Universidad de las Ciencias Informáticas

Facultad 2

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

Funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica.

Autores: Yasmani Ballester Leyva

Yariel Pérez Salazar

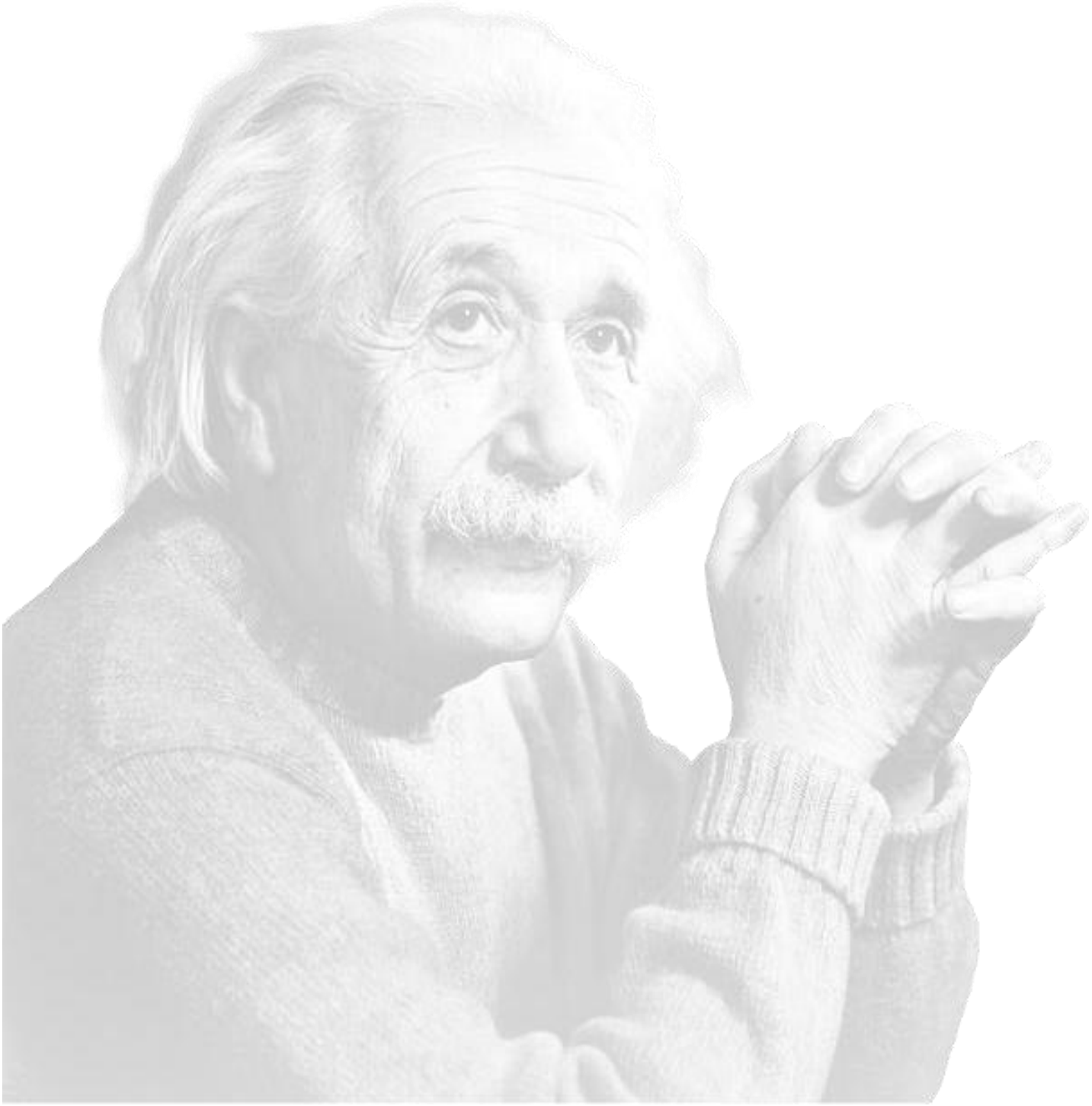
Tutores: Ing. Lixandra Cordova Eireos

Ing. Yuniel García Sánchez

Co-tutor: Ing. Adrián Amado Hidalgo Chang

La Habana, Junio de 2014

"Año 56 de la Revolución"



"LA EDUCACIÓN ES LO QUE QUEDA DESPUÉS DE QUE UNO HA OLVIDADO TODO LO QUE APRENDIÓ EN LA ESCUELA."

ALBERT EINSTEIN

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los_25_días del mes de_Junio_del año_2014_.

Firma del autor
Yasmani Ballester Leyva

Firma del autor
Yariel Pérez Salazar

Firma del tutor
Lixandra Cordova Eireos

Firma del tutor
Yuniel García Sánchez

Firma del cotutor
Adrián Amado Hidalgo Chang

DATOS DE CONTACTO

Ing. Lixandra Cordova Eireos

Graduada en el año 2011 de Ingeniera en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Especialista General, Supervisora Evaluadora y Tutora (SET) en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM). Actualmente se desempeña como analista del Sistema de Información Hospitalaria del Centro.

Correo electrónico: lcordova@uci.cu

Ing. Yuniel García Sánchez

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2011. Actualmente labora en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM). Se desempeña como desarrollador del Sistema de Información Hospitalaria del Centro.

Correo electrónico: ygarcias@uci.cu

Ing. Adrián Amado Hidalgo Chang

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2013. Actualmente se encuentra en etapa de adiestramiento y labora en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM). Se desempeña como desarrollador del Sistema de Información Hospitalaria del Centro.

Correo electrónico: aahidalgo@uci.cu

AGRADECIMIENTOS

Queremos agradecer a nuestros tutores por brindarnos su incondicional ayuda a lo largo del desarrollo de este trabajo diploma, en especial a Lixandra por su dedicación, paciencia y entrega que nos guió por el camino correcto. A nuestros compañeros de aula que en conjunto supimos apoyarnos y esforzarnos para alcanzar la meta trazada inicialmente y en general a todos los que a lo largo de la carrera contribuyeron a nuestra formación.

De Yasmani:

Quiero agradecer de manera muy especial a mis padres que son el orgullo de mi existencia, la luz que me ha guiado en todo momento en el transcurso de mi vida, a mi papi por ser el gran ejemplo de persona que siempre ha logrado inspirarme a seguir adelante y a luchar por lo que quiero, a mi mami por ser la alegría de mis días y por todo el amor y apoyo incondicional que siempre ha sabido transmitirme.

A mis hermanos, abuelos, tíos, sobrinos, primos y familia en general que siempre de una forma u otra se han preocupado por mí.

A mis amigos por estar siempre presentes cuando he necesitado su ayuda y por compartir alegrías y tristezas a lo largo de estos años, en especial a Marila y Jorgito que los considero mis hermanos, a Belkita que en un momento determinado contar con su apoyo y comprensión significó mucho para mí, a mi gente del apartamento y a mi compañero de tesis.

De Yariel:

Agradecer de manera muy especial a mis padres que son la razón de mi ser, por haber hecho de mí una mejor persona, por haberme apoyado en todo momento y darme todo su amor incondicional. A ti mami agradecerte lo mucho que te preocupas por mí, tu cariño y amor desinteresado que me trasmites todos los días, nunca nada podrá borrar eso de mi corazón. A ti papi agradecerte por ser mi ejemplo de hombre a seguir, y enseñarme a salir adelante y cumplir con todas mis metas. A mis hermanos por apoyarme en todos estos años de mi vida y que tanto se han preocupado por mí, sin ellos no estuviera ahora mismo donde estoy.

A todos mis amigos de la UCI por haber compartido juntos momentos felices y tristes todos estos años de universidad, en especial a la gente de mi grupo, de mi apartamento y a mi compañero de tesis.

DEDICATORIA

De Yasmani:

A mis padres por hacer que mi día a día sea más fácil, por la educación y formación que me han dado en todos estos años, por hacerme ver que puedo lograr todo lo que me proponga en la vida y por haberme inculcado los valores que en la actualidad me definen como persona. A mis familiares y amigos que me han brindado su apoyo en todo momento y han depositado toda su confianza en mí.

De Yariel:

A toda mi familia y de forma especial a mis padres y a mis hermanos, por entregarme de forma incondicional su amor y dedicación, su apoyo en todo momento y haberme encaminado de forma correcta en la vida.

RESUMEN

Actualmente en el Sistema de Información Hospitalaria del Centro de Informática Médica (CESIM) la gestión de los procesos asociados a los estudios especializados que se llevan a cabo en las instituciones hospitalarias se realiza incorrectamente, por lo que el objetivo del presente trabajo está enfocado al desarrollo de funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del CESIM, para permitir su correcta utilización por parte de los especialistas de las instituciones hospitalarias.

Este Sistema de Información Hospitalaria tiene como objetivo fundamental registrar, organizar, actualizar, conservar y mostrar la información de forma dinámica y segura. Su desarrollo está basado en tecnologías libres o de código abierto, multiplataforma y sobre una arquitectura en capas, empleándose el patrón Modelo Vista Controlador. Se utilizó Java como lenguaje de programación, PostgreSQL 8.4 como servidor de aplicaciones, el framework Hibernate para el acceso a datos y el framework Seam para la unión entre la capa de presentación y acceso a datos.

El desarrollo de funcionalidades relacionadas con la gestión de los procesos de estudios especializados del Sistema de Información Hospitalaria del CESIM permite organizar y administrar los procesos relacionados con los estudios especializados que se llevan a cabo en las instituciones hospitalarias, así como posibilitar la obtención de datos estadísticos reales, que faciliten la toma de decisiones administrativas y brindar la posibilidad a los médicos especialistas de indicar desde la propia consulta, los estudios especializados que consideren necesarios para el paciente.

PALABRAS CLAVES

Estudios especializados, framework, Sistema de Información Hospitalaria.

TABLA DE CONTENIDO

Introducción **1**

CAPÍTULO 1. Fundamentación teórica **7**

1.1. Conceptos básicos relacionados con el dominio del problema. *7*

1.2. Sistemas de Información Hospitalaria vinculados al campo de acción. *8*

1.3. Tendencias y tecnologías actuales a utilizar. *11*

1.4. Tecnologías utilizadas en el proceso de desarrollo. *13*

1.5. Interfaces de comunicación. *17*

1.6. Herramientas. *19*

CAPÍTULO 2. Características del sistema **21**

2.1. Flujo actual de los procesos en el campo de acción. *21*

2.2. Información que se maneja. *22*

2.3. Modelo de Negocio. *23*

2.4. Especificación de requerimientos de software. *25*

2.5. Modelo de casos de uso del sistema. *30*

CAPÍTULO 3. Análisis y diseño del sistema. **39**

3.1. Arquitectura. *39*

3.2. Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados. Estrategias de integración. *40*

3.3. Modelo de diseño. *40*

3.4. Modelo de datos. *60*

CAPÍTULO 4. Implementación del sistema. **65**

4.1. Modelo de Despliegue. *65*

4.2. Diagrama de Componentes. 66

4.3. Tratamiento de errores..... 67

4.4. Seguridad. 68

4.5. Estrategias de codificación. Estándares y estilos a utilizar. 68

Conclusiones generales 71

Recomendaciones 72

Referencias bibliográficas..... 73

Bibliografía 76

Anexos..... 80

Glosario de Términos 87

ÍNDICE DE TABLAS

Tabla 2.1 Trabajadores del Negocio. 25

Tabla 2.2 Asignar cita para estudio especializado en el módulo de Citas. 33

Tabla 2.3 Buscar cita para estudio especializado desde el módulo de Citas. 34

Tabla 2.4 Asignar cita para estudio especializado en consulta externa. 34

Tabla 2.5 Crear estudio especializado. 35

Tabla 2.6 Modificar estudio especializado. 35

Tabla 2.7 Eliminar estudio especializado. 36

Tabla 2.8 Crear horario para estudio especializado. 36

Tabla 2.9 Modificar horario para estudio especializado. 37

Tabla 2.10 Eliminar horario para estudio especializado. 37

Tabla 2.11 Ver datos de horario para estudio especializado. 38

Tabla 2.12 Ver detalles de horario para estudio especializado. 38

Tabla 3.1 Descripción de la clase AsignarCitaEstudioEspecializadoControlador. 50

Tabla 3.2 Descripción de la clase CrearEstudioEspecializadoControlador. 53

Tabla 3.3 Descripción de la clase CrearHorarioCitaEstudioEspecializadoControlador. 57

Tabla 3.4 Descripción de la clase CrearSolicitudEstudioEspecializadoControlador. 60

Tabla 3.5 Descripción de la tabla Usuario. 62

Tabla 3.6 Descripción de la tabla Médico. 62

Tabla 3.7 Descripción de la tabla Planificación_estudio_especializado. 62

Tabla 3.8 Descripción de la tabla Especialidad. 63

Tabla 3.9 Descripción de la tabla Cita. 63

Tabla 3.10 Descripción de la tabla Horario_estudio_especializado. 63

Tabla 3.11 Descripción de la tabla Hoja Frontal. 64

Tabla 3.12 Descripción de la tabla Estudio_especializado. 64

ÍNDICE DE FIGURAS

Figura 2.1 Asignar cita para estudio especializado. 24

Figura 2.2 Gestionar horario para estudio especializado. 24

Figura 2.3 Diagrama de Actores. 31

Figura 2.4 Asignar cita para estudio especializado en el módulo de Citas. 31

Figura 2.5 Asignar cita para estudio especializado desde Consulta Externa. 32

Figura 2.6 Buscar estudio especializado. 32

Figura 2.7 Gestionar horario de cita para estudio especializado. 33

Figura 3.1 Diagrama de paquetes. 43

Figura 3.2 Asignar cita para estudio especializado. 44

Figura 3.3 Buscar cita para estudio especializado. 45

Figura 3.4 Gestionar estudio especializado. 45

Figura 3.5 Gestionar horario para estudio especializado. 46

Figura 3.6 Modelo de datos. 61

Figura 4.1 Modelo de Despliegue. 65

Figura 4.2 Diagrama de componentes. 66

INTRODUCCIÓN

Desde el comienzo de la vida humana, el hombre fue evolucionando y al mismo tiempo creando herramientas de trabajo con el fin de hacer más fácil y eficiente su existencia en el planeta. Con el paso del tiempo fue logrando importantes descubrimientos entre los que cobran gran relevancia los equipos de cómputo para procesar algoritmos complejos. En la actualidad, el desarrollo y mejoramiento de estos equipos ha llevado a cabo el surgimiento de una nueva era en el desarrollo de la informática y las comunicaciones.

Con el desarrollo y avance de las Tecnologías de la Información y las Comunicaciones (TIC), los países se han trazado como meta desde hace unos años, la implementación de sistemas computarizados para informatizar la sociedad. Han experimentado un incremento moderado en el uso de las tecnologías de la información, que ha proporcionado diariamente un aumento de la calidad y el nivel de vida de sus habitantes, así como una eficacia y eficiencia de procesos que en estos países se llevan a cabo.

En la informatización de los procesos hospitalarios se han alcanzado excelentes resultados debido a la importancia que tiene el sistema de salud en la sociedad. Estos procesos pueden conducir una demora en la atención al paciente debido a que la manipulación de la información en las instituciones hospitalarias es engorrosa y compleja. Los hospitales como actores principales del sistema sanitario generan un importante volumen de información, pero en la mayoría de los casos esta se encuentra dispersa o no está disponible en tiempo y forma necesaria, por lo tanto se evidencian una de las prioridades fundamentales para informatizar el Sistema de Salud.

En países subdesarrollados atendiendo a las políticas que lo rigen y atendiendo a sus factores internos y necesidades, desde hace algún tiempo se comenzaron a desarrollar sistemas informáticos computarizados para instituciones hospitalarias. Estos sistemas son los denominados Sistemas de Información Hospitalaria (más conocidos por sus siglas en inglés como HIS), los cuales tienen como objetivo principal sistematizar y optimizar las actividades para incrementar la calidad de la atención al paciente.

Los HIS son un instrumento que permite recoger y tratar la información de modo que sea útil para la toma de decisiones. Además, permite la recolección, el almacenamiento, el procesamiento, la

recuperación y comunicación de información de atención al paciente y administrativa para todas las actividades relacionadas con el hospital. Con los HIS, los pacientes son beneficiados al contar con una atención más organizada y eficiente en la consulta determinada, y los profesionales de la salud encuentran en estos un recurso idóneo, amigable y flexible que responda a las necesidades de información de la institución hospitalaria.

Cuba no está ajena a la necesidad de contar con sistemas informáticos que manejen este tipo de información hospitalaria y sobre todo por ser reconocida como una potencia mundial en la esfera de la salud. Por la importancia de estos sistemas, a la Universidad de las Ciencias Informáticas y en específico al Centro de Informática Médica (CESIM), se le encomendó la tarea de desarrollar un HIS para facilitar todo el proceso de gestión de la información dentro de los hospitales.

