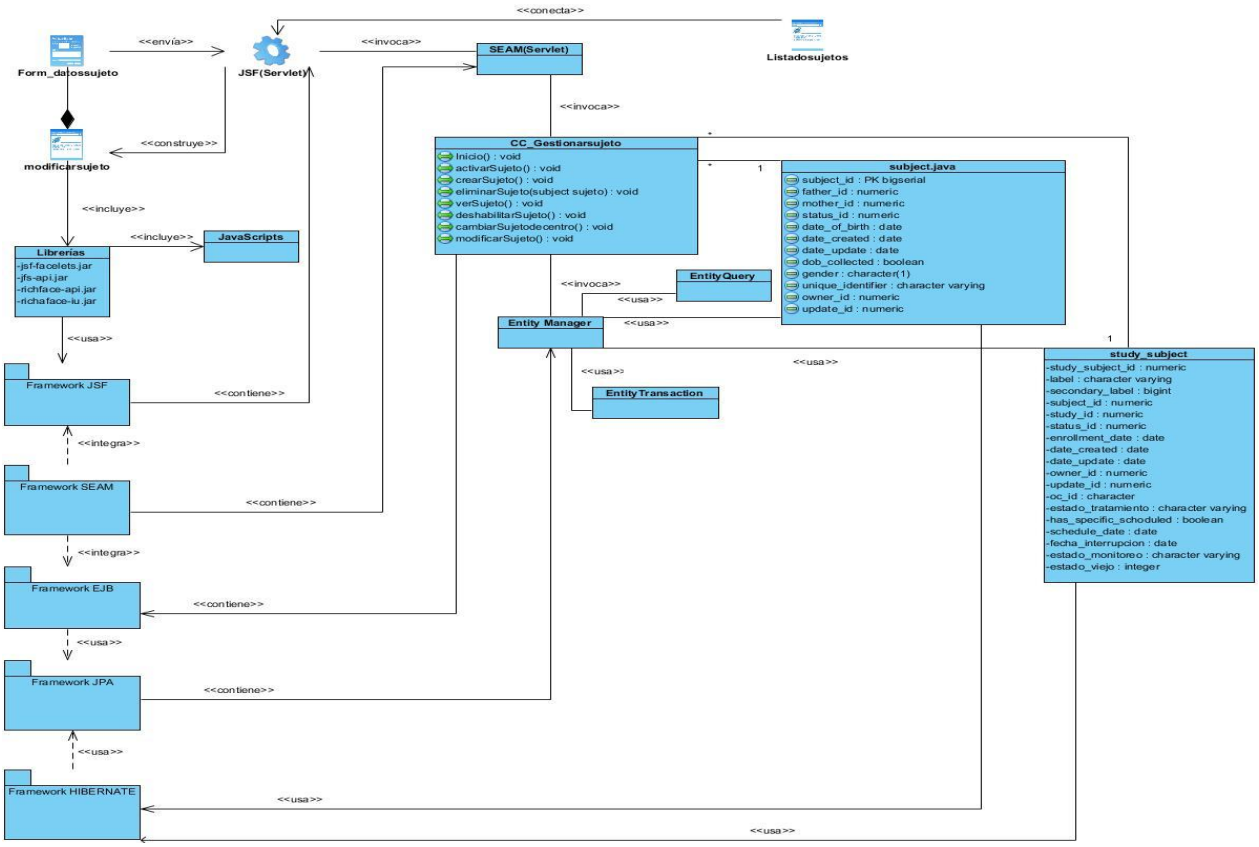

Figura 5. Modelo de clases de diseño del CU Modificar datos de un sujeto

3.3. Diagrama de paquetes

Muestra cómo está estructurado el sistema. Cada paquete puede contener otros paquetes o clases, que tienen interfaces o realizan cierta funcionalidad.

Entidades: formado por las clases autogeneradas especificadas en el diseño, en dependencia de las tecnologías usadas en la implementación y las personalizadas. Estas clases se autogeneran desde la base de datos utilizando el ORM Hibernate (generación de objetos java desde la base de datos) y personalizadas son aquellas que se modifican para una mejor gestión de la información, por lo que pueden heredar de las autogeneradas.

Sesiones: integrada por las clases controladoras autogeneradas por el entorno de desarrollo, por las clases controladoras personalizadas y por las controladoras del proceso.

Vistas: formada por los contenidos web referentes a las páginas clientes y los formularios que las componen, además de las vistas que interactúan con el usuario.

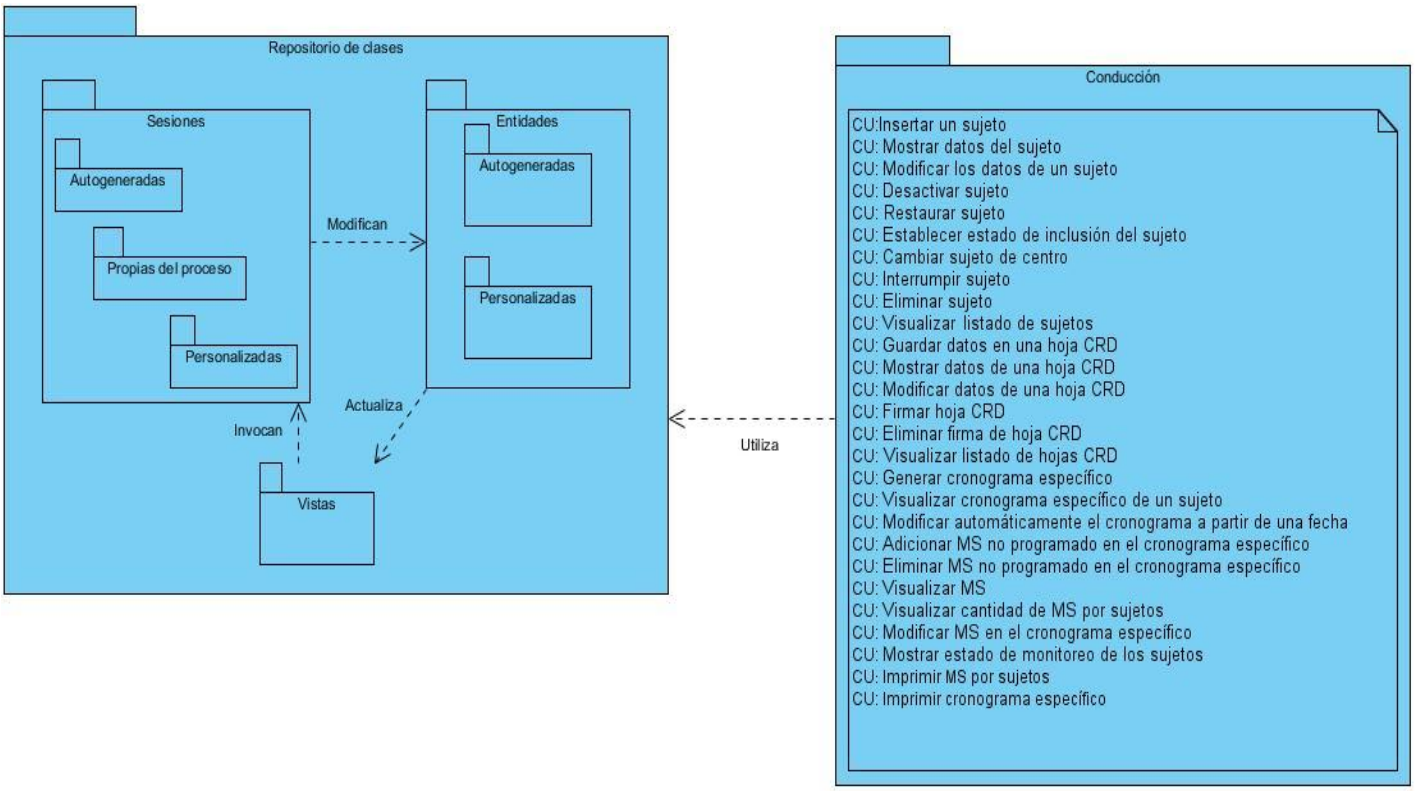

Figura 6. Diagrama de Paquetes del Módulo Conducción

Con el desarrollo del diagrama de paquetes se da a conocer la estructura y división en agrupaciones lógicas del sistema. Mientras que con la realización de los diagramas de diseño queda expuesta la arquitectura definida, evidenciando las clases con las que contará para su correcto funcionamiento. Además se describen los patrones de arquitectura y diseño a utilizar.