El Sistema de Información Hospitalaria del CESIM tiene como objetivo gestionar toda la información que se genera en las diferentes áreas de las instituciones hospitalarias. Está compuesto por diecisiete módulos integrados entre sí, entre los que se encuentra Citas, el cual está concebido para la asignación de citas a los pacientes teniendo en cuenta los tipos de consulta, asociadas a servicios y especialidades que se llevan a cabo en la institución. Los tipos de citas que maneja el sistema son: cita de primera, cita de control o sucesiva, cita para interconsulta, cita por referencia, cita para charla, cita para pre-empleo y cita para terminación.

Este módulo se encuentra estrechamente relacionado con el módulo Consulta Externa debido a que, en éste, se gestiona la planificación de horarios con la disponibilidad que tiene el personal médico (especialistas, residentes, internos y profesionales de salud) para realizar los diferentes tipos de consultas. Esta disponibilidad es consultada cuando se realiza la asignación de una cita a determinado paciente.

En el módulo Consulta Externa, cuando el médico desarrolla el proceso de atención al paciente y accede a registrar la información resultante de la interacción con el mismo en una hoja de consulta, el sistema brinda la posibilidad de indicar algunas solicitudes de estudios especializados en dependencia de la especialidad de atención de la consulta y además, permite indicar solicitudes de estudios especializados asociados a la especialidad gastroenterología desde todas las hojas de consulta

En caso de que el médico determine que el paciente requiere alguno de los estudios especializados concernientes a otra especialidad, se hace necesario otorgarle una cita para la especialidad desde la cual se puede indicar la solicitud del estudio; lo cual genera una hoja de consulta innecesaria en el sistema cuando realmente en la mayoría de los casos no se realiza una consulta sino meramente el estudio indicado, esta hoja de consulta generada trae como consecuencia la obtención de estadísticas irreales para el médico y la institución, al registrarse ésta como una atención al paciente cuando solo se indicó el estudio especializado requerido.

Además, influye negativamente en la organización de la información que se registra en el sistema, teniendo en cuenta la relación que existe entre los tipos de estudios y cada una de las especialidades médicas, o sea, en ocasiones se indica que el estudio a realizar constituye la especialidad de la consulta, cuando realmente un estudio especializado no es una especialidad sino que pertenece a una especialidad determinada y por lo tanto cuando ocurre esta situación, la información se registra incorrectamente en el sistema.

Basado en lo antes expuesto se tiene como **problema a resolver**: ¿Cómo reestructurar la gestión de los procesos relacionados con los estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica, para su correcta utilización por parte de los especialistas de las instituciones hospitalarias?

Para dar cumplimiento al problema planteado se define como **objeto de estudio**: los procesos relacionados con los estudios especializados en las instituciones hospitalarias. El **campo de acción** está centrado en la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica.

Para resolver el problema identificado se propone el siguiente **objetivo general**: Desarrollar las funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica, que permita su correcta utilización por parte de los especialistas de las instituciones hospitalarias.

Tareas de la Investigación:

1. Analizar los procesos relacionados con los estudios especializados en las instituciones hospitalarias.

2. Analizar los Sistemas de Información Hospitalaria que posean funcionalidades relacionadas con la gestión de estudios especializados.
3. Asimilar la arquitectura definida por el Departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
4. Desarrollar los artefactos asociados a los flujos de trabajo de: Modelado de Negocio, Gestión de Requisitos, Diseño e Implementación.
5. Implementar las funcionalidades relacionadas con la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica.

Para lograr desarrollar la investigación correspondiente con éxito y que esta influya luego positivamente en los resultados esperados, es importante la utilización de métodos científicos. Según plantea Rolando Alfredo, los métodos científicos representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional. (36)

Pueden ser encontrados distintos métodos, los cuales pueden ser aplicados a las investigaciones para obtener de manera más rápida y confiable los resultados esperados. Los métodos teóricos posibilitan ir más allá de las particularidades fenomenológicas y triviales de la situación en análisis, además de la situación existente del fenómeno en una etapa determinada. Los métodos empíricos posibilitan extraer de los fenómenos examinados las informaciones que se requieren sobre ellos a través de observaciones.

Como métodos científicos utilizados durante la investigación, todos bajo la concepción dialéctico materialista se han definido los que se listan a continuación:

Métodos teóricos:

- **Histórico – Lógico:** permite la realización del estudio de los fenómenos y acontecimientos en un orden cronológico, para conocer la evolución y desarrollo de la esencia de la investigación en curso. Además, se usó con el objetivo de estudiar todo lo referente a las características fundamentales de las herramientas a utilizar en la investigación.
- **Analítico-Sintético:** permite resumir, enunciar y describir los requerimientos funcionales de la solución propuesta, debido a que este método permite buscar la esencia del problema

planteado en la investigación, los rasgos que lo caracterizan y lo distinguen. Además, ofrece la posibilidad de extraer los elementos más importantes que se relacionan con el objeto de estudio. También permite organizar y sintetizar toda la información obtenida a través del estudio del estado del arte para la solución propuesta, ofreciendo la posibilidad de comprender toda la información investigada.

- **Modelación:** método teórico que ofrece parte de la información necesaria acerca del objeto que se estudia, se trata de explicar la realidad con la creación de diagramas; los cuales pueden ser presentados en sustitución de la realidad. Mediante su utilización se elaborarán diferentes tipos de diagramas que brindarán información clara sobre el tema de estudio, mediante el descubrimiento de nuevas relaciones y cualidades del objeto de estudio.

Métodos-Empíricos

- **La entrevista:** utilizada como conversación planificada para obtener información, constituye un medio para el conocimiento cualitativo de los fenómenos relacionado con el problema planteado en la investigación. En este caso se utilizó la entrevista no estructurada. (Ver anexo 1 y anexo 2).

Es importante destacar que el desarrollo de las funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del CESIM, proporcionará un grupo de beneficios entre los cuales se pueden mencionar los siguientes:

1. Organizar y administrar los procesos relacionados con los estudios especializados que se llevan a cabo en el Sistema de Información Hospitalarias del CESIM.
2. Brindar la posibilidad a los médicos especialistas de indicar desde la propia consulta, los estudios especializados que consideren necesarios para el paciente.
3. Posibilitar la obtención de datos estadísticos reales, que faciliten la toma de decisiones administrativas.

Con el propósito de organizar y darle una estructura al trabajo se ha decidido dividirlo en cuatro capítulos que se desglosan a continuación:

Capítulo 1: Fundamentación teórica: Brinda un estado del arte de los Sistemas de Información Hospitalaria existentes que poseen funcionalidades relacionadas con la gestión de los procesos de estudios especializados. Se estudian las tecnologías, metodologías y herramientas a utilizar en el proceso de desarrollo.

Capítulo 2: Características del sistema: Contiene información que será manipulada por el sistema como la descripción de los procesos del negocio, el modelo de negocio, la especificación de los requisitos de software y la definición de los casos de uso.

Capítulo 3: Análisis y diseño del sistema: Se centra en la definición del modelo de análisis, modelo de clases de análisis y en el diseño. Se describe la arquitectura a utilizar y se realiza un análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados.

Capítulo 4: Implementación del sistema: Se implementan las clases y subsistemas en términos de componentes. Se elabora el modelo de datos y se realiza una descripción de las tablas que lo integran. Además, se exponen aspectos referentes a la seguridad del sistema, las estrategias de codificación y la forma en que se realizará el tratamiento de errores.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los sistemas existentes vinculados al campo de acción, semejanzas y sus rasgos fundamentales, para de esta forma entender la necesidad de una nueva solución acorde a los requerimientos que en la actualidad tienen los hospitales. Por último, se describen las características de las tecnologías, herramientas y metodologías que serán empleadas en el desarrollo de la solución propuesta.

1.1. Conceptos básicos relacionados con el dominio del problema.

Historia Clínica (HC)

Documento que se crea para almacenar el comportamiento evolutivo de un paciente durante su estancia en el hospital. No se limita a ser una narración o exposición de hechos simplemente, sino que incluye juicios, documentos y procedimientos; es un documento que se va haciendo en el tiempo, documentando fundamentalmente la relación médico-paciente. (1)

Cita médica

Una cita médica no es más que un encuentro previamente acordado entre el paciente y el personal médico en ejercicio de su profesión encargado de atender los problemas relacionados con la salud de esta persona estableciendo una fecha, hora y lugar determinado. (2)

Entre los tipos de citas médicas se encuentran:

Cita de primera

Responde a una consulta de primera y no es más que una cita que se le planifica a un paciente que viene por primera vez a atenderse por determinada patología. (2)

Cita de control o sucesiva

Responde a una consulta de control y no es más que una cita que se le planifica a un paciente que viene a atenderse por una patología anteriormente diagnosticada, la cual requiere ser controlada con determinada frecuencia. (2)

Cita para interconsulta

Responde a las interconsultas solicitadas por determinados especialistas, no es más que la cita que se le emite a un paciente cuando necesita ser valorado por un médico de otro servicio. (2)

Cita por referencia

Esta responde a una consulta de primera, pero es una cita para el caso de los pacientes que vengan remitidos de un servicio de la entidad o de una entidad externa. (2)

Cita para charla

En este caso no responden específicamente a un tipo de consulta, pero existen centros hospitalarios que antes de asistir a determinada consulta el paciente debe asistir a un conjunto de charlas con determinado objetivo, por ejemplo en el caso de los pacientes diabéticos le enseñan a cómo mejorar sus hábitos alimenticios, así como a los hipertensos. Está también el caso de las mujeres que salen embarazada por primera vez. Estas citas pueden ser para médicos, enfermeras, entre otros. (2)

Cita para pre-empleo

Responde a una consulta de pre-empleo que se les planifica a las personas que deben incorporarse como trabajadores a una determinada empresa o centro laboral. (2)

Cita para consulta de terminación

Responde a una consulta de terminación que se les planifica a los trabajadores cuando culminan sus labores en una determinada empresa o centro laboral. (2)

Cita para estudios especializados

Responde a una consulta de primera y puede convertirse en una consulta de control o sucesiva, es aquella cita donde el paciente necesita un estudio especializado como: Electrocardiograma, Audiometría, Epirometría, etc. (2)

1.2. Sistemas de Información Hospitalaria vinculados al campo de acción.

1.2.1. Sistema de Información Hospitalaria (x-HIS)

Es uno de los HIS que contribuye hoy en día a hacer más eficiente y eficaz el trabajo. Desarrollado por ISOFT una compañía del grupo IBA Health. El x-HIS es un sistema multiplataforma, presenta

soporte para la arquitectura cliente-servidor y/o 3 capas, permite la integración con otros sistemas como RIS, PACs mediante los estándares HL7, CMBD, Codificación ICD-9-CM/ICD 10, SNOMED, entre otros; y es un sistema multilenguaje. Su alcance está enfocado en las Áreas de Admisión, Urgencias, Consulta Externa, Bloque Quirúrgico, Archivo de Historia Clínica, Gestión de Prescripciones y resultados médicos, Historia Clínica Electrónica, Hoja de Prescripciones, Anatomía Patológica, Radiología, Hemoderivados, Nutrición y Dietética, Fisioterapia. A pesar de ser un sistema fácil, ágil y extensible el sistema es una aplicación de escritorio por lo que se requiere de mayor cantidad de recursos para lograr su distribución y despliegue. (3)

1.2.2. Sistema de Información Hospitalaria (Galenhos)

El sistema Galenhos(R) ha sido diseñado con el propósito de apoyar a los establecimientos de salud en el correcto registro de información, clínica o administrativa, y la generación de información gerencial. Su desarrollo nace de la asistencia técnica prestada por la agencia de Estados Unidos para el desarrollo internacional (USAID), a través del proyecto Partners for Health Reform plus (PHRplus), a un hospital de Perú. El sistema Galenhos cuenta con los módulos Consulta Externa y dentro de este un submódulo para la asignación de citas, Hospitalización, Emergencia, Facturación, Archivo Clínico, Farmacia y Laboratorio. (4)

1.2.3. Sistema de Información Hospitalaria (Sigho)

Sigho es un Sistema de Información para la Gerencia Hospitalaria, que la Secretaría de Salud a través de la Dirección General de Información en Salud (DGIS) ha liberado para su implementación en apoyo a la gerencia de todas las Unidades Médicas del sector salud en México.

Es un software basado en la Norma Oficial Mexicana NOM-168-SSA1-1998 referente al resguardo y uso del expediente clínico electrónico para facilitar las actividades de gerencia dentro del hospital y se apoya de estándares internacionales para el diagnóstico de enfermedades y realización de procedimientos tales como el CIE-10 y CIE9MC. Consta de 14 módulos fundamentales como son: Agenda, Consulta Externa, Admisión, Hospitalización, Urgencias o Enfermería, Toco-cirugía, Cirugía, Trabajo Social, Imagenología, Laboratorio, Patología, Banco de Sangre, Caja, Relaciones Publicas, Farmacia y Almacén y Tablero de Control. Este sistema es una aplicación de escritorio. (5)

1.2.4. Sistema de Información Hospitalaria del CESIM

El Sistema de Información Hospitalaria del CESIM es un sistema de gestión hospitalaria que permite a los hospitales la recolección, almacenamiento, procesamiento, recuperación y comunicación de información de atención al paciente y administrativa para todas las actividades relacionadas con el hospital. El mismo está concebido para llevar el control de las actividades de salud orientadas a los pacientes, permitiendo además gestionar y controlar los recursos de cada una de las áreas de las instituciones hospitalarias. El módulo de Citas permite asignar citas de primera, sucesiva, interconsulta, charla, pre-empleo y terminación. Además brinda la posibilidad de buscar citas para modificarlas o eliminarlas en caso de ser necesario. Gestiona los procesos correspondientes a los estudios especializados asociados a la especialidad de Gastroenterología y algunos estudios especializados específicos pertenecientes a otras especialidades. (6)

Ventajas

- Dispone de una información correcta y puntual la cual contribuye a la toma de decisiones.
- Permite organizar internamente los diferentes procesos que se llevan a cabo en la institución que se despliegue.
- Las funcionalidades del sistema permiten ahorrar recursos, ya que ayuda a que la planificación en las diferentes unidades médicas sea mucho más exacta y permite llevar un estricto control de los recursos.
- Desde el punto de vista social mejora sustancialmente la calidad de la atención a los pacientes y brinda herramientas para la prevención de enfermedades, epidemias, entre otras.

Según el análisis realizado previamente se llega a la conclusión de que los Sistemas de Información Hospitalaria x-HIS, Galenhos y Sigho no brindan información suficiente sobre las funcionalidades con que cuentan o sobre las herramientas con las que fueron desarrollados, por lo que se hace difícil decidir si se ajustan o no a las condiciones del Sistema de Información Hospitalaria del CESIM.

Estos sistemas poseen licencia de software propietario y están desarrollados bajo tecnologías privativas impidiendo acceder o modificar su código fuente y no cumplen con la política para el desarrollo de la informática en el sector de la salud en Cuba, definiendo que todos los productos y servicios informáticos se realizarán sobre sistemas abiertos, utilizando software libre y de calidad, por

lo que se excluye la posibilidad de integración de estos sistemas con el Sistema de Información Hospitalaria del CESIM.

Además la utilización de alguno de estos sistemas no resolvería el problema identificado, ya que, el mismo, está centrado en cómo reestructurar la gestión de los procesos relacionados con los estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica, por lo que se empleó lo aprendido en el estudio de los mismos, para mejorar la calidad y aumentar la eficiencia de la solución.

1.3. Tendencias y tecnologías actuales a utilizar.

1.3.1. Arquitectura en 3 capas

Se utilizó la arquitectura basada en capas la cual se enfoca en la distribución de roles y responsabilidades de forma jerárquica, lo que proporciona una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. Entre sus principales características se pueden mencionar por ejemplo que describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas. Además, las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas). También cabe resaltar que los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas; por lo que este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella. (7)

Esta arquitectura es la más utilizada actualmente y está diseñada en:

- Capa de presentación: es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser fácil de entender y utilizar para el usuario.
- Capa de negocio: es donde residen los programas que se ejecutan. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los

resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

- Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

1.3.2. Patrón de arquitectura Modelo-Vista-Controlador

Se hace uso del patrón de arquitectura de software Modelo Vista Controlador (MVC) el cual separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Este se ve frecuentemente en aplicaciones web, donde la vista presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. El controlador responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y en la vista. El modelo es la representación específica de la información con la cual el sistema opera. Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). (8)

1.3.3. Modelo Cliente Servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa denominado servidor, que le da respuesta. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma. (9)

Entre las características fundamentales que presenta la arquitectura cliente-servidor se encuentran las siguientes:

- El Cliente y el Servidor pueden actuar como una sola entidad o como entidades separadas, y realizar actividades o tareas independientes.
- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes. Esto se logra al combinar los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, lo que proporciona el servicio más efectivo para el usuario final.