CAPÍTULO 4: IMPLEMENTACIÓN

El flujo de trabajo Implementación definido por la metodología de desarrollo RUP, describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. En este capítulo se presenta el modelo de datos obtenido y la descripción de las tablas que forman parte de este. Así como los principales artefactos relacionados con este flujo de trabajo: los diagramas de despliegue y de componentes.

4.1. Modelo de datos

Un modelo de datos es un conjunto de conceptos, reglas y convenciones que proporciona una representación visual y física de los datos persistentes del sistema, que finalmente formarán las tablas de la base de datos.

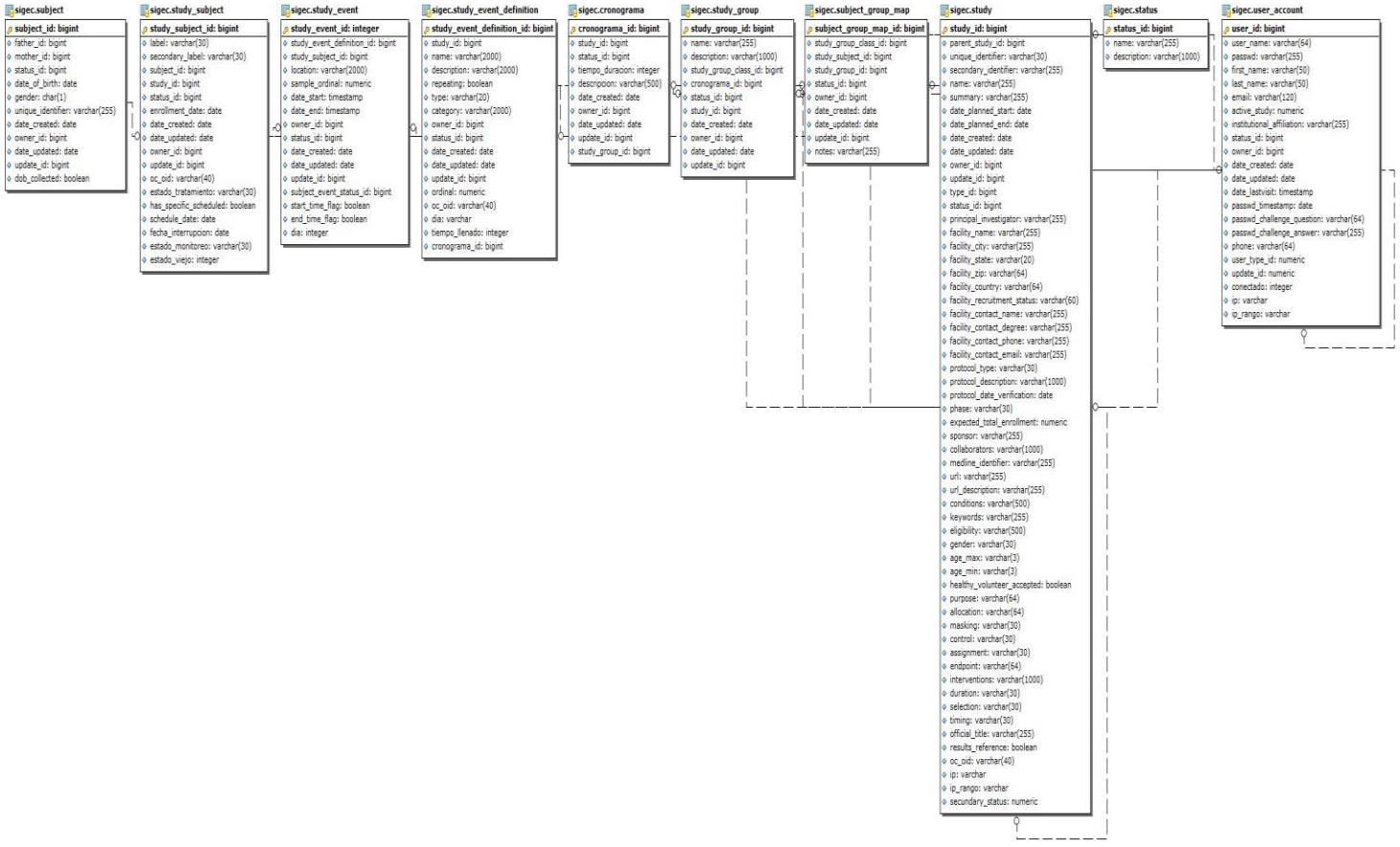

Figura 7. Modelo de datos del Módulo Conducción

4.1.1. Descripción de las tablas

La descripción de las tablas que forman parte del modelo de datos permite la comprensión de los diagramas que son resultado del diseño de la base de datos del sistema. En ella se encuentra la información más detallada de las clases y atributos que en su conjunto conformarán la base de datos. A continuación se muestran las descripciones:

Nombre: study		
Descripción		En esta entidad se insertan los datos de centros y estudios después de creados, además cuando se muestra el listado de estudios y centros, se cargan los datos a mostrar de dicha tabla, así como otras funcionalidades con respecto a centros y estudios.
Atributo	**Tipo**	**Descripción**
study_id	Bigint	Representa el identificador único del estudio.
parent_study_id	Bigint	Indica si el estudio tiene un padre.
unique_identifier	varchar	Representa el Id único del protocolo del estudio.
name	varchar	Representa el titulo abreviado del estudio.
summary	varchar	Representa el resumen con respecto al estudio.
date_planned_start	date	Fecha de comienzo del estudio.
date_planned_end	date	Fecha de finalización del estudio.
date_created	date	Día de la creación del estudio.
date_updated	date	Día de la actualización del estudio.
update_id	bigint	Representa la entidad que realizó la actualización del estudio.
status_id	bigint	Estado en el que se encuentra el estudio.
principal_investigator	varchar	Nombre del Investigador principal que realiza el estudio.
facility_name	varchar	Nombre de la instalación donde se realiza el estudio.
facility_city	varchar	Nombre de la Ciudad / Municipio del Centro donde se realiza el estudio.
facility_state	varchar	Nombre del Estado / Provincia de la instalación donde se realiza el estudio.

facility_zip	varchar	Código Postal del centro donde se realiza el estudio.
facility_country	varchar	País del centro donde se realiza el estudio.
facility_contact_name	varchar	Nombre del contacto en el centro donde se realiza el estudio.
facility_contact_degree	varchar	Titulación del contacto en el centro donde se realiza el estudio.
facility_contact_phone	varchar	Teléfono del contacto en el centro donde se realiza el estudio.
facility_contact_email	varchar	Correo electrónico del contacto en el centro donde se realiza el estudio.
protocol_type	varchar	Representa el tipo de protocolo: observacional o intervencionista.
protocol_description	varchar	Descripción detallada del protocolo seleccionado.
phase	varchar	Representa la fase en la que se encuentra el estudio.
expected_total_enrollment	varchar	Representa el número esperado de inclusiones del estudio.
sponsor	varchar	Nombre del patrocinador del estudio.
collaborators	varchar	Nombre de los colaboradores del estudio.
medline_identifier	varchar	Identificador de MEDLINE.
url	varchar	URL de referencia del estudio.
url_description	varchar	Descripción de la URL del estudio.
conditions	varchar	Patología del estudio.
keywords	varchar	Palabras claves (separadas por coma) del estudio.
eligibility	varchar	Criterios de elegibilidad del estudio.
gender	varchar	Sexo del sujeto para la realización del estudio.
age_max	varchar	Edad máxima de los sujetos del estudio.
age_min	varchar	Edad mínima de los sujetos del estudio.
ealthy_volunteer_accepted	boolean	Representa el valor (si o no) existen voluntarios sanos para el estudio.
purpose	varchar	Representa si es un tratamiento, prevención diagnóstico o educacional/asesoramiento.
allocation	varchar	Representa si es un Estudio Clínico con Asignación Aleatoria o con Asignación no Aleatoria.
masking	varchar	Representa el tipo de enmascaramiento: abierto, ciego simple o

		doble ciego.
control	varchar	Contiene el tipo de control: Placebo, Activo, Sin Controles, Histórico o Comparación de Datos.
assignment	varchar	Representa el tipo de asignación: Grupo Único, Paralelo, Cruzado o Factorial.
endpoint	varchar	Punto final: Seguridad, Eficacia, Seguridad/Eficacia, Bio-Equivalencia, Bio-Disponibilidad, Farmacocinética, Farmacodinámica, Farmacocinética/Farmacodinámica.
interventions	varchar	Contiene el nombre del propósito (Ej. medicamento dipirona).
duration	varchar	Período de duración del estudio.
selection	varchar	Este campo representa el tipo de selección: Muestra por Conveniencia, Población Definida, Muestra Aleatoria, Control de Caso.
timing	varchar	Ajuste temporal del estudio.
official_title	varchar	Título oficial del estudio.
results_reference	boolean	Referencias de los resultados del estudio realizado.
oc_oid	varchar	Indica el identificador único del campo unique_identifier.
Ip	varchar	Representa la dirección IP asignada al estudio.
ip_rango	varchar	Representa el rango de IP asignado al estudio.