1.4. Tecnologías utilizadas en el proceso de desarrollo.

1.4.1. JSF

JavaServerFaces (JSF) es un framework de componentes de interfaz de usuario del lado del servidor para aplicaciones web basadas en Java. Contiene dos librerías de etiquetas personalizadas para JavaServerPages (JSP) que permiten expresar una interfaz JavaServer Faces dentro de una página JSP. Incluye un API para representar componentes de interfaz de usuario (UI), manejar sus estados, manejar sus eventos, la validación del lado del servidor y la conversión de datos, definir la navegación entre páginas y proporcionar extensibilidad para todas estas características. JSF es un framework de desarrollo basado en el patrón MVC. Ofrece una clara separación entre el comportamiento y la presentación. Une los componentes UI con los conceptos de la capa-web sin limitarse a una tecnología de script o lenguaje de marcas particular. (10)

1.4.2. Librería RichFaces

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. RichFaces se integra perfectamente en el ciclo de vida de JSF, incluye

funcionalidades Ajax, 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). Es posible crear interfaces de usuario de manera rápida y eficiente, basado en componentes que están listos para usar y son altamente configurables. RichFaces, además, es un proyecto que fue desarrollado con una arquitectura abierta para que fuera compatible con la mayor cantidad de entornos. (11)

1.4.3. Ajax4JSF

Ajax4jsf es una librería de código abierto 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 se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones automáticas al servidor y controlar cualquier evento de usuario. (12)

1.4.4. Java

Java nace como un lenguaje de programación fácil de utilizar, lo que lo convierte en uno de los lenguajes más elaborados y utilizados para la creación de software. Este permite a los desarrolladores desarrollar software en un Sistema Operativo y ejecutarlo en prácticamente cualquier otro, crear programas para que funcionen en un navegador web y en servicios web, combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados, así como desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital. (13)

1.4.5. JBoss Seam

JBoss Seam es un framework desarrollado por JBoss que combina a los 2 frameworks Enterprise JavaBeans EJB3 y JavaServerFaces JSF. Se puede acceder a cualquier componente EJB desde la capa de presentación refiriéndote a él mediante su nombre de componente Seam.

Seam introduce el concepto de contextos. Cada componente existe dentro de un contexto. El contexto conversacional por ejemplo captura todas las acciones del usuario hasta que éste sale del

sistema o cierra el navegador, inclusive puede llevar un control de múltiples pestañas y mantiene un comportamiento consistente cuando se usa el botón de regresar del navegador. (14) (15) (16)

1.4.6. JBoss Server

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro, lo que permite que sea utilizado en cualquier sistema operativo que lo soporte. Implementa todo el paquete de servicios de J2EE y es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4. Combina una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado, y distribuido sin restricciones por la licencia. (17)

1.4.7. PostgreSQL

PostgreSQL, es un SGBD, que constituye una mejora del sistema de base de datos Postgres, ya que mantiene los puntos fuertes de su antecesor como ser su poderoso modelo de datos y toda su riqueza en cuanto a tipos de datos, y remplaza el lenguaje de consultas PostQuel con un subconjunto más extenso de SQL. Es un proyecto de código abierto mucho más puro, ya que no tiene empresas comerciales detrás como es MySQL o Firebird. PostgreSQL no tiene costo asociado por lo que cualquiera puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. PostgreSQL presenta alta concurrencia, para esto utiliza la tecnología de Control de Concurrencia Multi-Versión (Multiversion Concurrency Control (MVCC)), con lo que se logra que ningún lector sea bloqueado por un escritor. Es altamente extensible, soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. (17)

1.4.8. Enterprise JavaBeans (EJB3)

Una de las metas de la arquitectura EJB es la de poder escribir de manera fácil aplicaciones de negocio orientadas a objetos y distribuidas, basadas en el lenguaje de programación Java. El propósito de EJB3 es el de proveer el soporte de la arquitectura de EJB y al mismo tiempo reducir la complejidad para el desarrollo de aplicaciones empresariales. (18)

1.4.9. Java Persistence API (JPA)

Es una API de persistencia de POJOs (Plain Old Java Object), es decir, objetos simples que no heredan ni implementan otras clases (como los EJBs). En su definición, ha combinado ideas y

conceptos de los principales frameworks de persistencia, como Hibernate, Toplink y JDO (Java Data Object) y de las versiones anteriores de EJB. Es una especificación de Oracle para la persistencia de objetos Java a cualquier base de datos relacional. Proporciona un estándar para gestionar datos relacionales en aplicaciones Java SE o Java EE, de forma que además se simplifique el desarrollo de la persistencia de datos. Esta API fue desarrollada para la plataforma JEE. (19)

1.4.10. Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java 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) que permiten establecer estas relaciones además de diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se podrán generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySQL, entre otros. Además, es Open Source, lo que supone, entre otras cosas, que no se tiene que pagar nada por adquirirlo.

Entre las principales características técnicas que aporta Hibernate están las siguientes:

- Modelo de programación natural. Hibernate es una capa de persistencia objeto/relacional que permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.
- Gran escalabilidad. Hibernate es muy eficiente, tiene una arquitectura de caché de doble capa y podría ser usado en un clúster.
- Persistencia transparente. Ofrece soporte para un amplio conjunto de las colecciones de Java, propiedades del estilo de persistencia de JavaBeans, etc. que abstraen al usuario.
- Mapeado flexible gracias a las asociaciones bidireccionales, la persistencia transitiva, colecciones de tipos básicos, etc. el mapeado resulta mucho más flexible.
- Facilidades en consultas. Debido en parte a que se realizan en un potente lenguaje de consultas orientado a objetos.

- Facilidades en metadatos. Soporta el formato del mapeado de XML, diseñado para ser editado a mano y el mapeado basado en anotaciones. Además de validación basada en anotaciones. (20)

1.5. Interfaces de comunicación.

1.5.1. Lenguaje de Modelado UML

Permite modelar, construir y documentar los elementos que forman un sistema software Orientado a Objetos. UML7 se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). (21)

1.5.2. Herramienta de Modelado Visual Paradigm

Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta CASE, proporciona además abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (22)

1.5.3. Notación para el Modelado de Procesos de Negocio (BMPN)

El objetivo principal de esta notación, es mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio, que se deben modelar, automatizar, integrar, monitorizar y optimizar de forma continua. A través del modelado de las actividades y procesos puede lograrse un mejor entendimiento del negocio y muchas veces esto presenta la oportunidad de mejorarlos. La automatización de los procesos reduce errores, asegurando que los mismos se comporten siempre de la misma manera y dando elementos que permitan visualizar el estado de los mismos. La administración de los procesos permite asegurar que los mismos se ejecuten eficientemente, y la obtención de información que luego puede ser usada para mejorarlos. Es a través de la información

que se obtiene de la ejecución diaria de los procesos, que puede identificar posibles ineficiencias en los mismos, y actuar sobre ellas para optimizarlos. (23)

1.5.4. Metodología de Desarrollo RUP

Es el resultado de varios años de trabajo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML (Unified Modeling Language), y trabajo de muchas metodologías utilizadas por los clientes. En RUP se han agrupado las actividades en grupos lógicos en los que se definen nueve flujos de trabajo principales, los seis primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo. El ciclo de vida de RUP se caracteriza por ser dirigido por caso de uso, centrado en la arquitectura, iterativo e incremental. (24)

1.5.5. XML

XML3 es un metalenguaje extensible de etiquetas desarrollado por el W3C4. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fácil y fiable. XML, sin embargo permite al diseñador crear sus propias marcas, especificando su sintaxis y funcionalidad de tal forma que las nuevas marcas puedan ser reconocidas e interpretadas por los visualizadores, sirve fundamentalmente para almacenar la información de una forma estructurada, con estructuras que siguen ciertas reglas y que son accesibles de forma manual o automatizada. Proporciona un mecanismo que permite almacenar e intercambiar la información de una forma estructurada y en un formato comprensible por aplicaciones situadas en sistemas heterogéneos. (25)

1.5.6. Lenguaje de Marcado de Hipertexto Extensible (XHTML)

XHTML, acrónimo inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas

funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium de lograr una web semántica, donde la información y la forma de presentarla estén claramente separadas. (26)

1.6. Herramientas.

Con la descripción de la información reflejada anteriormente se hace paso a las herramientas que conforman el Ambiente de Desarrollo para gestionar los procesos relacionados con los estudios especializados en el Sistema de Información Hospitalaria del CESIM. Es importante destacar que se hace uso de estas herramientas por ser las utilizadas para desarrollar el Sistema de Información Hospitalaria del CESIM y con esto se da cumplimiento a una de las tareas de investigación anteriormente mencionadas.

Primeramente, se utilizará el Visual Paradigm 8.0 como herramienta CASE, esta es una herramienta que se encarga de realizar el ciclo de vida completo del desarrollo de software. La notación BPMN es utilizada para generar la documentación del negocio y el lenguaje UML 2.0 para crear los diagramas de casos de uso del flujo de requerimientos, así como los correspondientes a los de los flujos de diseño e implementación. Como cliente de base de datos pgAdmin III, debido a que es un ambiente relacional, gratuito, poderoso y puede ser ejecutado sobre la mayoría de los sistemas operativos que existen hoy en día.

Por último se hará uso del Eclipse, un Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés) que tiene la posibilidad de añadir funcionalidades nuevas al editor, a través de nuevos módulos, lo que es beneficioso para la utilización de JBoss Tools, conjunto de herramientas que posibilitan el desarrollo desde el IDE Eclipse con RichFaces, Seam e Hibernate, además de realizar la administración y configuración del servidor JBoss AS.

Conclusiones parciales:

Los autores del presente trabajo consideran que el estudio del capítulo evidenció que los sistemas existentes mencionados en el estudio del arte que gestionan procesos relacionados con estudios especializados en las instituciones hospitalarias, no pueden ser integrados por las razones mencionadas con anterioridad y no se cuenta con información suficiente acerca de los mismos. Esto evidenció la necesidad de desarrollar las funcionalidades relacionadas con la gestión de los procesos

de estudios especializados para que el usuario pueda hacer un correcto uso de estas y también para que exista una documentación para desarrollos futuros y se confeccione con tecnologías y herramientas no privativas. Además, se justificó la aplicación del patrón de arquitectura y el empleo de las tecnologías y herramientas propuestas para la obtención de las funcionalidades.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

En este capítulo se presenta formalmente el problema científico, el objeto de automatización, punto en el cual se describen los procesos de negocio, así como una descripción de los documentos que se procesan y la información que se manipula. Además se exponen el modelo de negocio, la especificación de requerimientos funcionales y no funcionales del sistema y finalmente se muestran los diagramas de casos de uso como respuesta a la situación problémica actual.

2.1. Flujo actual de los procesos en el campo de acción.

2.1.1. Asignar cita para estudio especializado.

Permitir asignar una cita para estudios especializados al paciente que solicita asistencia médica.

2.1.1.1. Buscar cupo de cita para estudio especializado.

El paciente se presenta con una solicitud de cita para estudios especializados hacia el técnico de registros médicos. El técnico busca el libro de control de citas y localiza un cupo disponible teniendo en cuenta el médico, el tipo de estudio y la fecha de la cita en caso de se haya reflejado. Si no tiene el médico especificado solo se tienen en cuenta el tipo de estudio y la fecha de la cita en caso de que se haya reflejado.

2.1.1.2. Asignar cupo de cita para estudio especializado.

El técnico de registros médicos luego de localizar un cupo disponible en el libro de control de citas registra en este, el número de HC del paciente, nombre y apellidos. Además escribe el tipo de estudio a realizar y la fecha del día de la consulta.

2.1.2. Gestionar horario para estudio especializado.

Permitir gestionar un horario de un estudio especializado para un médico de la institución hospitalaria.

2.1.2.1. Buscar médico para crear horario de estudio especializado.

El técnico de registros médicos busca en el libro de control de citas el médico al cual se le va crear un horario de estudio especializado.

2.1.2.2. Registrar horario de estudio especializado.

El técnico de registros médicos después de haber localizado al médico, registra en el libro de control de citas el tipo de estudio, la cantidad de pacientes y la fecha en la cual se podrá asignar la cita.

2.1.3. Objeto de automatización.

El siguiente proceso de negocio será objeto fundamental de automatización para que el sistema funcione correctamente.

Asignar cita para estudio especializado: Permitir asignar una cita al paciente que requiera la realización de estudios especializados.

Gestionar horario para estudio especializado: Permitir gestionar un horario de un estudio especializado para un médico de la institución hospitalaria.

2.2. Información que se maneja.

Existen numerosos documentos que se generan recogiendo y archivando toda la información relacionada con la planificación de citas para estudios especializados dentro de una institución hospitalaria. La automatización de estos documentos son claves para el correcto funcionamiento del sistema. A continuación se presentan los documentos que han sido seleccionados para su automatización:

Tarjeta de cita: En la tarjeta se recogen los datos de la cita una vez se haya planificado con anterioridad la cita del paciente, en la tarjeta se muestran datos como el nombre del médico que va atender al paciente, la hora de inicio y fin de la cita, la fecha en que se creó la cita, el servicio al cual pertenece ese paciente y el tipo de cita que se ha planificado. (2)

Libro de cita: Es donde se guardan las citas que tienen planificadas los médicos y el fondo de tiempo, que primeramente se llenan con los días y después se llena con los nombres de los pacientes. En el libro se anotan los días que el médico tiene que trabajar y donde se anotan los pacientes citados en los días planificados. (2)

Estos documentos son los principales en el módulo de Citas para poder realizar la planificación de las citas lo más óptima posible y para tratar de reestructurar todo el proceso de control y asignación de citas para estudios especializados.

2.3. Modelo de Negocio.

La creación de modelos que organicen y representen los detalles de situaciones reales vinculadas con sistemas a desarrollar, son necesarios para entender y analizar la complejidad del problema a resolver. Un modelo previo al desarrollo de un sistema, es el Modelo de negocio, que tiene como propósitos comprender los problemas actuales de la organización e identificar mejoras potenciales de estos, además de asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización.

El proceso de modelado permite obtener una visión de la organización que permita definir los procesos, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos.

Para realizar el modelo del negocio por procesos se siguió la notación BPMN. Esto aporta una mayor visibilidad de las actividades que se realizan y ayuda a lograr un mejor entendimiento del flujo de trabajo existente entre las áreas hospitalarias, lo que permite definir con claridad las actividades innecesarias a la hora de automatizar el negocio.

Mediante los elementos gráficos que la notación BPMN define se pueden construir los diagramas de procesos. Para un mejor entendimiento es preciso describir los elementos gráficos empleados:

Existen eventos de inicio y fin, los cuales indican una acción lógica en el flujo del proceso. Se identifican subprocesos y dentro de ellos se encuentran actividades atómicas que se relacionan por una línea continua que representa un flujo de secuencia.

Las bifurcaciones determinan la ramificación del flujo a través de decisiones inclusivas o exclusivas. Los artefactos brindan la información de los documentos o registros que se requieren y se producen en cada actividad. Estos se relacionan mediante líneas discontinuas con la actividad que lo emplea.

Los diagramas están compuestos por calles, que representan a los actores y trabajadores del negocio; donde el mayor cúmulo de actividades está en las calles de los trabajadores. En estas se muestran las actividades que realizan en orden lógico, respondiendo al proceso de negocio descrito.