Tabla 5. Descripción de la tabla: study

Nombre: subject		
Descripción	Almacena la información referente a los sujetos.	
Atributo	**Tipo**	**Descripción**
subject_id	bigint	Representa el identificador del sujeto.
father_id	bigint	Representa el identificador del padre.
mother_id	bigint	Representa el identificador de la madre.
status_id	bigint	Representa el identificador del estado en que se encuentra el sujeto.
date_of_bird	date	Indica la fecha de nacimiento del sujeto.

gender	Char(1)	Género.
unique_identifier	varchar(255)	Representa el identificador único.
date_created	date	Indica la fecha de creación del sujeto.
owner_id	bigint	Identificador del usuario que insertó o modificó.
date_update	date	Fecha de actualización.
update_id	bigint	Identificador de actualización.

Tabla 6. Descripción de la tabla: subject

Nombre: study_subject		
Descripción	Contiene todos los “sujetos” que se deciden estudiar en un ensayo.	
Atributo	**Tipo**	**Descripción**
study_subject_id	bigint	Representa el identificador de estudio.
label	varchar(30)	Indica el nombre del estudio.
subject_id	bigint	Representa el identificador del sujeto incluido en el estudio.
study_id	bigint	Representa el identificador.
status_id	bigint	Representa el estado.
enrollement_date	date	Indica la fecha de inclusión
date_created	date	Indica la fecha de creación.
date_updated	date	Indica la fecha de actualización.
owner_id	bigint	Representa el identificador del usuario que insertó o modificó.
updated_id	bigint	Representa el identificador de actualización.
estado_tratamiento	varchar(30)	Presenta el estado en que se encuentra el tratamiento del sujeto.
fecha_interrupcion	date	Recoge la fecha de interrupción
estado_monitoreo	varchar(30)	Representa el estado de monitoreo del sujeto en el estudio.

Tabla 7. Descripción de la tabla: study_subject

Nombre: usser_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	bigserial	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	numeric	Representa el estudio activo al que pertenece el usuario.
institutional_affiliation	varchar	Institución a la que pertenece el usuario.
status_id	bigint	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	numeric	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 8. Descripción de la tabla: usser_account

Nombre: status		
Descripción	Esta entidad contiene los diferentes estados que tiene las entidades del sistema.	
Atributo	**Tipo**	**Descripción**
status_id	bigserial	Representa el identificador del estado.

Name	varchar	Indica el nombre del estado.
Description	varchar	Descripción del estado.

Tabla 9. Descripción de la tabla: status

Nombre: subject_group_map		
Descripción	Esta entidad contiene las asociaciones de los sujetos a los grupos de sujeto.	
Atributo	**Tipo**	**Descripción**
subject_group_map_id	bigint	Representa el identificador del sujeto asociado al grupo de sujeto.
study_subject_id	bigint	Representa el identificador del sujeto.
study_group_id	bigint	Representa el identificador del grupo de sujetos.
status_id	bigint	Indica el estado.
date_created	date	Especifica la fecha de creación.
date_updated	date	Especifica la fecha en que se actualizó.
owner_id	bigint	Representa el identificador del sujeto o usuario que insertó o modificó.
update_id	bigint	Representa el identificador para cada vez que se modifique.

Tabla 10. Descripción de la tabla: subject_group_map

Nombre: cronograma		
Descripción	Esta entidad contiene los datos del cronograma.	
Atributo	**Tipo**	**Descripción**
Cronograma_id	bigint	Representa identificador del cronograma.
status_id	bigint	Indica el estado del cronograma.
study_id	bigint	Recoge el identificador del estudio activo en el que se realiza el cronograma.
Tiempo_duracion	integer	Indica el tiempo que se estima durará el cronograma.
Descripcion	varchar(500)	Especifica la descripción del cronograma.
Study_group_id	bigint	Identificador del grupo de sujetos al cual pertenece dicho cronograma.
date_created	date	Especifica la fecha de creación.
date_updated	date	Especifica la fecha en que se actualizó.
owner_id	bigint	Representa el identificador del sujeto o usuario que insertó o modificó.
update_id	bigint	Representa el identificador para cada vez que se modifique.

Tabla 11. Descripción de la tabla: cronograma

Nombre: study_group		
Descripción	Esta entidad contiene los datos de los grupos de sujeto.	
Atributo	**Tipo**	**Descripción**
study_group _id	bigint	Representa el identificador del grupo.
name	varchar(255)	Indica el nombre del grupo.
description	varchar(1000)	Recoge una descripción del grupo de sujetos.
study_group_class_id	bigint	Indica el identificador de la clase a la que pertenece el grupo.
cronograma_id	bigint	Especifica el cronograma asociado al grupo de sujetos.
status_id	bigint	Indica el estado.
study_id	bigint	Identificador del estudio al que pertenece el grupo de sujetos.
date_created	date	Especifica la fecha de creación.
date_updated	date	Especifica la fecha en que se actualizó.
owner_id	bigint	Representa el identificador del sujeto o usuario que insertó o modificó.
update_id	bigint	Representa el identificador para cada vez que se modifique.

Tabla 12. Descripción de la tabla: study_group

Nombre: study_event_definition		
Descripción	Esta entidad contiene la definición de los momentos de seguimiento.	
Atributo	**Tipo**	**Descripción**
study_event_definition_id	bigint	Representa el identificador del momento de seguimiento.
name	varchar(2000)	Indica el nombre del momento de seguimiento.
description	varchar(2000)	Recoge una descripción del momento de seguimiento.
status_id	bigint	Indica el estado del momento de seguimiento.
study_id	bigint	Identificador del estudio activo en el que se realiza el momento de seguimiento.
type	varchar(20)	Especifica el tipo de momento de seguimiento.
date_created	date	Especifica la fecha de creación.
date_updated	date	Especifica la fecha en que se actualizó.

owner_id	bigint	Representa el identificador del sujeto o usuario que insertó o modificó.
update_id	bigint	Representa el identificador para cada vez que se modifique el momento de seguimiento.
ordinal	numeric	Representa el identificador que tiene un momento de seguimiento en un estudio determinado.
dia	varchar	Días en que se realizarán los momentos de seguimiento.
cronograma_id	bigint	Identificador del cronograma al cual pertenecen de los momentos de seguimiento.
tiempo_llenado	integer	Especifica el tiempo de llenado de una hoja CRD en el momento de seguimiento.

Tabla 13. Descripción de la tabla: study_event_definition

Nombre: study_event		
Descripción	Esta entidad contiene los datos de los momentos de seguimiento asociados a la conducción de un estudio.	
Atributo	**Tipo**	**Descripción**
study_event_id	integer	Identificador del momento de seguimiento en la conducción del estudio.
study_event_definition_id	bigint	Representa el identificador del momento de seguimiento en la etapa de diseño.
location	varchar(2000)	Indica la localización.
sample_ordinal	numeric	Representa el identificador que tiene un momento de seguimiento en un estudio determinado.
date_start	timestamp	Especifica la fecha de inicio.
date_end	timestamp	Especifica la fecha de fin
status_id	bigint	Indica el estado del momento de seguimiento.
date_created	date	Especifica la fecha de creación de la hoja CRD.
date_updated	date	Especifica la fecha en que se actualizó la hoja CRD.
owner_id	bigint	Representa el identificador del sujeto o usuario que insertó o

		modificó la hoja CRD.
update_id	bigint	Representa el identificador para cada vez que se modifique la hoja CRD.
subject_event_status_id	bigint	Representa el identificador del estado que tiene un momento de seguimiento en un estudio determinado.
dia	varchar	Días en que se realizarán los momentos de seguimiento.

Tabla 14. Descripción de la tabla: study_event

4.2. Implementación

Objetivos de la implementación

- Definir la organización del código, en términos de los subsistemas de implementación organizados en capas.
- Implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- Probar y desarrollar componentes como unidades.
- Integrar los resultados producidos por los implementadores individuales (o equipos) en un sistema ejecutable.
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.