A continuación se representa los diagramas de procesos de negocio que engloba los procesos asociados al objeto de estudio de la investigación:

- **Asignar cita para estudio especializado**

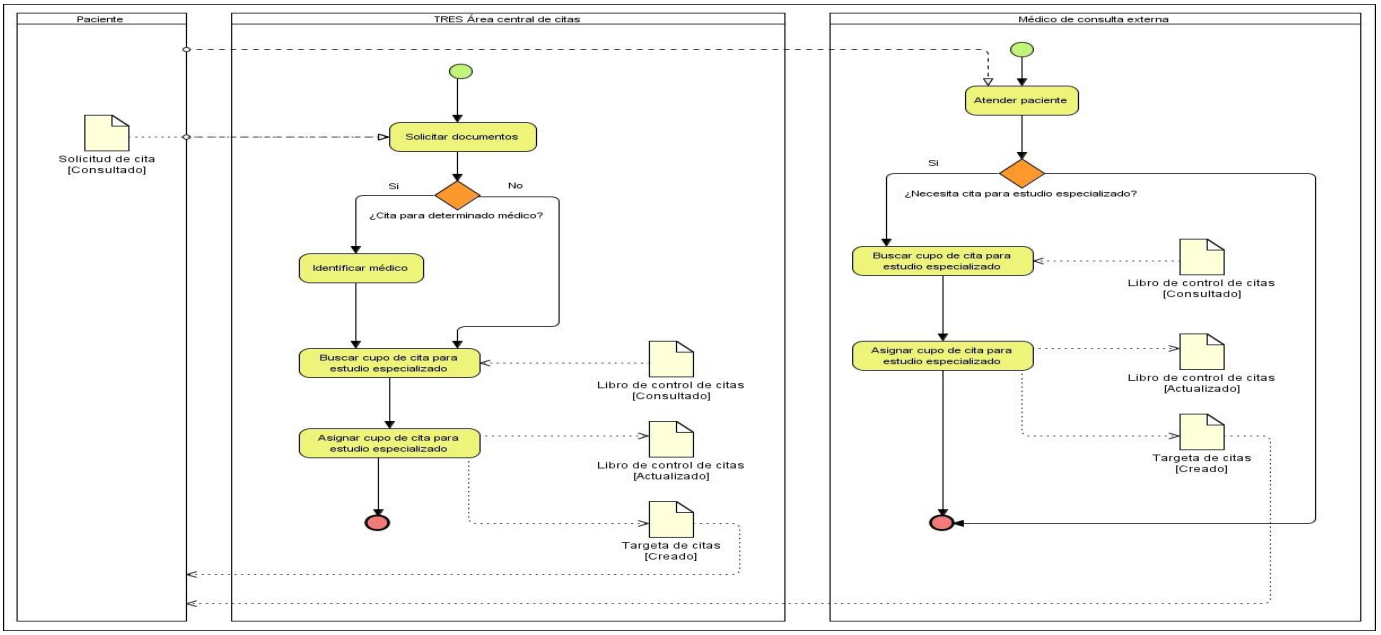


Figura 2.1 Asignar cita para estudio especializado.

- **Gestionar horario para estudio especializado**

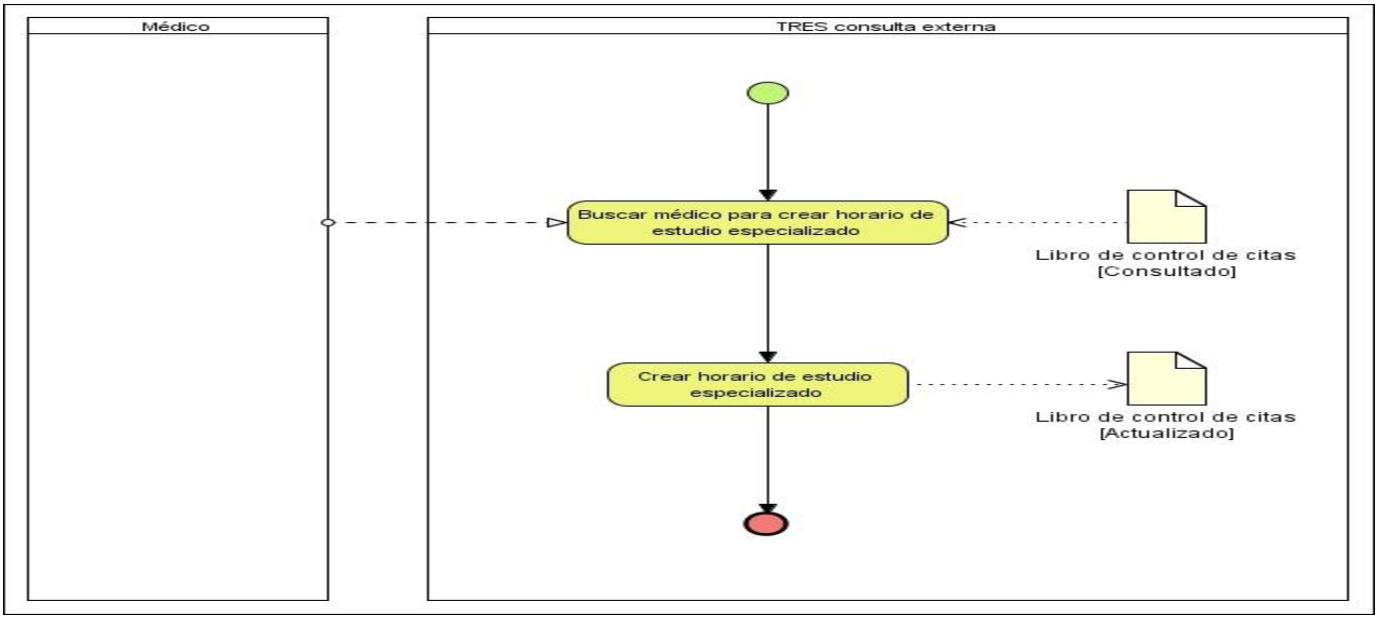


Figura 2.2 Gestionar horario para estudio especializado.

Una vez presentados los diagramas de procesos se describen los principales actores involucrados representados en los mismos:

Roles	Descripción
Paciente	Persona que llega a la institución hospitalaria para atenderse por alguna patología.
Técnico de registros y estadísticas de salud en módulo de Citas	Se encarga de buscar cupos disponibles y asignar citas para consultas de control, interconsultas, de primera, charlas y estudios especializados
Técnico de registros y estadísticas de salud en Consulta Externa.	Se encarga de crear, modificar y eliminar un horario para una cita de un médico en específico.
Médico de Consulta Externa	Se encarga de atender al paciente en la consulta y puede indicarle una cita de control o para estudio especializado.

Tabla 2.1 Trabajadores del Negocio.

2.4. Especificación de requerimientos de software.

Un requerimiento es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. (27)

2.4.1. Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir y que definen el comportamiento interno del software. Estos describen los servicios que se espera que el sistema cumpla para satisfacer las necesidades del usuario.

Módulo de Citas

1. Buscar paciente para asignar cita de estudio especializado.
2. Asignar cita para estudio especializado.
3. Buscar cita para estudio especializado.

Módulo de Configuración

1. Buscar estudio especializado.
2. Crear estudio especializado.

3. Modificar estudio especializado.
4. Eliminar estudio especializado.

Módulo Consulta Externa

1. Gestionar horario para estudios especializado.
2. Crear horario para estudios especializado.
3. Modificar horario para estudios especializado.
4. Ver datos de horario para estudios especializado.
5. Eliminar horario para estudios especializado.
6. Ver detalles de horario para estudios especializado.
7. Asignar cita para estudio especializado.
8. Modificar datos de cita para estudio especializado.
9. Ver datos de cita para estudio especializado.
10. Eliminar cita para estudio especializado.
11. Crear solicitud de estudio especializado
12. Modificar solicitud de estudio especializado
13. Ver datos de solicitud de estudio especializado
14. Eliminar solicitud de estudio especializado

2.4.2. Requisitos no funcionales.

Los requisitos no funcionales son condiciones que debe cumplir un sistema para satisfacer un contrato o una especificación. Están regidos por las necesidades del usuario para poder resolver un problema o conseguir un beneficio determinado. Se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad de los dispositivos de entrada/salida, y la representación de datos que se utilizan en las interfaces del sistema.

Estos requisitos son de gran significación en la aceptación del software, debido a que representan las ventajas más visibles al usuario y repercuten en el óptimo funcionamiento y mantenimiento del sistema. (28)

- **Usabilidad**

El sistema estará diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo en un tiempo reducido:

Usuarios normales: 20 días

Usuarios avanzados: 30 días

- **Seguridad**

Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema. Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando sólo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión. Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.

Se establecerán mecanismos de control y verificación para los procesos susceptibles de fraude. El sistema proporcionará un registro de actividades de cada usuario en el sistema. Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos, independientemente de que para el sistema, este elemento ya no exista.

- **Soporte**
    - Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP.
    - Se permitirá administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación.
    - Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo, además de recuperar la base de datos a partir de los respaldos realizados.

- Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema.
- Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

- **Hardware**

Estaciones de trabajo.

En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria del CESIM, las que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y siga los estándares web, por lo que se escogieron estaciones de trabajo de 512 MB de memoria RAM y un microprocesador de 2.0 Hz con Sistema operativo GNU/Linux.

- **Servidores**

La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.

- Servidores de Base de Datos: Servidor con cualquier tecnología o sistema operativo que soporte a PostgreSQL Server 8.4 o superior en los servidores de base de datos de cada hospital, y Oracle 10g o superior para los servidores de base de datos del Centro de Datos, 16 GB de memoria y 1 TB de disco duro.
- Servidores de Aplicaciones: 16 GB de memoria y 1 TB de disco duro con tecnología o sistema operativo que soporte el Java Runtime Environment (JRE) 1.6.0_24 o superior y al JBoss AS 4.2.2.GA.
- Servidores de Intercambio: 16 GB de memoria y 1 TB de disco duro y sistema operativo Linux.

- **Software**

El sistema debe correr en los siguientes Sistemas Operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java virtual machine, JBoss AS y PostgreSQL). Los clientes deberán disponer de

un navegador web, estos pueden ser IE 7 o superior, Opera 9 superior, Google Chrome 1 o superior y Firefox 4.X o superior.

- **Restricciones de diseño**

La capa de presentación contendrá todas las vistas y la lógica de la presentación. El flujo web se manejará de forma declarativa y basándose en definiciones de procesos del negocio. La capa del negocio mantendrá el estado de las conversaciones y procesos del negocio que concurrentemente pueden estar siendo ejecutados por cada usuario. La capa de acceso a datos contendrá las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

- **Requisitos para la documentación de usuarios en línea y ayuda del sistema.**

Se posibilitará el uso de ayudas dinámicas y tutoriales en línea sobre el funcionamiento del sistema. El objetivo principal de estos materiales es servirle de ayuda al usuario para que pueda aclarar sus dudas respecto al sistema.

- **Interfaz de usuario**

Las ventanas del sistema contendrán bien estructurados los datos, además de permitir la interpretación correcta de la información. La interfaz contará con accesos directos y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada e informada al usuario. Todos los textos y mensajes en pantalla aparecerán en idioma español.

- **Portabilidad**

El producto podrá ser utilizado bajo los sistemas operativos Linux o Windows ya que son los sistemas operativos más comunes. La aplicación podrá ser desplegada sobre Sistemas Operativos Linux (Oracle Linux 5.5+, 6.x, Ubuntu 8.04 LTS (Long Term Support) Desktop Edition, Ubuntu Linux 10.04 y superior, entre otras versiones) y Windows (2000, XP, Server 2003, Server 2008, Server 2012, Vista, 7, 8).

2.5. Modelo de casos de uso del sistema.

Los requisitos o capacidades internas del sistema a desarrollar han sido expresados como casos de uso teniendo en cuenta que estos son actividades atómicas que ofrecen un resultado definido para cada acción que realiza el usuario.

El modelo de casos de uso del sistema documenta el comportamiento del sistema desde el punto de vista del usuario, permitiendo representar las funciones que se desean en el sistema (casos de uso), el entorno del sistema (actores), y las relaciones entre ellos. Aunque la parte más visible de dicho modelo son los diagramas de casos de uso, suele ir acompañado de una especificación textual de cada uno de los casos de uso.

Los actores del sistema no forman parte del sistema, sino que representan elementos que interactúan con él. Estos elementos son nombrados roles que puede estar formado por una o varias personas, un equipo o un sistema automatizado. Un actor puede introducir o recibir información del sistema. (29)

2.5.1. *Definición de actores del sistema.*

- **Técnico (a) de registros y estadísticas de salud en módulo de Citas**

Es la persona encargada de asignar, buscar, modificar y eliminar una cita para cualquier servicio del hospital.

- **Técnico (a) de registros y estadísticas de salud en consulta externa**

Es la persona encargada de crear, modificar y eliminar un horario para una cita de un médico en específico.

- **Administrador del sistema**

Es la persona encargada de crear, modificar, eliminar un estudio especializado.

- **Médico de consulta externa**

Se encarga de atender al paciente en la consulta y puede indicarle una cita de control, cita para estudios especializados o una solicitud de estudio especializado determinada.

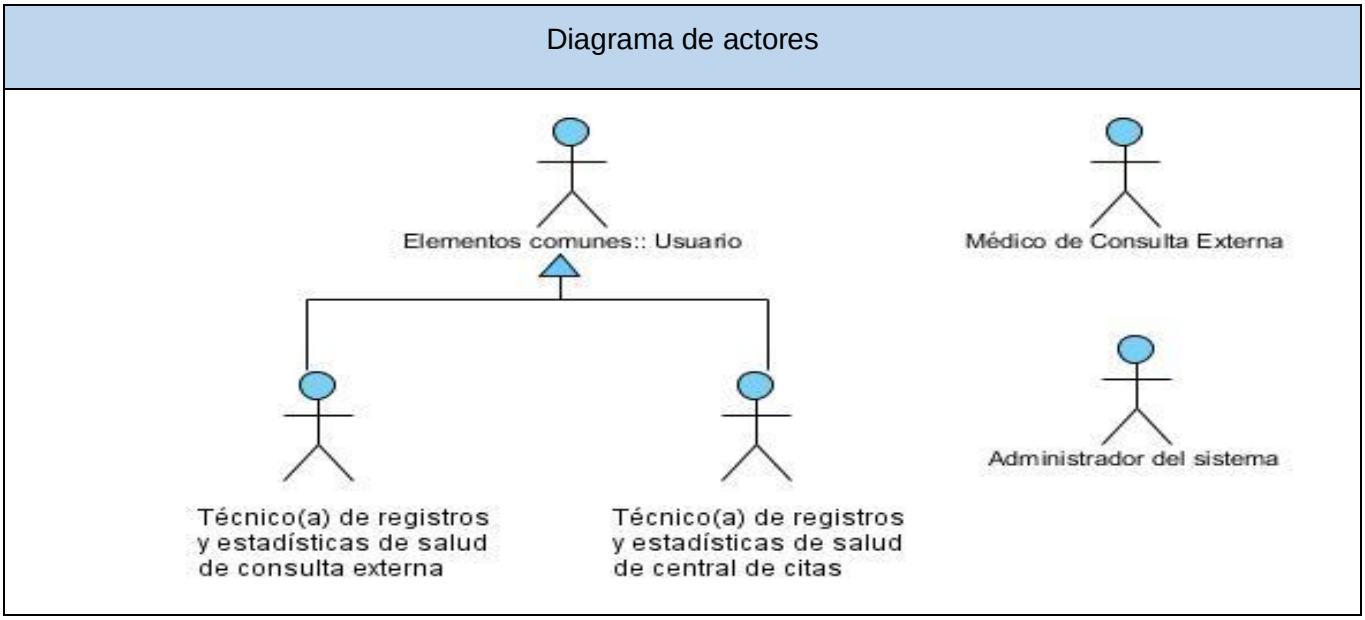


Figura 2.3 Diagrama de Actores.

2.5.1.1. Diagrama de casos de uso.

- **Asignar cita para estudio especializado en el módulo de Citas**

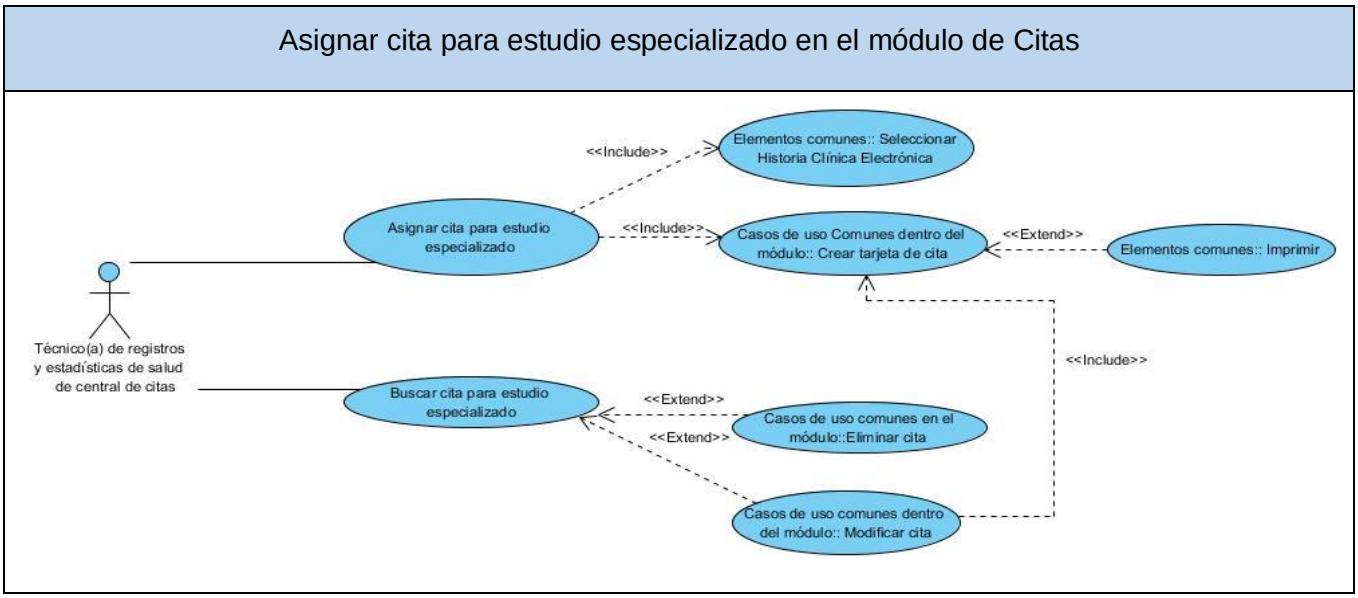


Figura 2.4 Asignar cita para estudio especializado en el módulo de Citas.