4.2.1. Diagrama de despliegue

Es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue (distribución) de los artefactos del software en los destinos de despliegue. Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son archivos ejecutables, bibliotecas, archivos, esquemas de bases de datos, archivos de configuración, etc; mientras que el destino de despliegue está generalmente representado por un nodo que es o bien de los dispositivos de hardware o bien algún entorno de ejecución de software. Los nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria.(41)

El sistema estará desplegado en el Servidor de Aplicaciones, el cual se conectará mediante el protocolo TCP/IP al Servidor de Base de Datos, para realizar las consultas a la base de datos del sistema y poder dar respuesta a las peticiones de la PC cliente. Dicha PC se conecta al Servidor de Aplicaciones mediante el protocolo HTTP/HTTPS, y a su vez estará conectada a una impresora por el puerto USB/LPT, en dependencia de la marca y modelo, permitiendo imprimir los momentos de seguimiento por sujeto y el cronograma específico asociado a él.

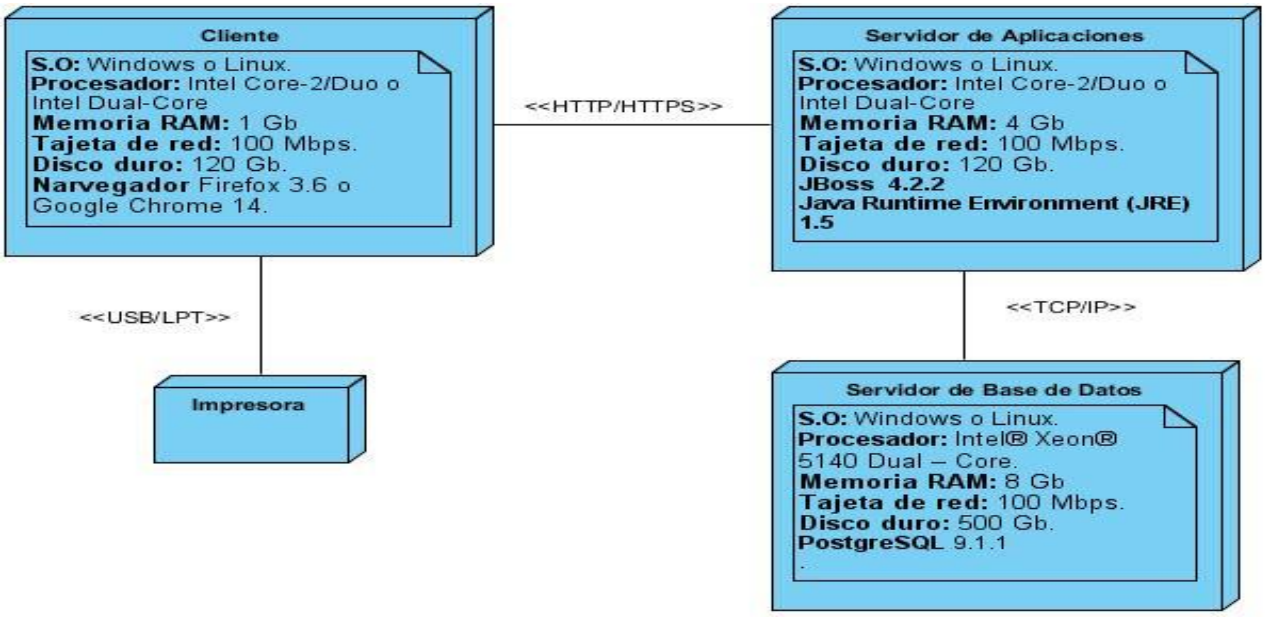

Figura 8. Diagrama de despliegue del Módulo Conducción

4.2.2. Diagrama de componentes

Los diagramas de componentes muestran los componentes del software y los artefactos por los que está compuesto, como las librerías y las tablas de una base de datos, mostrando además las dependencias lógicas entre ellos. Siguiendo la arquitectura, la estructuración en subsistemas de implementación es la siguiente:

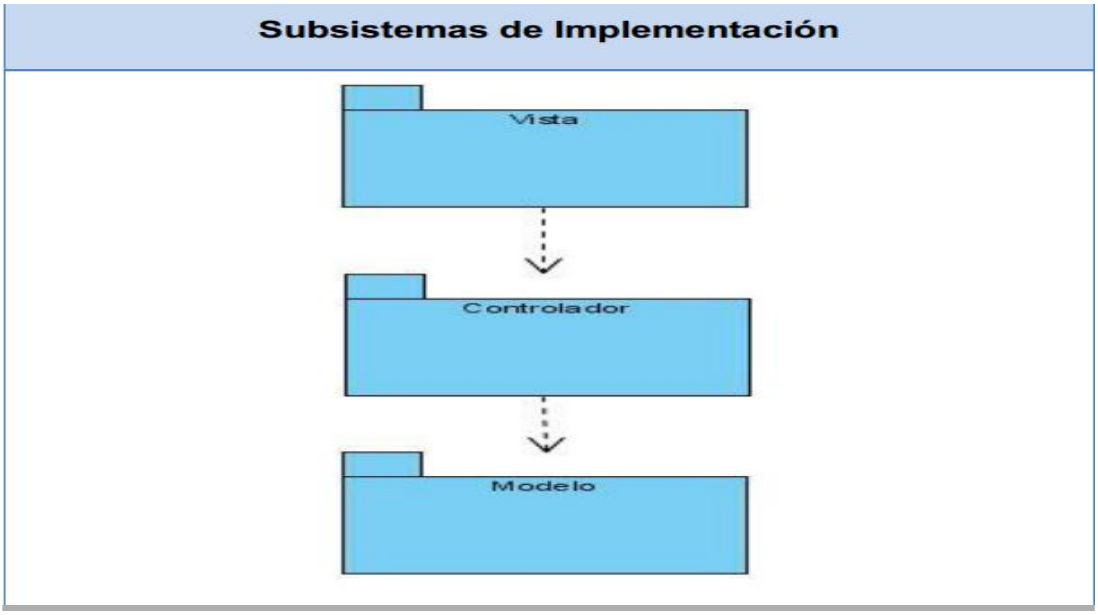

Figura 9. Subsistema de Implementación

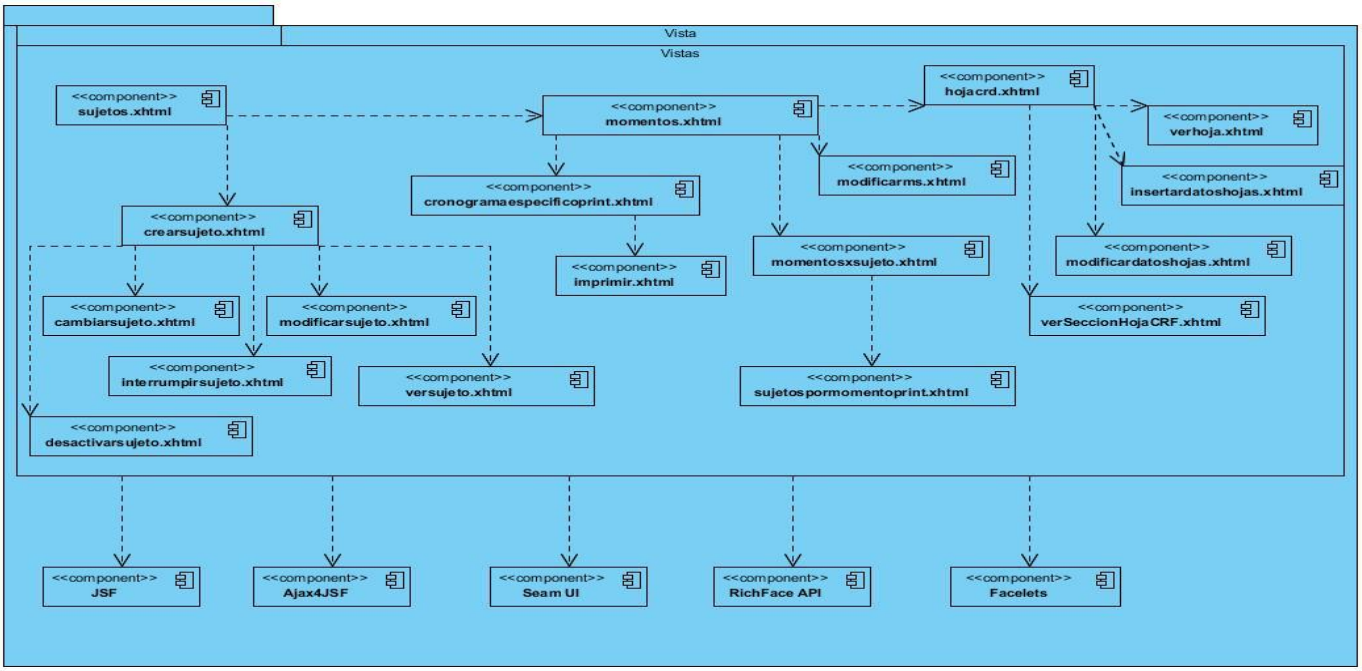

Figura 10. Diagrama de componentes del Módulo Conducción: Vista

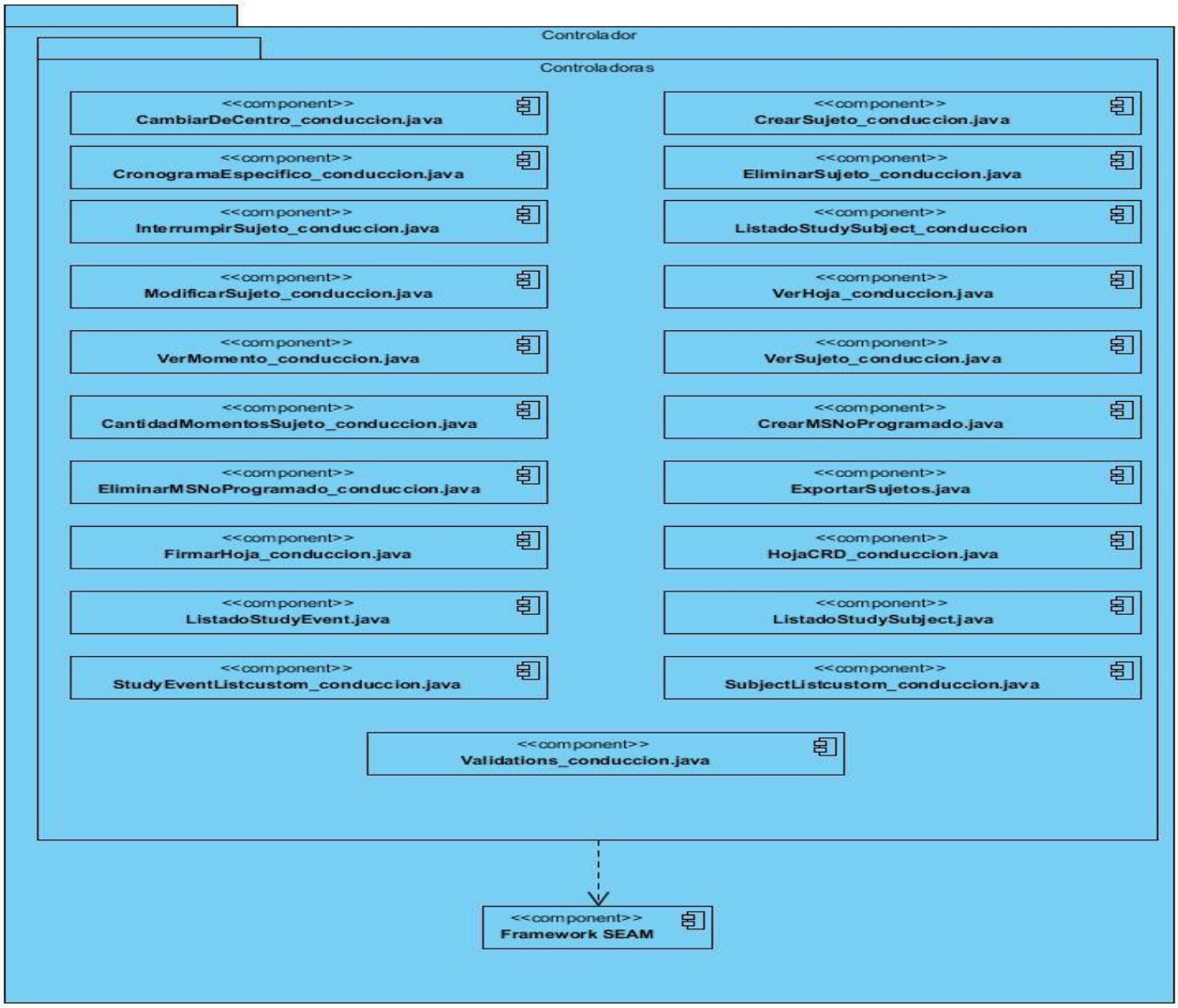

Figura 11. Diagrama de componentes del Módulo Conducción: Controlador

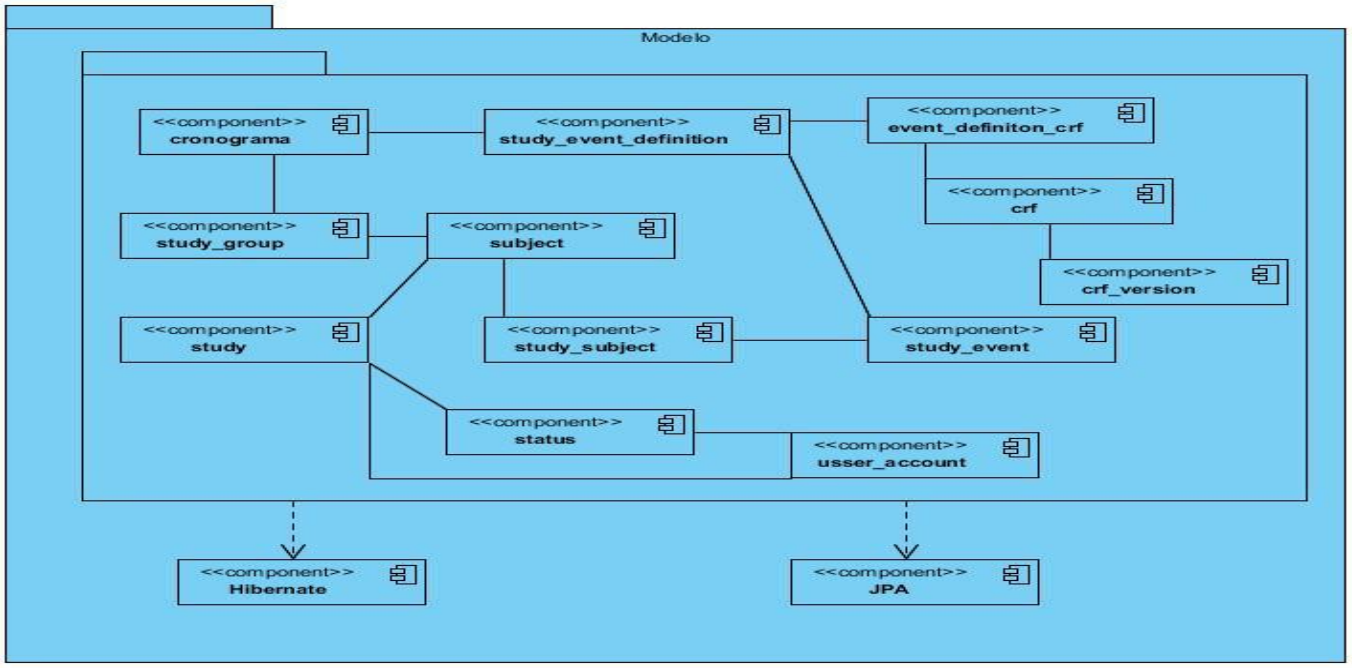

Figura 12. Diagrama de componentes del Módulo Conducción: Modelo

4.2.3. Tratamiento de errores

Un error es algo equivocado o desacertado. Puede ser una acción, un concepto o una cosa que no se realizó de manera correcta. En el desarrollo de las funcionalidades se tiene en cuenta el tratamiento de todos los posibles errores que puedan aparecer, para alcanzar así la integridad y confiabilidad de los datos; esto se garantiza mediante los mensajes de confirmación, los cuales son de fácil comprensión para el usuario y se presentan en caso de que se desee eliminar o modificar datos. Se emplean componentes propios de la librería Richfaces, capaces de validar la entrada por el usuario de valores numéricos, así como las validaciones en el código para la consistencia de los datos.