- **Asignar cita para estudio especializado desde Consulta Externa**

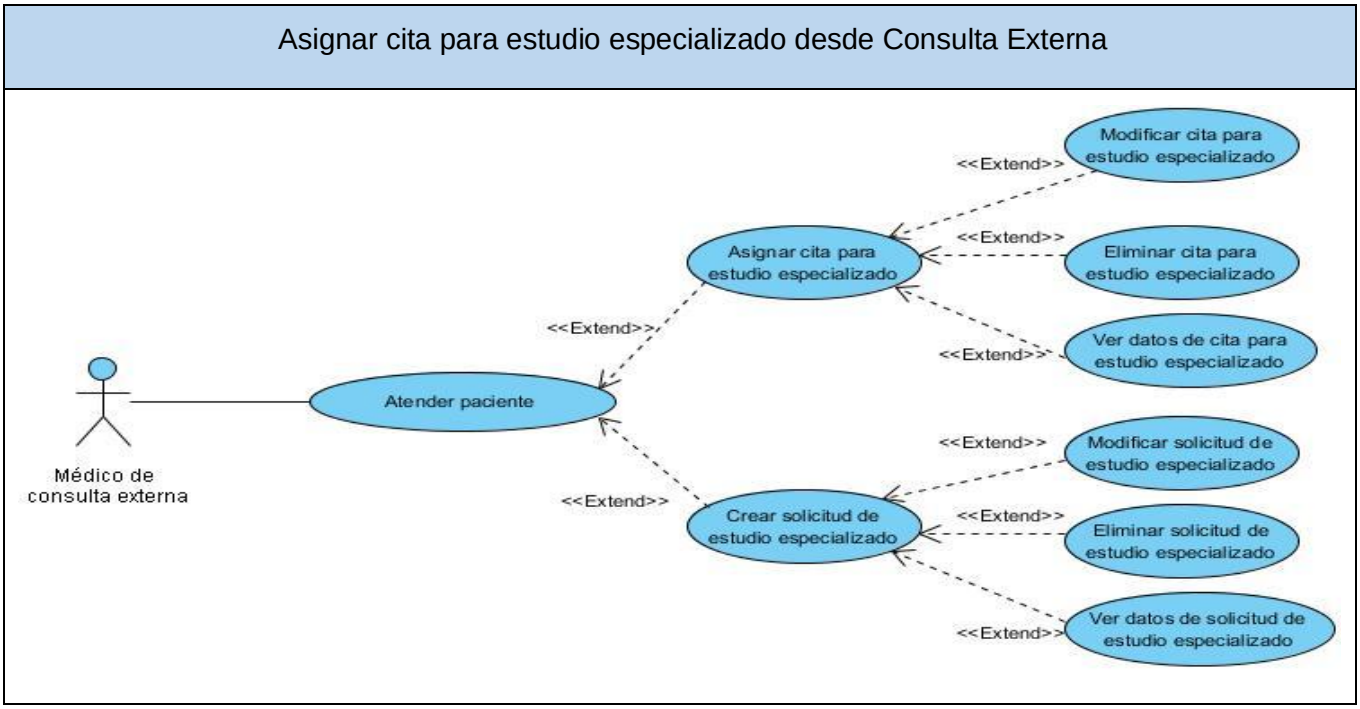


Figura 2.5 Asignar cita para estudio especializado desde Consulta Externa.

- **Buscar estudio especializado**

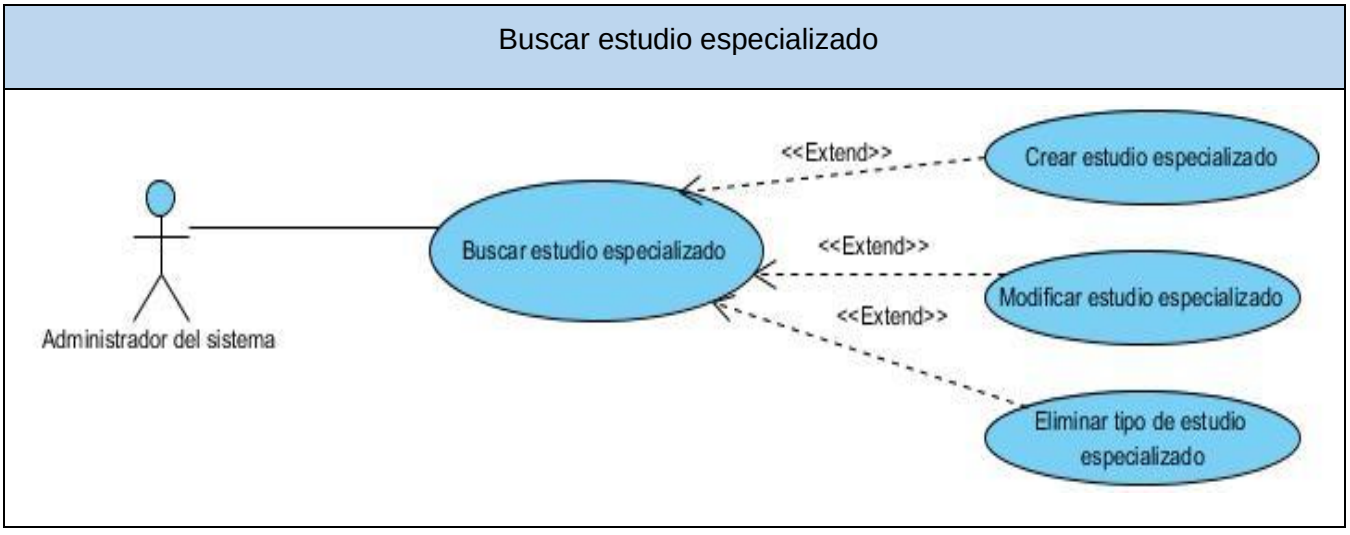


Figura 2.6 Buscar estudio especializado.

- **Gestionar horario de cita para estudio especializado**

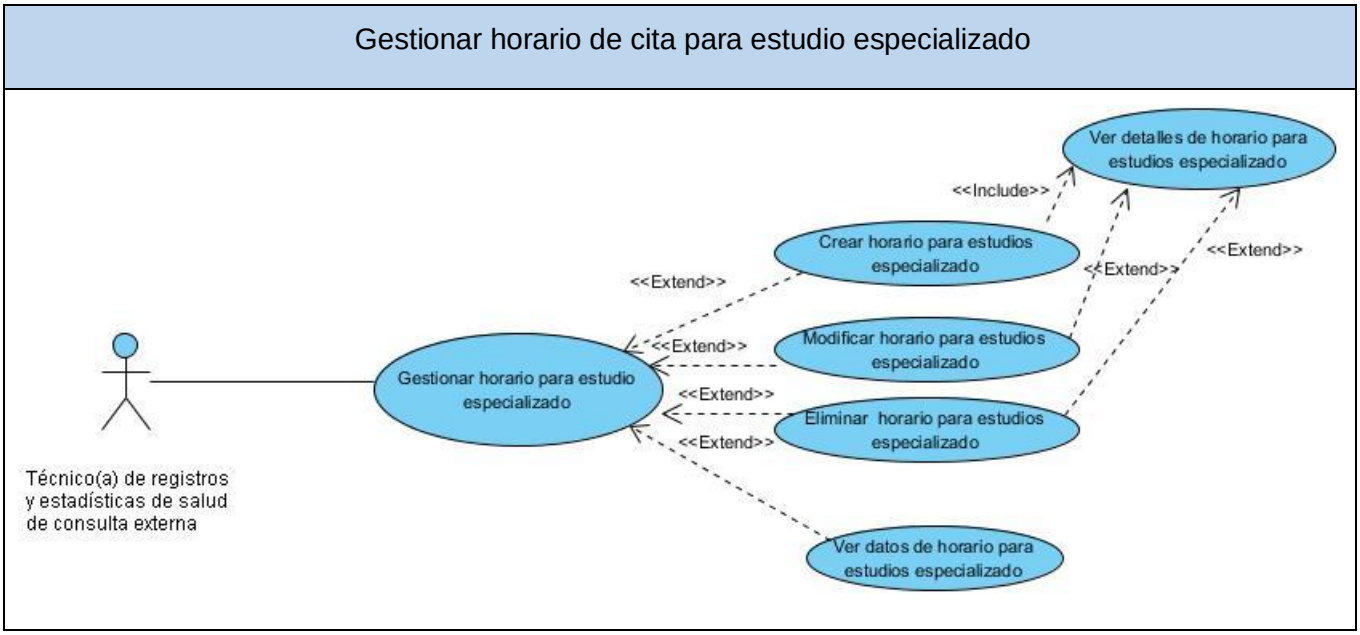


Figura 2.7 Gestionar horario de cita para estudio especializado.

2.5.1.2. Descripción textual de los casos de uso.

Asignar cita para estudio especializado en el módulo de Citas	
Propósito	Asignar cita para estudio especializado en la módulo de Citas
Actores:	Técnico (a) de registros y estadísticas de salud de módulo de Citas
Resumen:	El caso de uso inicia cuando el actor accede a la opción Asignar cita para estudio especializado, el sistema brinda la posibilidad de seleccionar estudio, médico y fecha para citar al paciente, el actor selecciona el estudio, el médico, la fecha y confirma la cita, el sistema crea la cita para estudio especializado y la Tarjeta de cita, el caso de uso termina.
Precondición	No existe
Poscondición	Se creó una Cita para estudio especializado
Referencias:	RF_2 del Módulo Citas

Tabla 2.2 Asignar cita para estudio especializado en el módulo de Citas.

Buscar cita para estudio especializado desde el módulo de Citas	
Propósito	Buscar cita para estudio especializado desde la módulo de Citas
Actores:	Técnico (a) de registros y estadísticas de salud de módulo de Citas
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar cita para estudio especializado, el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar la cita, el actor introduce los datos que considera como criterios para realizar una búsqueda, el sistema busca y muestra las citas que cumplen con los criterios de búsqueda, el sistema permite modificar o eliminar una cita para estudio especializado, el caso de uso termina.
Precondición	No existe
Poscondición	Se buscó cita para estudio especializado dado criterios
Referencias:	RF_3 del Módulo Citas

Tabla 2.3 Buscar cita para estudio especializado desde el módulo de Citas.

Asignar cita para estudio especializado en Consulta Externa	
Propósito	Asignar cita para estudio especializado en consulta externa
Actores:	Médico de consulta externa
Resumen:	El caso de uso inicia cuando el actor accede a la opción Asignar cita para estudio especializado, el sistema brinda la posibilidad de seleccionar estudio y fecha para citar al paciente, el actor selecciona el estudio, la fecha y confirma la cita, el sistema crea la cita para estudio especializado y la Tarjeta de cita, el caso de uso termina.
Precondición	No existe
Poscondición	Se creó una Cita para estudio especializado
Referencias:	RF_7 del Módulo Consulta Externa

Tabla 2.4 Asignar cita para estudio especializado en consulta externa.

Crear estudio especializado	
Propósito	Crear estudio especializado
Actores:	Administrador del sistema
Resumen:	El caso de uso inicia cuando el actor accede a la opción Adicionar estudio especializado, el sistema brinda la posibilidad de introducir los datos para crear el estudio especializado, el actor introduce los datos del estudio especializado, el sistema crea el estudio especializado, el caso de uso termina.
Precondición	No existe
Poscondición	Se creó un estudio especializado
Referencias:	RF_2 del Módulo Configuración

Tabla 2.5 Crear estudio especializado.

Modificar estudio especializado	
Propósito	Modificar estudio especializado
Actores:	Administrador del sistema
Resumen:	El caso de uso inicia cuando el actor selecciona un estudio especializado y accede a la opción Modificar estudio especializado, el sistema muestra los datos del estudio especializado y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos del estudio especializado, el caso de uso termina.
Precondición	No existe
Poscondición	Se modificó un estudio especializado
Referencias:	RF_3 del Módulo Configuración

Tabla 2.6 Modificar estudio especializado.

Eliminar estudio especializado	
Propósito	Eliminar estudio especializado
Actores:	Administrador del sistema
Resumen:	El caso de uso inicia cuando el actor selecciona un estudio especializado y accede a la opción Eliminar estudio especializado, el sistema elimina el estudio especializado, el caso de uso termina.
Precondición	No existe
Poscondición	Se eliminó un estudio especializado
Referencias:	RF_4 del Módulo Configuración

Tabla 2.7 Eliminar estudio especializado.

Crear horario para estudio especializado	
Propósito	Crear horario para estudio especializado
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor accede a la opción Adicionar horario, el sistema brinda la posibilidad de introducir los datos para crear el horario del estudio especializado, el actor introduce los datos del horario del estudio especializado, el sistema crea el horario del estudio especializado, el caso de uso termina.
Precondición	No existe
Poscondición	Se creó un horario para cita de estudio especializado
Referencias:	RF_2 del Módulo Consulta Externa

Tabla 2.8 Crear horario para estudio especializado.

Modificar horario para estudio especializado	
Propósito	Modificar horario para estudio especializado
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa

Resumen:	El caso de uso inicia cuando el actor selecciona un estudio especializado y accede a la opción Modificar horario, el sistema muestra los datos del horario y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos del horario, el caso de uso termina.
Precondición	No existe
Poscondición	Se modificó un horario para estudio especializado
Referencias:	RF_3 del Módulo Consulta Externa

Tabla 2.9 Modificar horario para estudio especializado.

Eliminar horario para estudio especializado	
Propósito	Eliminar horario para estudio especializado
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor selecciona un horario y accede a la opción Eliminar, el sistema elimina el horario, el caso de uso termina.
Precondición	No existe
Poscondición	Se eliminó un horario para estudio especializado
Referencias:	RF_5 del Módulo Consulta Externa

Tabla 2.10 Eliminar horario para estudio especializado.

Ver datos de horario para estudio especializado	
Propósito	Ver datos de horario para estudio especializado
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor selecciona un horario y accede a la opción Ver del Gestionar horario para estudio especializado, el sistema muestra los datos del horario, el caso de uso termina.
Precondición	No existe

Poscondición	Se mostraron los datos de un horario para estudio especializado
Referencias:	RF_4 del Módulo Consulta Externa

Tabla 2.11 Ver datos de horario para estudio especializado.

Ver detalles de horario para estudio especializado	
Propósito	Ver detalles de horario para estudio especializado
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor crea un horario, el sistema muestra el ver detalles del horario y permite eliminar o modificar el horario, el caso de uso termina.
Precondición	No existe
Poscondición	Se mostraron detalles de un horario para estudio especializado
Referencias:	RF_6 del Módulo Consulta Externa

Tabla 2.12 Ver detalles de horario para estudio especializado.

Conclusiones parciales:

El desarrollo de este capítulo ha permitido analizar los procesos del negocio que se desarrollan en el módulo de Citas y Consulta Externa referente a los procesos de asignación de citas y gestión de horarios para estudios especializados respectivamente. Esto ha hecho posible alcanzar un mejor entendimiento del sistema y las restricciones que deben existir para satisfacer las necesidades de los clientes. Además, se especificaron todos los requerimientos funcionales y no funcionales del sistema, y se identificaron los actores que intervienen, lo cual permitió obtener una guía para el desarrollo de las funcionalidades a desarrollar. También se definieron los casos de uso del sistema con su descripción asociada que reflejan las funcionalidades recogidas previamente en los requerimientos, lo que permitió una mejor comprensión del proceso de negocio.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DEL SISTEMA.

En el presente capítulo se expone la arquitectura a utilizar y el análisis de posibles implementaciones, componentes o módulos que podrían ser rehusados así como las estrategias de integración. Además se representa los distintos diagramas de clases referentes al diseño y el diagrama de paquetes. Se da una breve descripción de las clases entidades y controladoras usadas en el flujo de trabajo Análisis y Diseño.