Ejemplos de validaciones

```
<td>
#{messages.identificador}
<rich:spacer height="0px" style="display: block"></rich:spacer>

<h:inputText id="identificador"
value="#{crearSujeto_conduccion.identificador}"
validator="#{validations_conduccion.validarCadenaAlfanumerica}"
required="#{!empty param['form2:crear']}"
requiredMessage="Este campo es obligatorio"/>
 <rich:message for="identificador" showDetail="false" tooltip="true"
               style="color:red;">
               <f:facet name="errorMarker">
               <h:outputText value="*"
               title="#{facesMessages.getCurrentMessagesForControl('identificador').get(0).getDetail()}" />
               </f:facet>
 </rich:message>
   <h:spacer height="8px" style="display: block"></rich:spacer>
```

Figura 13. Ejemplo de validación

```
/**
 * Valida una cadena alfanumerica
 */
@In(create = true)
EntityManager entityManager;
public void validarCadenaAlfanumerica(FacesContext context,
        UIComponent component, Object value) {
        if(!value.toString().matches("^(\\s*[0-9A-Za-záéíóúÁÉÍÓÚñÑüÜ]+\\s*)+$")){
            validatorManagerExeption(SeamResourceBundle.getBundle().getString("validator.pattern"));

    }
        String identificador=value.toString();
        int dif=entityManager.createQuery("select s from Subject s where s.uniqueIdentifier =:identifica
        if (dif>0) {
            validatorManagerExeption(SeamResourceBundle.getBundle().getString("validator.pattern.id"));
        }
}

/**
```

Figura 14. Ejemplo de validación

Se utiliza además el manejo de excepciones como técnica para controlar los errores que ocurren durante la ejecución del sistema. Una excepción no es más que la indicación de un problema que ocurre con poca

frecuencia y su manejo permite al programador la creación de sistemas robustos y tolerantes a fallas; para lo cual hace uso de herramientas, como es el caso de los bloques try (intentar) que encierran el código que puede lanzar una excepción y los bloques catch (atrapar) que lidian con las excepciones que surjan.

```
@suppresswarnings("unchecked")
public void crearSujeto()
  {
     try {
        usuario=(UserAccount) entityManager.createQuery("select u from UserAccount u where u.userName =:username")
        .setParameter("username", credentials.getUsername()).getSingleResult();
        padre=1;
        estado=(Status_Disenno) entityManager.createQuery("select s from Status_Disenno s where s.name =:nombreestado")
        .setParameter("nombreestado",nombreestado).getSingleResult();
        studyGroup=(StudyGroup) entityManager.createQuery("select s from StudyGroup s where s.name =:nombregruposujeto")
        .setParameter("nombregruposujeto",nombregruposujeto).getSingleResult();
       StudyEvent studyEvent;
    for(int i=0;i<listamomentos.size();i++)
    {
       StudyEventDefinition momento=listamomentos.get(i);
       String[] dias=momento.getDia().split("_");
       for(int j=0;j<dias.length;j++)
       {
          Integer dia=Integer.valueOf(dias[j]);
          if (dia==0)
          {   studyEvent=new StudyEvent();
              studyEvent.setUserAccountByOwnerId(usuario);
              entityManager.persist(studyEvent);
              entityManager.flush();
          }
       }
    }
       catch (Exception e) {
          e.printStackTrace();
       }
```

Figura 15. Ejemplo de tratamiento de excepciones

Internacionalización

Es un proceso de suma importancia dentro del sistema, pues le dará la característica de poder ser adaptado a diferentes idiomas, sin necesidad de cambios de ingeniería. Los usuarios finales podrán disfrutar de una interfaz donde los mensajes que se muestran sean en su idioma nativo.

4.2.4. Seguridad

En la gestión de información en la conducción de ensayos clínicos es necesario que se garantice la seguridad, como elemento importante para preservar la integridad, confiabilidad y autenticidad de los

datos. Dentro de las medidas tomadas para alcanzar dicho principio se encuentra que, para acceder al sistema se solicita nombre de usuario y contraseña, verificando que los datos introducidos sean válidos, si es así, el sistema muestra al actor las funcionalidades a las que tiene acceso. Las acciones realizadas en el sistema por el usuario una vez que inicie o cierre sesión, modifique algún campo o realice cualquier otra operación se registrarán en trazas en la base de datos. También se brinda la posibilidad de asignar o denegar permiso a roles y usuarios. Todas estas medidas se gestionan en el Módulo Configuración del SIGEC.

4.2.5. Estrategias de codificación. Estándares y estilos a utilizar

Los estándares de codificación son guías relacionadas con la generación del código, para facilitar la legibilidad, comprensión y mantenimiento del código entre distintos programadores.

Diseño de Etiquetas:

Todas las etiquetas estarán en el mismo idioma, serán palabras similares a las operaciones que los usuarios deseen llevar a cabo.

Todas las etiquetas estáticas terminarán con dos puntos (:).

Las etiquetas mostrarán al lado derecho un ícono rojo, señalando que el campo es de llenado obligatorio.

Las etiquetas de selección deberán llevar por defecto el texto <<Seleccione>>.

El orden de las opciones será: <<Seleccione>> y el resto de las opciones en orden alfabético ascendente.

Mensajería, algunos ejemplos usados en el sistema:

En los mensajes de creación de una nueva entidad, el sistema debe mostrar el siguiente mensaje:

Se adicionó (el ó la) <Entidad> correctamente.‖

En los mensajes de eliminar una entidad, el sistema debe mostrar el siguiente mensaje:

(La o él) <Entidad> ha sido eliminada‖.

En los mensajes de modificar una entidad, el sistema debe mostrar el siguiente mensaje:

Se modificó (el ó la) <Entidad> correctamente.‖

Todos los mensajes terminarán con punto final.

En las ventanas de Advertencia se utilizarán dos botones centrados en la parte inferior y contendrán los textos: “**Si**” y “**No**”. El botón “**Si**” estará a la izquierda y el “**No**” a la derecha, como se muestra en las Figuras 16 y 17.

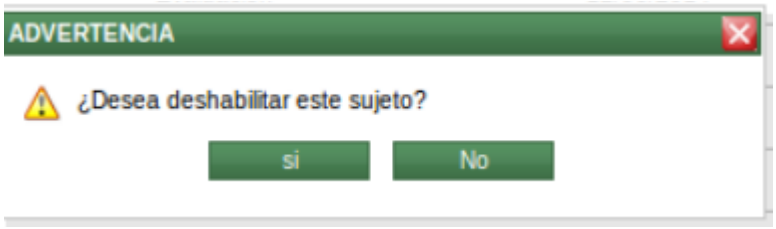

Figura 16. Ejemplo de ventana de advertencia: Deshabilitar sujeto

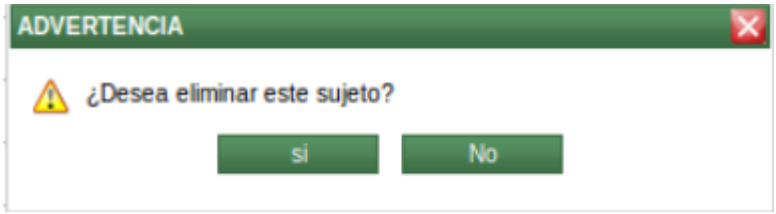

Figura 17. Ejemplo de ventana de advertencia: Eliminar sujeto

Identación o sangría

Objetivo: Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.

El identado debe ser de dos espacios por bloque de código. 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.

Nunca colocar {en la línea de un código cualquiera, esto requiere una línea propia.

Comentarios

Objetivo: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.

Los comentarios deben estar ubicados al inicio de cada clase o función y debe especificar el objetivo de la misma.

Se identificarán por el /*.

```
71    /*Este metodo devuelve la lista de variables pertenecientes
72     * a la seccion que se les pasa por parametro*/
73    public List<Item> ListaVariblesSeccion(Section section)
74    {
75        List<Item> listadovaribles= new ArrayList<Item>();
76        listadovaribles=entityManager.createQuery("select i.item from ItemFormMetadata i wher
77        return listadovaribles;
78    }
79    /*Este metodo devuelve los item formmetadata de una seccion especifica
80     * lo que contiene un item y el ResponseSet de esa variable*/
      public List<ItemFormMetadata> ListaItemFormMetadata(Section section)
      {
```

Figura 18. Ejemplo de comentario

Otras Convenciones de Nomenclatura

En el caso de la opción de “Búsqueda” se utilizará para ello un hipervínculo precedido del ícono  es decir 

.

Los botones aparecerán alineados en la parte inferior a la derecha y el orden será de tal forma que las acciones positivas al flujo sean de derecha a izquierda.

El nombre de las variables se escribe en minúscula, si es una palabra compuesta la primera palabra en minúscula y las demás comenzarán con mayúscula.

Los nombres de los formularios se escriben en minúscula. Ejemplo: datossujeto.

El nombre de los métodos tiene que empezar con un infinitivo y en minúscula, en caso de ser compuesto se pone la primera palabra en minúscula y las demás empezarán con mayúscula. Ejemplo: eliminarSujeto().

Los nombres de las clases controladoras deben comenzar con la primera letra en mayúscula y en caso de ser compuesta la primera palabra comenzará con mayúscula y las demás en minúscula. Ejemplo: Gestionarsujeto.

Pautas de diseño

La confección de las vistas se realizó teniendo en cuenta las pautas de diseño para las aplicaciones web definidas en el Centro de Informática Médica (CESIM), las cuales están descritas en el documento Pautas_Diseño_AplicacionesWEB_CESIM, logrando así la estandarización del diseño de todos los productos desarrollados.

4.2.6. Pantallas de la aplicación

Para obtener una visión simplificada del diseño de interfaz gráfico, a continuación se muestran algunas de las interfaces pertenecientes al módulo Conducción del Sistema de Gestión de Ensayos Clínicos.

Visualizar listado de sujetos:

El usuario registrado puede visualizar el listado de los sujetos incluidos en un estudio, además el sistema le brinda las opciones de Eliminar, Mostrar, Modificar, Desactivar, Restaurar, Interrumpir y Cambiar un sujeto seleccionado por él de centro.

Figura 19. Interfaz: Visualizar listado de sujetos

Modificar datos del sujeto:

El usuario registrado puede visualizar los datos de un sujeto seleccionado de la lista de sujetos, además el sistema permite modificar algunos de los datos pertenecientes al sujeto seleccionado. Los datos que se pueden modificar son: Identificador del sujeto, Sexo, Fecha de nacimiento, Fecha de creación y Centro al que pertenece. En el caso que el usuario modifique la Fecha de creación de un sujeto esta deberá ser menor que las fechas de los momentos de seguimientos pertenecientes al mismo.

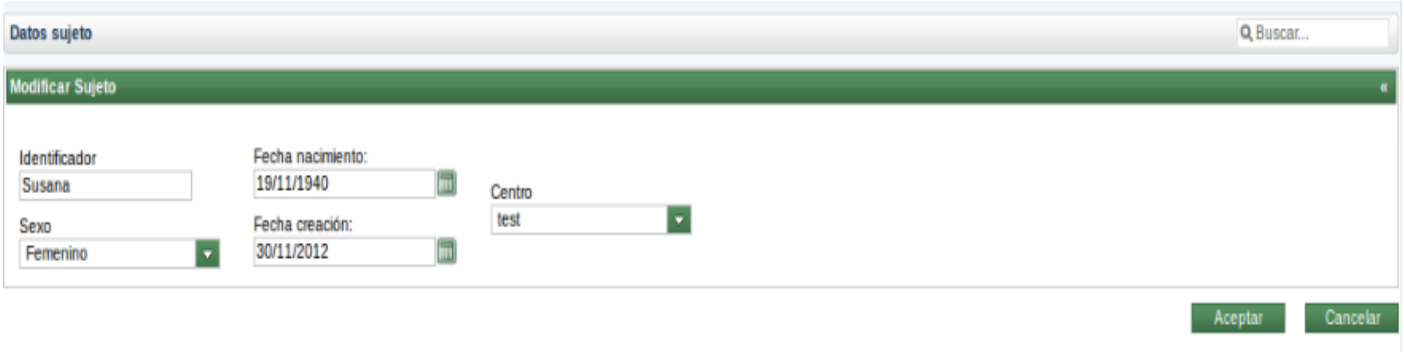

Figura 20. Interfaz: Modificar datos del sujeto

Eliminar sujeto:

El usuario puede eliminar un sujeto del listado de sujetos.

Figura 21. Interfaz: Eliminar sujeto

Mostrar los datos de un sujeto:

Permite mostrar los datos del sujeto seleccionado, además de su cronograma específico asociado.

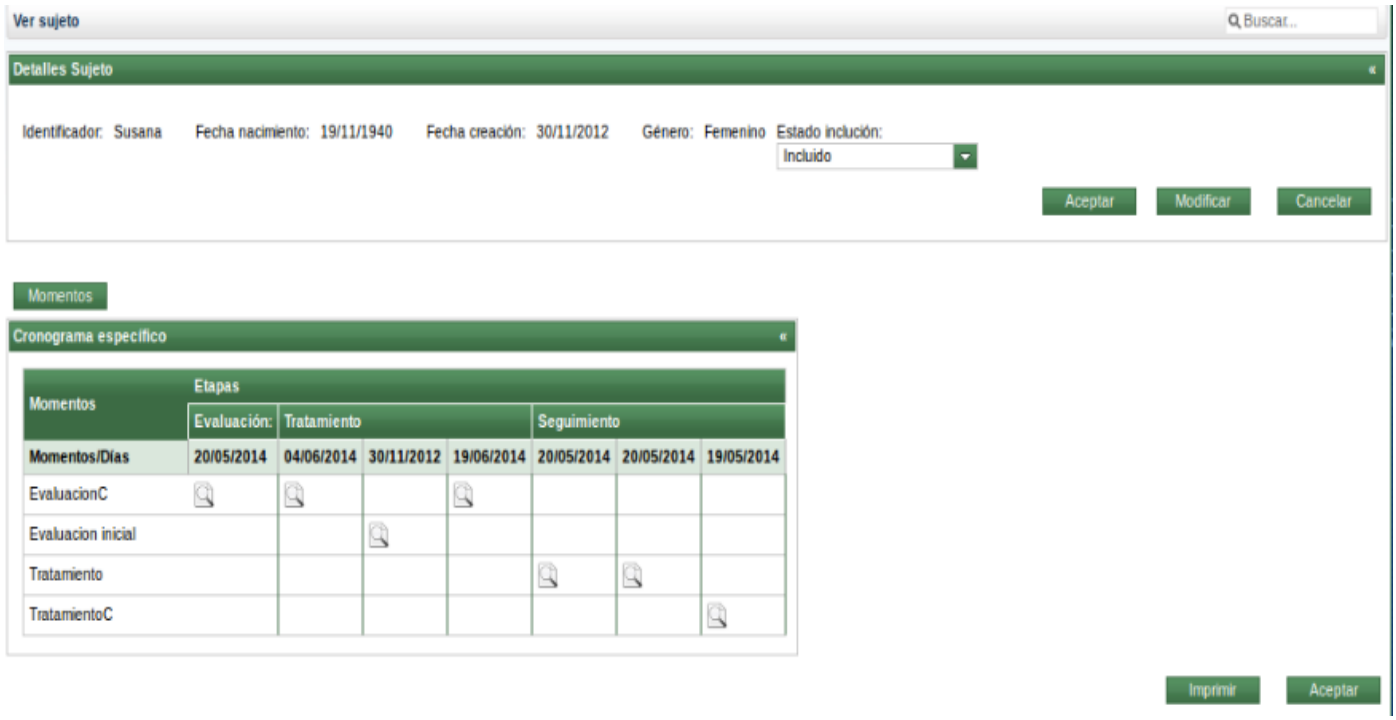

Figura 22. Interfaz: Mostrar los datos de un sujeto

Interrumpir sujeto:

Luego de entrar una fecha de interrupción a un sujeto seleccionado del listado, se le establece el estado de tratamiento “Interrumpido” y el sistema no obligará el llenado de los momentos posteriores al último Momento de Seguimiento llenado.