3.1. Arquitectura.

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

El sistema que se propone presenta una arquitectura basada en uno de los estilos más utilizados del patrón arquitectónico Modelo Vista Controlador (MVC), expuesto en el Capítulo 1: el patrón en capas, el cual separa los elementos de la presentación, el negocio y el acceso a datos del sistema, para lograr que cada capa se comunique con sus adyacentes, permitiendo que los cambios de una capa puedan realizarse sin afectar a todo el sistema.

El uso del framework JSF permite contar con la implementación de dicho patrón. Brindándole a la aplicación la posibilidad de tener una separación clara entre cómo se muestra la información al usuario, cómo se manejan las acciones que el usuario desea hacer sobre el sistema y cómo se realizan estas acciones modificando y validando la información.

Este patrón se evidencia de la siguiente forma: la Vista se corresponde con las páginas xhtml las cuales son interfaces de usuario que le presentan el sistema a este, manejan las acciones realizadas sobre la interfaz por el usuario y recogen la información entrada por el mismo. El controlador se corresponde con las clases controladoras para cada caso de uso, que se encargan del procesamiento de la información en correspondencia con la lógica del negocio en cuestión. La información manejada en todo el sistema coincide con el modelo que es una representación orientada a objetos, en forma de clases de entidad, de las tablas de la base de datos del sistema.

3.2. Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados. Estrategias de integración.

Con el objetivo de reducir tiempo, minimizar las redundancias y aprovechar el trabajo realizado anteriormente, la práctica de reutilización de código fuente trae consigo un conjunto de ventajas para toda aplicación que se encuentre en desarrollo. La creación de componentes reutilizables para evitar la duplicidad del mismo es la forma más eficiente de esta reutilización. La manera más fácil de llevar a cabo esta reutilización, es copiar total o parcialmente, pero resulta trabajoso mantener múltiples copias del mismo código, por lo que es recomendable mantener el código reusable en un solo lugar, para luego ser llamado en donde sea requerido.

En el Sistema de Información Hospitalaria del CESIM se utilizan varios componentes comunes en todos los módulos de dicho sistema, con el fin de lograr uniformidad en el desarrollo y mejorar la calidad del trabajo. Componentes como la clase ActiveModule con el objetivo de conocer en qué módulo y entidad se encuentra el usuario que está utilizando el sistema, la clase Bitácora para tener el control de las trazas de todas las acciones que se realizan con la aplicación y la clase User para saber qué usuario está trabajando con la aplicación en tiempo real; se emplean en la reutilización de código.

3.3. Modelo de diseño.

Mediante el modelo de diseño se hace un refinamiento del proceso de análisis. Para ello se tienen en cuenta los requisitos no funcionales del sistema ya que el principal propósito del modelado del diseño es crear un plano del modelo de implementación. Los casos de uso son realizados por las clases del diseño y sus objetos, a partir de los cuales se forma el diagrama de clases del diseño.

Patrones de diseño: Los patrones de diseño son el soporte de las soluciones a problemas comunes en el desarrollo de un software y de otros contornos referentes al diseño de interacción o interfaces. A su vez brindan una solución ya aprobada y documentada de problemas de desarrollo del software que están sometidos a contextos equivalentes. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (30)

En el desarrollo de las funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del CESIM se emplean los patrones generales para la asignación de responsabilidades, comúnmente conocidos como GRASP. A continuación se muestran las características de los patrones empleados:

Experto: es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

Creador: ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase.

Alta cohesión: expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase.

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Su empleo se evidencia en la asignación de responsabilidades asignando a cada clase las responsabilidades o tareas que pueden realizar según la información que poseen, poniéndose de manifiesto los patrones Experto y Creador. Logrando así conservar el encapsulamiento ya que los objetos realizan lo que se les pide a partir de la información que poseen.

Por su parte el empleo de los patrones Bajo acoplamiento y Alta cohesión, se pone de manifiesto cuando se modelan las clases controladoras, encargadas de realizar las operaciones del sistema, el diseño de la aplicación permite la interacción entre las mismas sin afectar su reutilización y su correcto funcionamiento por separado.

Es preciso describir los términos empleados en el modelo de diseño para un mejor entendimiento del mismo:

Diagrama de clase de diseño: Una clase de diseño es una abstracción de una clase o construcción en la implementación del sistema y un diagrama de clases de diseño expone un conjunto de interfaces, colaboraciones y sus relaciones. Se utiliza para modelar la vista de diseño estática de un sistema. Permite visualizar, especificar y documentar modelos estructurales. Esta forma parte de las realizaciones de casos de uso. (31)

Para la elaboración del modelo de diseño, se define una estructura de paquetes que permite dividir el sistema en fragmentos manejables para su futura implementación. Se emplea el criterio de empaquetamiento por proceso, siguiendo la estructura de procesos definidos en el sistema.

Los paquetes son graficados mostrando la relación que guardan entre sí. Estos utilizan el paquete repositorio de clases para su funcionamiento.

Un paquete referente a procesos, está conformado por subpaquetes que responden a las realizaciones de casos de uso, donde cada una de ellas contiene un diagrama de clases del diseño y los respectivos diagramas de secuencia.

El paquete repositorio contiene tres subpaquetes, uno para las entidades, otro para las sesiones y otro para las vistas. En el subpaquete de entidades se encuentran las clases autogeneradas definidas en el diseño de acuerdo a las tecnologías que serán usadas en la implementación. Las clases autogeneradas, como su nombre lo indica, se autogeneran desde la base de datos utilizando el ORM Hibernate (permite la generación de objetos java desde la base de datos). Las clases personalizadas son aquellas que se modifican, por lo que pueden heredar de las entidades autogeneradas.

El subpaquete de sesiones está conformado por las clases controladoras autogeneradas por el entorno de desarrollo, además de las clases controladoras personalizadas son las especializaciones de las controladoras autogeneradas y las clases controladoras del proceso, que son las clases controladoras de las funcionalidades automatizadas.

El subpaquete Vistas está compuesto por los contenidos web referentes a las páginas clientes y a los formularios de cada una.

A continuación se muestra el modelo de diseño del sistema:

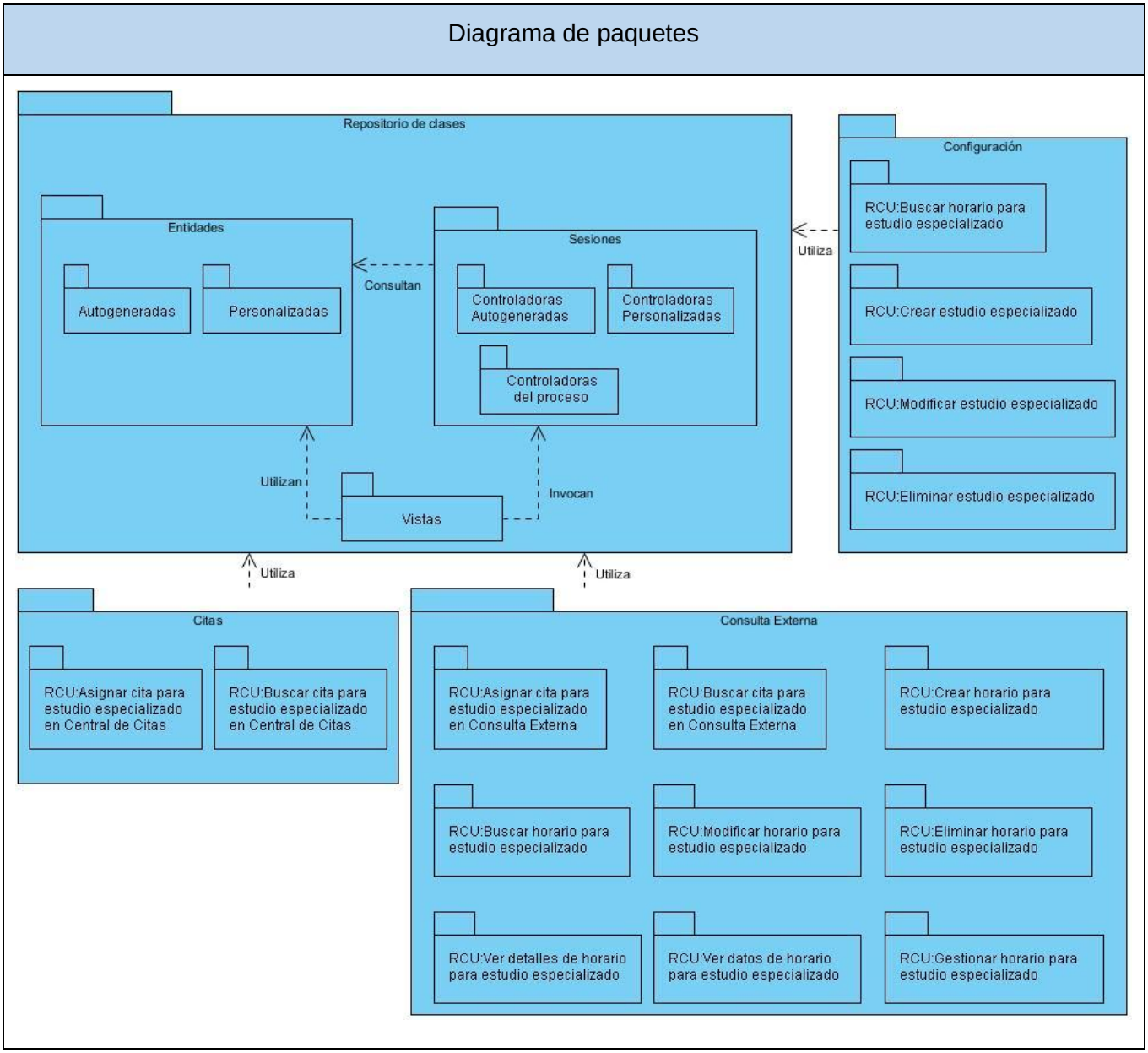


Figura 3.1 Diagrama de paquetes.

Las clases del diseño están agrupadas en:

Páginas Servidoras: Están compuestas por componentes Facelets, RichFaces, JSF, Seam UI, así como código HTML. Todo este código será ejecutado en el servidor web, generando páginas clientes que pueden ser representadas por los navegadores web.

Páginas Clientes: Están compuestas por código HTML, CSS, JavaScript. Son interpretadas por los navegadores web presentándole al usuario la interfaz con la que puede interactuar con el sistema.

Formularios: Un formulario HTML es una sección de un documento enmarcado entre tags <form> y que puede contener elementos especiales llamados controles (casillas de verificación (checkboxes), radio botones (radio buttons), menús, entre otros.), y rótulos (labels) en esos controles. Los usuarios normalmente "completan" un formulario modificando sus controles (introduciendo texto, seleccionando objetos de un menú, etc.), y lo envían al servidor donde estos son procesados. Es una manera de obtener en el servidor información entrada por el usuario en el cliente.

Controladoras: Las clases controladoras o controladoras como también se les dice son clases que implementan la lógica del negocio que se está informatizando, generalmente cada una de estas se encargan de la implementación de un caso de uso o un proceso en dependencia de la complejidad de los mismos.

A continuación se muestran los diagramas de clases:

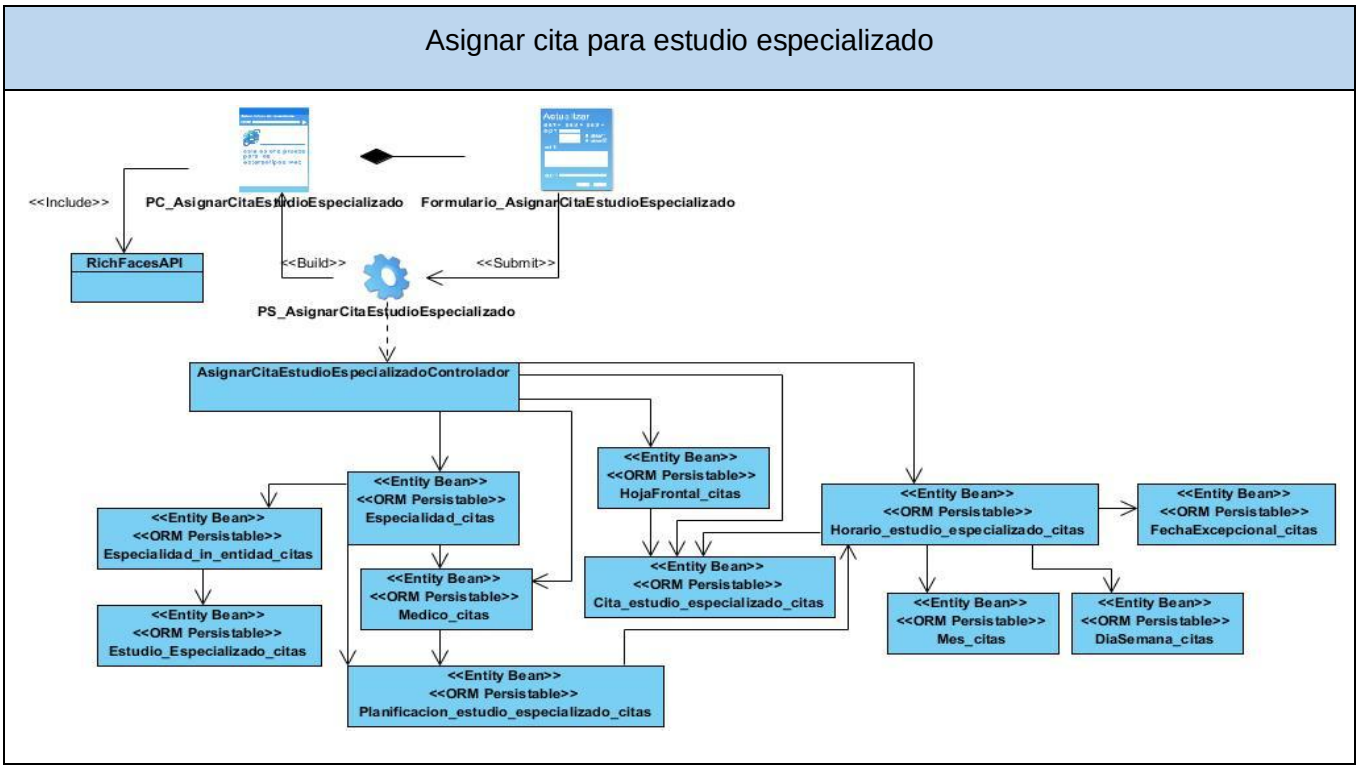


Figura 3.2 Asignar cita para estudio especializado.

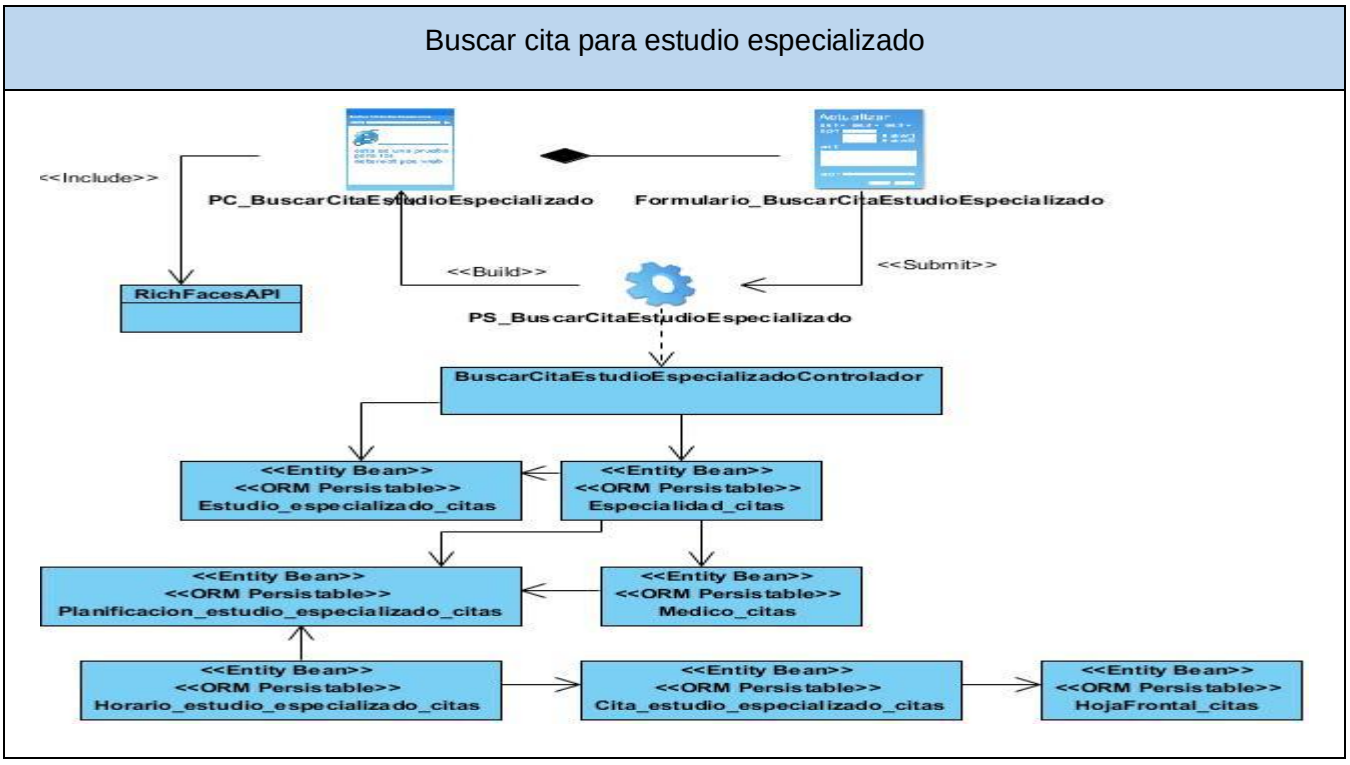


Figura 3.3 Buscar cita para estudio especializado.