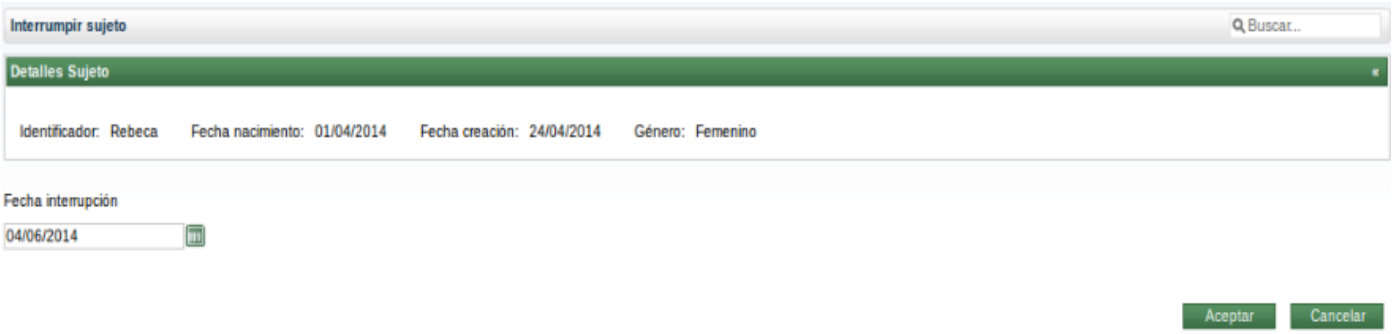

Figura 23. Interfaz: Interrumpir sujeto

Se evidencia cómo construir y distribuir el sistema, a partir de los diagramas de componentes y despliegue, permitiendo la obtención de un producto que cumple con las funcionalidades propuestas. Se muestra el Modelo de Datos y las tablas que lo conforman, las cuales fueron detalladas, así como los estándares de diseño y codificación, tratamiento de errores y seguridad.

CONCLUSIONES

Con la presente investigación se le dio cumplimiento al objetivo general propuesto, por lo que se concluye que:

- Los sistemas informáticos relacionados con la conducción de ensayos clínicos analizados en la investigación, demostraron que no cumplen con las necesidades del Centro de Inmunología Molecular. No obstante permitieron identificar las tendencias actuales en cuanto a diseño y construcción de los sistemas de gestión de ensayos clínicos.
- La utilización de las herramientas, tecnologías y metodologías definidas por el CESIM para el desarrollo del sistema, demostró que son factibles para agilizar el proceso de desarrollo.

- Se elaboró el diseño y la implementación del sistema utilizando la metodología RUP, comprendiendo y analizando su naturaleza iterativa en el desarrollo de proyectos y la evolución de los artefactos generados.
- Se implementaron las funcionalidades para la conducción de estudios en el Sistema de Gestión de Ensayos Clínicos, facilitando la gestión de la información referente a los sujetos, los cronogramas específicos y las hojas CRD, obteniendo un producto de calidad.

RECOMENDACIONES

Con el paso de los años se hacen necesarias nuevas actualizaciones y correcciones de los productos, debido al surgimiento de nuevas necesidades, por tanto se recomienda:

- Se recomienda el uso de este sistema a otros centros a nivel nacional que se encarguen de la conducción de ensayos clínicos.
- Realizar la planificación de los momentos de seguimiento en el cronograma específico teniendo en consideración los días excepcionales.
- Incorporar funcionalidades para la gestión de las notas, viabilizando el monitoreo de los datos.

BIBLIOGRAFÍA

1. **Sanchéz, G. B.** Dirección de Información. *Analisis y Diseno del Sistema para la Gestion de Posgrados en la UCI.* [En línea] 2007. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_0308_07.

2. **Padilla, D. R.** Dirección de Información. *Sistema de informacion estadistico complementario de salud. Modulo: Consulta externa.* [En línea] 2008. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_1000_08.

3. **Jera, Erislán Martínez.**

4. **III, Instituto de salud Carlos.** Legislación Vigente de Medicamentos Huérfanos. *Ensayos Clínicos.* [En línea] http://www.ub.edu/legmh/ereensay.htm.

5. **Universidad Católica de Chile.** Estudios experimentales. [En línea] EPI-CENTRO. http://escuela.med.puc.cl/recursos/recepidem/estExper03.htm.

6. **Imagys.** Medical imaging solutions in clinical trials. [En línea] http://www.imagys.com/index.php?option=com_content&view=article&id=3&Itemid=5.

7. **OBS Medical.** Mobile Patient Reported Outcomes with mPro® Clinical Trial Manager . [En línea] http://www.obsmedical.com/products/mobile-patient-reported-outcomes.

8. **ARCHIMEDES Quantifying Healthcare.** ARCHeS Trial Designer. [En línea] http://archimedesmodel.com/arches-td.

9. **Biocapax Technologies S.L.** BCX Research. [En línea] http://www.biocapax.com/web/bcx_research_es.asp.

10. —. BCX Clinical. [En línea] http://www.biocapax.com/web/bcx_clinical_es.asp.

11. **Caduceus.** OpenClinica. [En línea] http://www.caduceus.es/productos-y-servicios/15-productos/55-openclinica-gestion-estudios-clinicos.html.

12. **Lucía Rodríguez García, Martha Denia Hernández Ramírez, Richard Díaz Pompa, Erislán Martínez Jera, Leandro Hernández Cuello, Landy González.** "ALASCLÍNICAS": SISTEMA DE GESTIÓN DE ENSAYOS CLÍNICOS. *Universidad de las Ciencias Informáticas.* [En línea] http://www.rcim.sld.cu/revista_24/articulo_pdf/alasclinicas.pdf.

13. **Pressman, Roger S.***Ingeniería de software, un enfoque práctico .* 2005.

14. IT ahora . *La revista del comprador tecnológico.* [En línea] http://www.itahora.com/internet/que-es-tecnologia-definicion-de-tecnologia.

15. Tratando de entenderlo. [En línea] http://tratandodeentenderlo.blogspot.com/2009/07/instalacion-de-jboss-tools.html.

16. Adictos al trabajo. *Introducción a Ajax4Jsf.* [En línea] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.

GLOSARIO DE TÉRMINOS

17. Adictos al trabajo. *Introducción a RichFaces.* [En línea] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro.

18. Osmosis Latina. *AJAX.* [En línea] http://tecencliente.osmosislatina.com/curso/ajax.htm.

19. EcuRed. [En línea] http://www.ecured.cu/index.php/Hibernate.

20. Kioskea.net. *CSS: Hojas de estilo.* [En línea] http://es.kioskea.net/contents/156-css-hojas-de-estilo.

21. **Luis Rondon Grados .** JAVA J2EE: JPA - Java Persistence API. [En línea] http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html.

22. **Jatun S.R.L.** Java Enterprise Edition. [En línea] http://www.jatun.com/web/company/training/javaee5.

23. [En línea] www.magma.com.ni/~jorge/upoli_uml/refs/Resumen_de_UML.doc.

24. El mundo Informático y tú en que mundo vives? *Lenguajes de Programación.* [En línea] http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/.

25. JAVA. [En línea] http://www.infor.uva.es/~jmrr/tgp/java/JAVA.html.

26. MASTERMAGAZINE. *Definición de Herramienta.* [En línea] http://www.mastermagazine.info/termino/5234.php.

27. Free Download Mananger. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

28. PgAdmin III- Guía Ubuntu. [En línea] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

29. EcuRed. *Sistema Gestor de Base de Datos.* [En línea] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

30. PostgreSQL. [En línea] http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

31. Scribd. [En línea] http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.

32. Elementos de UML. [En línea] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

33. jojooa - tecnología, marketing y crm. [En línea] marzo de 2014. https://sites.google.com/site/jojooa/analisis-de-sistemas/definicion-de-actor-que-es-un-actor.

34. Ecured. *Arquitectura de software.* [En línea] http://www.ecured.cu/index.php/Arquitectura_de_software.

35. Ecured. *Modelo Vista Controlador.* [En línea] http://www.ecured.cu/index.php/Patr%C3%B3n_Modelo_Vista_Controlador.

36. [En línea] http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/Complemento%20Material%20Didactico/Maest-Ing-Soft-Sergio/Cuerpoconocimiento/Dise%C3%B1o%20de%20software.htm.

37. MeRinde. [En línea] http://merinde.net/index.php?option=com_content&task=view&id=92&Itemid=296.

38. Microsoft Developer Network. *¿Qué es un Patrón de diseño?* [En línea] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

39. Ecured. [En línea] http://www.ecured.cu/index.php/Patr%C3%B3n_de_dise%C3%B1o_de_software.

40. Ecured. [En línea] http://www.ecured.cu/index.php/Patrones_de_Asignaci%C3%B3n_de_Responsabilidades.

41. UML Diagrama de despliegue. [En línea] http://umldiagramadespliegue.blogspot.com/.

42. **Biocapax Technologies S.L.** BCX Research. [En línea] http://www.biocapax.com/web/BCX_Research_Metodologia_es.asp.

43. Mundo geek. *Hibernate.* [En línea] http://mundogeek.net/archivos/2007/01/27/hibernate/.

44. **Estela López Suárez, Claudia Flores Villa.** Desarrollo del Módulo Archivo del Sistema de Información Hospitalaria alas HIS. [En línea] 2011. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_04013_11.