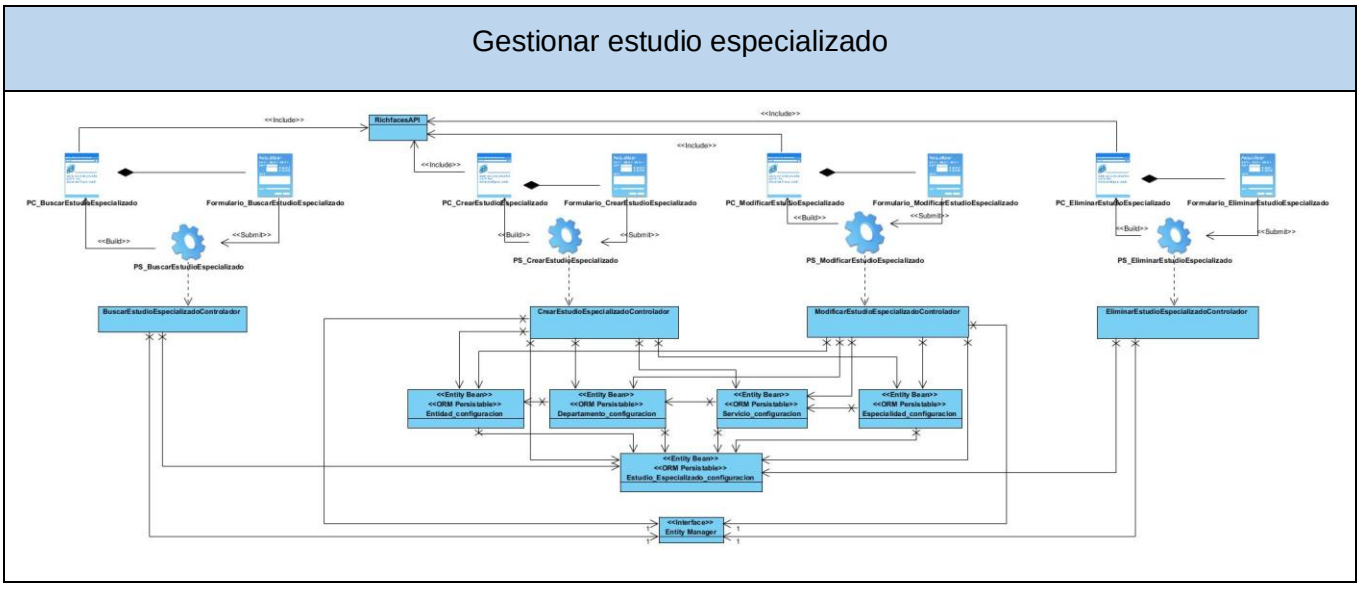


Figura 3.4 Gestionar estudio especializado.

Gestionar horario para estudio especializado

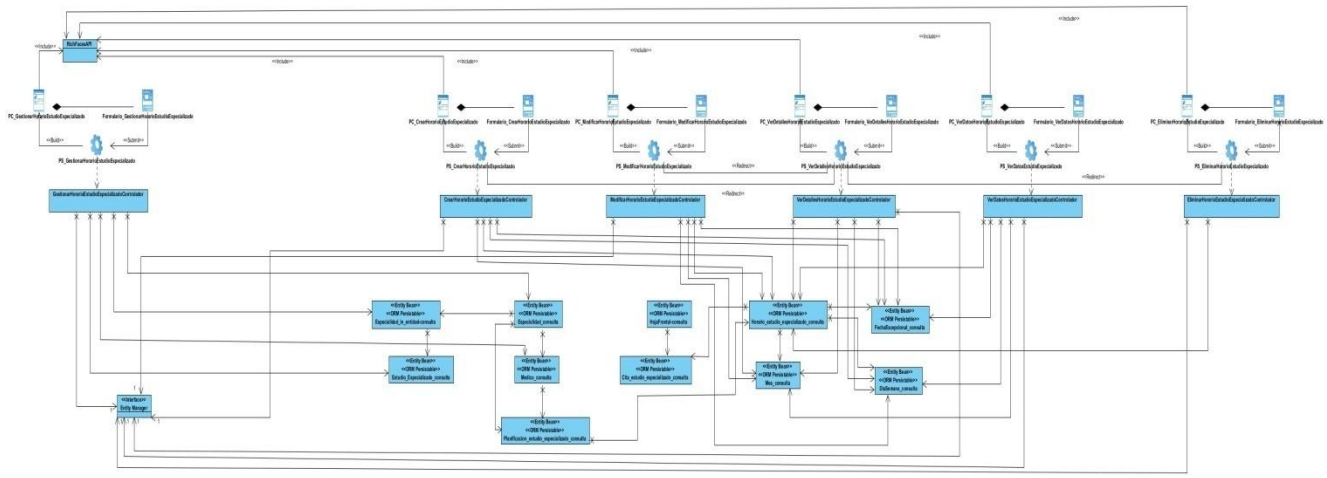


Figura 3.5 Gestionar horario para estudio especializado.

Seguidamente serán explicadas algunas de las clases que han sido identificadas para su futura implementación describiéndose las responsabilidades que realizarán las páginas servidoras que responden a la Lógica de Negocio. De esta manera se tendrá una comprensión mayor del funcionamiento que tendrá el sistema en desarrollo.

Nombre: AsignarCitaEstudioEspecializadoControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
calendarioMedico	CalendarioMedicoEstudioEspecializado
tarjetaCita	CrearTarjetaCitaEstudioEspecializadoControlador
cid	long
paciente	HojaFrontal_citas
diasSemana	List<DiaSemana_citas>
medico	Medico_citas
cupoExtra	boolean
titular	HojaFrontal_citas
flagTitular	boolean
soloTitulares	boolean
horariosCita	List<HorarioEstudioEspecializadoCustom>
citasPaciente	List<CitaEstudioEspecializado_citas>
cantTotalMinutos	int
cantTotalPacientes	int
horaInicio	Date

horaFin	Date
citasSet	Hashtable<String, List<CitaEstudioEspecializado_citas>>
fechaCuposExtras	Hashtable<String, Date>
citas	List<CitaEstudioEspecializado_citas>
especialidad	EspecialidadInEntidad_citas
estudio	EstudioInEntidad_citas
servicioCbx	String
especialidadCbx	String
medicoCbx	String
estudioCbx	String
confirmar	boolean
messgValidation	Hashtable<String, String>
puedeCrearTarjeta	boolean
Por cada responsabilidad	
Nombre:	boolean isConfirmar()
Descripción:	Devuelve la variable confirmar
Nombre:	void setConfirmar(boolean confirmar)
Descripción:	Cambia el valor de la variable confirmar por uno pasado por parámetro
Nombre:	List<CitaEstudioEspecializado_citas> getCitas()
Descripción:	Devuelve la variable citas
Nombre:	void cargarReglaCupoExtra()
Descripción:	Método para cargar la regla de negocio de los cupos extras
Nombre:	void cargarReglaCupoExtra(Usuario_citas usuario)
Descripción:	Método que carga la regla de negocio asociada a la asignación de los cupos extras
Nombre:	void selectPaciente(HojaFrontal_citas selpaciente)
Descripción:	Método para seleccionar al paciente para asignarle la cita
Nombre:	void disableConfirmar()
Descripción:	Método para desactivar la opción de confirmar
Nombre:	boolean validarReservar(CitaEstudioEspecializadoCustom horario)
Descripción:	Método para validar cada vez que se reserva una cita
Nombre:	boolean validarCitasMemoria()
Descripción:	Método para validar que las citas que estén en memoria no coincidan en el día antes y el después
Nombre:	boolean compararFechas(Calendar uno, Calendar dos)
Descripción:	Método para comprobar las fechas del horario del médico
Nombre:	String

	obtenerEspecialidadesCitas(CitaEstudioEspecializado_citas cita)
Descripción:	Método para obtener las especialidades a mostrar
Nombre:	String obtenerEstudiosCitas(CitaEstudioEspecializado_citas cita)
Descripción:	Método para obtener los estudio especializados a mostrar
Nombre:	void reservarCita(CitaEstudioEspecializadoCustom horario)
Descripción:	Método para reservar la cita
Nombre:	void cancelarCita(CitaEstudioEspecializadoCustom horario)
Descripción:	Método para cancelar la cita
Nombre:	void reservarCupoExtra(HorarioEstudioEspecializadoCustom h)
Descripción:	Método para reservar un cupo extra del horario del médico
Nombre:	boolean validarConfirmar()
Descripción:	Método para validar la opción de confirmar la cita
Nombre:	void confirmar()
Descripción:	Método para confirmar la reservación de la cita
Nombre:	void limpiarCampos()
Descripción:	Método para limpiar los campos y atributos
Nombre:	void cancelar()
Descripción:	Método para cancelar la cita
Nombre:	void llenarListaCitas(int cantPacientes, int cantRealPac, int cantMinutos, List<CitaEstudioEspecializado_citas> pacientesCitados, HorarioEstudioEspecializado_citas horarioCita, int cantTotalMinutos)
Descripción:	Método para llenar una lista de citas
Nombre:	HojaFrontal_citas getPaciente()
Descripción:	Devuelve el valor de la variable paciente
Nombre:	selectHorariosMedico()
Descripción:	Método para seleccionar el horario de un médico
Nombre:	void selectHorariosOnFechaSelect()
Descripción:	Método para seleccionar el horario de un médico a partir de una fecha seleccionada
Nombre:	Date getFechaSeleccionada()
Descripción:	Devuelve el valor de la variable fechaSeleccionada
Nombre:	void setFechaSeleccionada(Date fechaSeleccionada)
Descripción:	Cambia el valor de la variable fechaSeleccionada por uno pasado por parámetro

Nombre:	List<HorarioEstudioEspecializadoCustom> gethorariosCita()
Descripción:	Devuelve el valor de la variable horariosCita
Nombre:	void sethorariosCita(List<HorarioEstudioEspecializadoCustom> horariosCita)
Descripción:	Cambia el valor de la variable horariosCita por uno pasado por parámetro
Nombre:	void setPaciente(HojaFrontal_citas paciente)
Descripción:	Cambia el valor de la variable paciente por uno pasado por parámetro
Nombre:	Medico_citas getMedico()
Descripción:	Devuelve el valor de la variable médico
Nombre:	void setMedico(MedicoTree medico)
Descripción:	Cambia el valor de la variable médico por uno pasado por parámetro
Nombre:	boolean isPuedeCrearTarjeta()
Descripción:	Devuelve el valor de la variable puedeCrearTarjeta
Nombre:	void setPuedeCrearTarjeta(boolean puedeCrearTarjeta)
Descripción:	Cambia el valor de la variable puedeCrearTarjeta por uno pasado por parámetro
Nombre:	List<String> getServiciosMedico()
Descripción:	Método para mostrar los servicios en la vista
Nombre:	List<String> getEspecialidades()
Descripción:	Método para mostrar las especialidades en la vista
Nombre:	List<String> getEstudios()
Descripción:	Método para mostrar los estudios en la vista
Nombre:	List<String> getMedicos()
Descripción:	Método para mostrar los médicos en la vista
Nombre:	Servicio_citas getServicioActual()
Descripción:	Método para devolver el servicio actual que esta seleccionado en el combo box
Nombre:	EspecialidadInEntidad_citas getEspecialidadActual()
Descripción:	Método para devolver la especialidad actual que esta seleccionado en el combo box
Nombre:	EstudioInEntidad_citas getEstudioActual()
Descripción:	Método para devolver el estudio actual que esta seleccionado en el combo box
Nombre:	Medico_citas getMedicoActual()
Descripción:	Método para devolver el médico actual que esta seleccionado en el combo box
Nombre:	String getServicioCbx()
Descripción:	Devuelve la variable servicioCbx
Nombre:	void setServicioCbx(String servicioCbx)
Descripción:	Cambia el valor de la variable servicioCbx por uno

	pasado por parámetro
Nombre:	String getEspecialidadCbx()
Descripción:	Devuelve la variable especialidadCbx
Nombre:	void setEspecialidadCbx(String especialidadCbx)
Descripción:	Cambia el valor de la variable especialidadCbx por uno pasado por parámetro
Nombre:	String getEstudioCbx()
Descripción:	Devuelve la variable estudioCbx
Nombre:	void setEstudioCbx(String estudioCbx)
Descripción:	Cambia el valor de la variable estudioCbx por uno pasado por parámetro
Nombre:	String getMedicoCbx()
Descripción:	Devuelve la variable medicoCbx
Nombre:	void setMedicoCbx(String medicoCbx)
Descripción:	Cambia el valor de la variable medicoCbx por uno pasado por parámetro
Nombre:	boolean isCupoExtra()
Descripción:	Devuelve la variable cupoExtra
Nombre:	void setCupoExtra(boolean cupoExtra)
Descripción:	Cambia el valor de la variable cupoExtra por uno pasado por parámetro
Nombre:	HojaFrontal_citas getTitular()
Descripción:	Devuelve la variable titular
Nombre:	void setTitular(HojaFrontal_citas titular)
Descripción:	Cambia el valor de la variable titular por uno pasado por parámetro
Nombre:	boolean isFlagTitular()
Descripción:	Devuelve la variable flagTitular
Nombre:	void setFlagTitular(boolean flagTitular)
Descripción:	Cambia el valor de la variable flagTitular por uno pasado por parámetro
Nombre:	boolean isSoloTitulares()
Descripción:	Devuelve la variable soloTitulares
Nombre:	void setSoloTitulares(boolean soloTitulares)
Descripción:	Cambia el valor de la variable soloTitulares por uno pasado por parámetro

Tabla 3.1 Descripción de la clase AsignarCitaEstudioEspecializadoControlador.

Nombre: CrearEstudioEspecializadoControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**

nombre	String
cid	Long
Estudio	EstudioEspecializado
listaEntidadSource	List<Entidad_configuracion>
listaEntidadTarget	List<Entidad_configuracion>
listaDepartamentoSource	List<DepartamentoInEntidad_configuracion>
listaDepartamentoTarget	List<DepartamentoInEntidad_configuracion>
listaServicioInEntidadSource	List<ServicioInEntidad_configuracion>
listaServicioInEntidadTarget	List<ServicioInEntidad_configuracion>
listaEIE	List<EspecialidadInEntidad_configuracion>
listaSIE	<ServicioInEntidad_configuracion>
listaDIE	List<DepartamentoInEntidad_configuracion>
listaE	List<Entidad_configuracion>
tabSelect	String
Por cada responsabilidad	
Nombre:	void Source()
Descripción:	Inicia la clase controladora
Nombre:	void entidades()
Descripción:	Carga una lista de las entidades
Nombre:	void departamentos()
Descripción:	Carga una lista de los departamentos
Nombre:	void departamentoTargetValidation()
Descripción:	Valida la lista de los departamentos
Nombre:	void servicios()
Descripción:	Carga una lista de los servicios
Nombre:	void servicioTargetValidation()
Descripción:	Valida la lista de los servicios
Nombre:	void especialidades()
Descripción:	Carga una lista de las especialidades
Nombre:	void especialidadTargetValidation()
Descripción:	Valida la lista de las especialidades
Nombre:	void subirEspecialidades()
Descripción:	Lanza las especialidades
Nombre:	void entidadTargetValidation()
Descripción:	Valida la lista de las entidades
Nombre:	String crear()
Descripción:	Crear el estudio especializado
Nombre:	String getNombre()
Descripción:	Devuelve la variable nombre
Nombre:	setNombre(String nombre)
Descripción:	Cambia el valor de la variable nombre por uno pasado por parámetro
Nombre:	List<ServicioInEntidad_configuracion>

	getListaServicioInEntidadTarget()
Descripción:	Devuelve la variable listaServicioInEntidadTarget
Nombre:	void setListaServicioInEntidadTarget(List<ServicioInEntidad_configuracion> listaServicioInEntidadTarget)
Descripción:	Cambia el valor de la variable listaServicioInEntidadTarget por uno pasado por parámetro
Nombre:	List<ServicioInEntidad_configuracion> getListaServicioInEntidadSource()
Descripción:	Devuelve la variable listaServicioInEntidadSource
Nombre:	void setListaServicioInEntidadSource(List<ServicioInEntidad_configuracion> listaServicioInEntidadSource)
Descripción:	Cambia el valor de la variable listaServicioInEntidadSource por uno pasado por parámetro
Nombre:	Long getCid()
Descripción:	Devuelve la variable cid
Nombre:	void setCid(Long cid)
Descripción:	Cambia el valor de la variable cid por uno pasado por parámetro
Nombre:	List<Entidad_configuracion> getListaEntidadSource()
Descripción:	Devuelve la variable listaEntidadSource
Nombre:	void setListaEntidadSource(List<Entidad_configuracion> listaEntidadSource)
Descripción:	Cambia el valor de la variable listaEntidadSource por uno pasado por parámetro
Nombre:	List<Entidad_configuracion> getListaEntidadTarget()
Descripción:	Devuelve la variable listaEntidadTarget
Nombre:	void setListaEntidadTarget(List<Entidad_configuracion> listaEntidadTarget)
Descripción:	Cambia el valor de la variable listaEntidadTarget por uno pasado por parámetro
Nombre:	String getTabSelect()
Descripción:	Devuelve la variable tabSelect
Nombre:	void setTabSelect(String tabSelect)
Descripción:	Cambia el valor de la variable tabSelect por uno pasado por parámetro
Nombre:	List<DepartamentoInEntidad_configuracion> getListaDepartamentoSource()
Descripción:	Devuelve la variable listaDepartamentoSource
Nombre:	void setListaDepartamentoSource(List<DepartamentoInEntidad_configuracion>

	listaDepartamentoSource)
Descripción:	Cambia el valor de la variable listaDepartamentoSource por uno pasado por parámetro
Nombre:	List<DepartamentoInEntidad_configuracion> getListaDepartamentoTarget()
Descripción:	Devuelve la variable listaDepartamentoTarget
Nombre:	void setListaDepartamentoTarget(List<DepartamentoInEntidad_configuracion> listaDepartamentoTarget)
Descripción:	Cambia el valor de la variable listaDepartamentoTarget por uno pasado por parámetro
Nombre:	EstudioEspecializado_configuracion getEstudio()
Descripción:	Devuelve la variable estudio
Nombre:	void setEstudio(EstudioEspecializado_configuracion estudio)
Descripción:	Cambia el valor de la variable estudio por uno pasado por parámetro
Nombre:	List<EspecialidadInEntidad_configuracion> getListaEspecialidadInEntidadSource()
Descripción:	Devuelve la variable listaEspecialidadInEntidadSource
Nombre:	void setListaEspecialidadInEntidadSource(List<EspecialidadInEntidad_configuracion> listaEspecialidadInEntidadSource)
Descripción:	Cambia el valor de la variable listaEspecialidadInEntidadSource por uno pasado por parámetro
Nombre:	List<EspecialidadInEntidad_configuracion> getListaEspecialidadInEntidadTarget()
Descripción:	Devuelve la variable listaEspecialidadInEntidadTarget
Nombre:	void setListaEspecialidadInEntidadTarget(List<EspecialidadInEntidad_configuracion> listaEspecialidadInEntidadTarget)
Descripción:	Cambia el valor de la variable listaEspecialidadInEntidadTarget por uno pasado por parámetro

Tabla 3.2 Descripción de la clase CrearEstudioEspecializadoControlador.

Nombre: CrearHorarioCitaEstudioEspecializadoControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
planificacionDataModelEstudio_consulta	PlanificacionDataModelEstudio
cid	long

medico	Medico_consulta
especialidad	EspecialidadInEntidad_consulta
servicio	ServicioInEntidad_consulta
estudio	EstudioInEntidad_consulta
horaInicioM	String
horaInicioH	String
amHoraInicio	String
horaFinM	String
horaFinH	String
amHoraFin	String
fechaInicio	Date
fechaFin	Date
cantPacientes	String
fechaExcepcional	Date
meseSel	List<Mes_consulta>
meses	List<Mes_consulta>
dias	List<DiaSemana_consulta>
fechas	List<String>
fechasSel	List<String>
fechasExcepcionales	List<FechaExcepcional_consulta>
fechasExcepcionalesRecientes	List<FechaExcepcional_consulta>
puedeCrear	int
msgHashMeses	Hashtable<Integer, String>
msgHashDias	Hashtable<Integer, String>
msgMeses	String
msgDias	String
Por cada responsabilidad	
Nombre:	adicionar(EstudioInEntidad_consulta estudio,Medico_consulta medico,EspecialidadInEntidad_consulta especialidad,ServicioInEntidad_consulta servicio)
Descripción:	Método para adicionar un horario
Nombre:	String salvar()
Descripción:	Método para salvar el horario
Nombre:	void cancelar()
Descripción:	Método para cancelar el horario
Nombre:	boolean validar()
Descripción:	Método para validar el horario
Nombre:	boolean validarHoras()
Descripción:	Método para validar las horas del horario
Nombre:	boolean validarFechas()
Descripción:	Método para validar las fechas del horario
Nombre:	boolean validarHorarioMedico()

Descripción:	Método para validar el horario del médico
Nombre:	Boolean comprobarMeses(HorarioEstudioEspecializado_consulta h)
Descripción:	Método para comprobar los meses de horario pasado por parámetro
Nombre:	comprobarDiasSemana(List<DiaSemana_consulta> param,HorarioEstudioEspecializado_consulta h)
Descripción:	Método para comprobar los días de la semana de horario pasado por parámetro
Nombre:	boolean comprobarHoras(Date horai, Date horaf)
Descripción:	Método para comprobar las horas de horario
Nombre:	void actualizarFechasExcepcionales()
Descripción:	Método para actualizar las fechas excepcionales
Nombre:	void agregarFechaExcepcional()
Descripción:	Método para agregar una fecha excepcional
Nombre:	void eliminarFechaExcepcional(String f)
Descripción:	Método para eliminar una fecha excepcional
Nombre:	int buscarFecha(String fecha)
Descripción:	Método para buscar una fecha
Nombre:	void limpiarFechas()
Descripción:	Método para limpiar fechas
Nombre:	void eliminarFechaExcepcional()
Descripción:	Método para eliminar una fecha excepcional
Nombre:	void cargarFechasExcepcionales()
Descripción:	Método para cargar fechas excepcionales
Nombre:	void cargarMeses()
Descripción:	Método para cargar los meses
Nombre:	void cargaMesesFecha()
Descripción:	Método para cargar los meses de una fecha
Nombre:	String concatMes(int numeroMes)
Descripción:	Método para concatenar los meses con un número de mes pasado por parámetro
Nombre:	String concatDia(int numeroDia)
Descripción:	Método para concatenar los días con un número de días pasado por parámetro
Nombre:	HorarioEstudioEspecializado_consulta getHorario()
Descripción:	Devuelve la variable horario
Nombre:	void setHorario(HorarioEstudioEspecializado_consulta horario)
Descripción:	Cambia el valor de la variable horario por uno pasado por parámetro
Nombre:	Date getFechaExcepcional()
Descripción:	Devuelve la variable fechaExcepcional

Nombre:	void setFechaExcepcional(Date fechaExcepcional)
Descripción:	Cambia el valor de la variable fechaExcepcional por uno pasado por parámetro
Nombre:	List<Mes_consulta> getMeseSel()
Descripción:	Devuelve la variable meseSel
Nombre:	void setMeseSel(List<Mes_consulta> meses)
Descripción:	Cambia el valor de la variable meseSel por uno pasado por parámetro
Nombre:	List<DiaSemana_consulta> getDias()
Descripción:	Devuelve la variable dias
Nombre:	void setDias(List<DiaSemana_consulta> dias)
Descripción:	Cambia el valor de la variable dias por uno pasado por parámetro
Nombre:	List<String> getFechas()
Descripción:	Devuelve la variable fechas
Nombre:	void setFechas(List<String> fechas)
Descripción:	Cambia el valor de la variable fechas por uno pasado por parámetro
Nombre:	List<String> getFechasSel()
Descripción:	Devuelve la variable fechasSel
Nombre:	void setFechasSel(List<String> fechasSel)
Descripción:	Cambia el valor de la variable fechasSel por uno pasado por parámetro
Nombre:	String getAmHoraInicio()
Descripción:	Devuelve la variable amHoraInicio
Nombre:	void setAmHoraInicio(String amHoraInicio)
Descripción:	Cambia el valor de la variable amHoraInicio por uno pasado por parámetro
Nombre:	String getAmHoraFin()
Descripción:	Devuelve la variable amHoraFin
Nombre:	void setAmHoraFin(String amHoraFin)
Descripción:	Cambia el valor de la variable amHoraFin por uno pasado por parámetro
Nombre:	String getCantPacientes()
Descripción:	Devuelve la variable cantPacientes
Nombre:	void setCantPacientes(String cantPacientes)
Descripción:	Cambia el valor de la variable cantPacientes por uno pasado por parámetro
Nombre:	String getHoraInicioM()
Descripción:	Devuelve la variable horaInicioM
Nombre:	void setHoraInicioM(String horaInicioM)
Descripción:	Cambia el valor de la variable horaInicioM por uno pasado por parámetro
Nombre:	String getHoraInicioH()

Descripción:	Devuelve la variable horaInicioH
Nombre:	void setHoraInicioH(String horaInicioH)
Descripción:	Cambia el valor de la variable horaInicioH por uno pasado por parámetro
Nombre:	String getHoraFinM()
Descripción:	Devuelve la variable horaFinM
Nombre:	void setHoraFinM(String horaFinM)
Descripción:	Cambia el valor de la variable horaFinM por uno pasado por parámetro
Nombre:	String getHoraFinH()
Descripción:	Devuelve la variable horaFinH
Nombre:	void setHoraFinH(String horaFinH)
Descripción:	Cambia el valor de la variable horaFinH por uno pasado por parámetro
Nombre:	Date getFechaInicio()
Descripción:	Devuelve la variable fechaInicio
Nombre:	void setFechaInicio(Date fechaInicio)
Descripción:	Cambia el valor de la variable fechaInicio por uno pasado por parámetro
Nombre:	Date getFechaFin()
Descripción:	Devuelve la variable fechaFin
Nombre:	void setFechaFin(Date fechaFin)
Descripción:	Cambia el valor de la variable fechaFin por uno pasado por parámetro
Nombre:	List<Mes_consulta> getMeses()
Descripción:	Devuelve la variable meses
Nombre:	void setMeses(List<Mes_consulta> meses)
Descripción:	Cambia el valor de la variable meses por uno pasado por parámetro
Nombre:	int getPuedeCrear()
Descripción:	Devuelve la variable puedeCrear
Nombre:	void setPuedeCrear(int puedeCrear)
Descripción:	Cambia el valor de la variable puedeCrear por uno pasado por parámetro

Tabla 3.3 Descripción de la clase CrearHorarioCitaEstudioEspecializadoControlador.

Nombre: CrearSolicitudEstudioEspecializadoControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
hojaFrontal	HojaFrontal_consulta
idCita	int

cita	CitaEstudioEspecializado_consulta
error	int
estudioCbx	String
especialidad	Especialidad_consulta
cieConsList_custom	CieSolicitudConsList_custom
listaCie	List<Cie_consulta>
enfermedadesSeleccionadas	List<DiagnosticoMedicoEnfermedad_consulta>
enfSelected	Hashtable<Long,Cie_consulta>
diagnostico	DiagnosticoMedico_consulta
diagnosticoEnfermedades	Set<DiagnosticoMedicoEnfermedad_consulta>
idEnfSelect	Long
enfermedadesHoja	List<Cie_consulta>
usuario	Usuario_consulta
Por cada responsabilidad	
Nombre:	void inicializarSolicitud()
Descripción:	Método para iniciar la solicitud de estudio especializado
Nombre:	List<String> getEstudios()
Descripción:	Método para obtener los estudios especializados que no pertencen a la especialidad en atención
Nombre:	EstudioInEntidad_consulta getEstudioActual()
Descripción:	Método para obtener el estudio seleccionado en el combo box
Nombre:	void seleccionarEnfermedad()
Descripción:	Método para seleccionar una enfermedad
Nombre:	void eliminarEnfermedad(int pos)
Descripción:	Método para eliminar una enfermedad
Nombre:	List<DiagnosticoMedicoEnfermedad_consulta> getEnfermedadesSeleccionadas()
Descripción:	Devuelve la variable enfermedadesSeleccionadas
Nombre:	void setEnfermedadesSeleccionadas(List<DiagnosticoMedicoEnfermedad_consulta> enfermedadesSeleccionadas)
Descripción:	Cambia el valor de la variable enfermedadesSeleccionadas por uno pasado por parámetro
Nombre:	Hashtable<Long, Cie_consulta> getEnfSelected()
Descripción:	Devuelve la variable enfSelected
Nombre:	void setEnfSelected(Hashtable<Long, Cie_consulta> enfSelected)
Descripción:	Cambia el valor de la variable enfSelected por uno pasado por parámetro
Nombre:	Long getIdEnfSelect()
Descripción:	Devuelve la variable idEnfSelect
Nombre:	void setIdEnfSelect(Long idEnfSelect)

Descripción:	Cambia el valor de la variable idEnfSelect por uno pasado por parámetro
Nombre:	void cargarDiagnosticoPrevio()
Descripción:	Método para cargar un diagnóstico previo
Nombre:	List<Cie_consulta> getEnfermedadesHoja()
Descripción:	Devuelve la variable enfermedadesHoja
Nombre:	CieSolicitudConsList_custom getCieConsList_custom()
Descripción:	Devuelve la variable cieConsList_custom
Nombre:	Void setCieConsList_custom(CieSolicitudConsList_custom cieConsList_custom)
Descripción:	Cambia el valor de la variable cieConsList_custom por uno pasado por parámetro
Nombre:	List<String> getEntidades()
Descripción:	Método para devolver las entidades
Nombre:	List<String> getDepartamentos()
Descripción:	Método para devolver los departamentos
Nombre:	List<String> getServicios()
Descripción:	Método para devolver los servicios
Nombre:	boolean tieneEstudios()
Descripción:	Método para verificar si la especialidad en atención tiene estudios especializado
Nombre:	void salvar()
Descripción:	Método para salvar la soicitud de estudio
Nombre:	void cancelar()
Descripción:	Método para cancelar la solicitud de estudio
Nombre:	HojaFrontal_consulta getHojaFrontal()
Descripción:	Devuelve la variable hojaFrontal
Nombre:	void setHojaFrontal(HojaFrontal_consulta hojaFrontal)
Descripción:	Cambia el valor de la variable hojaFrontal por uno pasado por parámetro
Nombre:	int getError()
Descripción:	Devuelve la variable error

Nombre:	void setError(int error)
Descripción:	Cambia el valor de la variable error por uno pasado por parámetro
Nombre:	SolicitudEstudioEspecializado_consulta getSolicitud()
Descripción:	Devuelve la variable solicitud
Nombre:	void setSolicitud(SolicitudEstudioEspecializado_consulta solicitud)
Descripción:	Cambia el valor de la variable solicitud por uno pasado por parámetro
Nombre:	String getEstudioCbx()
Descripción:	Devuelve la variable estudioCbx
Nombre:	void setEstudioCbx(String estudioCbx)
Descripción:	Cambia el valor de la variable estudioCbx por uno pasado por parámetro

Tabla 3.4 Descripción de la clase CrearSolicitudEstudioEspecializadoControlador.

3.4. Modelo de datos.

El modelo de datos es la traducción del análisis de requisitos al esquema conceptual mediante una representación gráfica de las entidades y sus relaciones. Este, ayuda a entender y nombrar la información, evita la redundancia, asegura la corrección, validación y completitud de los datos y su organización refleja la política del negocio. Está compuesto por entidades, atributos y sus relaciones. Las entidades son objetos de los que el sistema necesita guardar información. Por ser un concepto o abstracción se suele emplear el sustantivo en singular para nombrar la entidad. Se simbolizan representando un rectángulo. Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. Son una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. Un modelo de datos se distingue de otro por el tratamiento que da a sus categorías, ya sea estática, dinámica o de integridad sobre las entidades y las operaciones. (32)

La descripción de los componentes del modelo de datos ayudará a entender mejor el diagrama que a continuación se presenta:

Modelo de datos

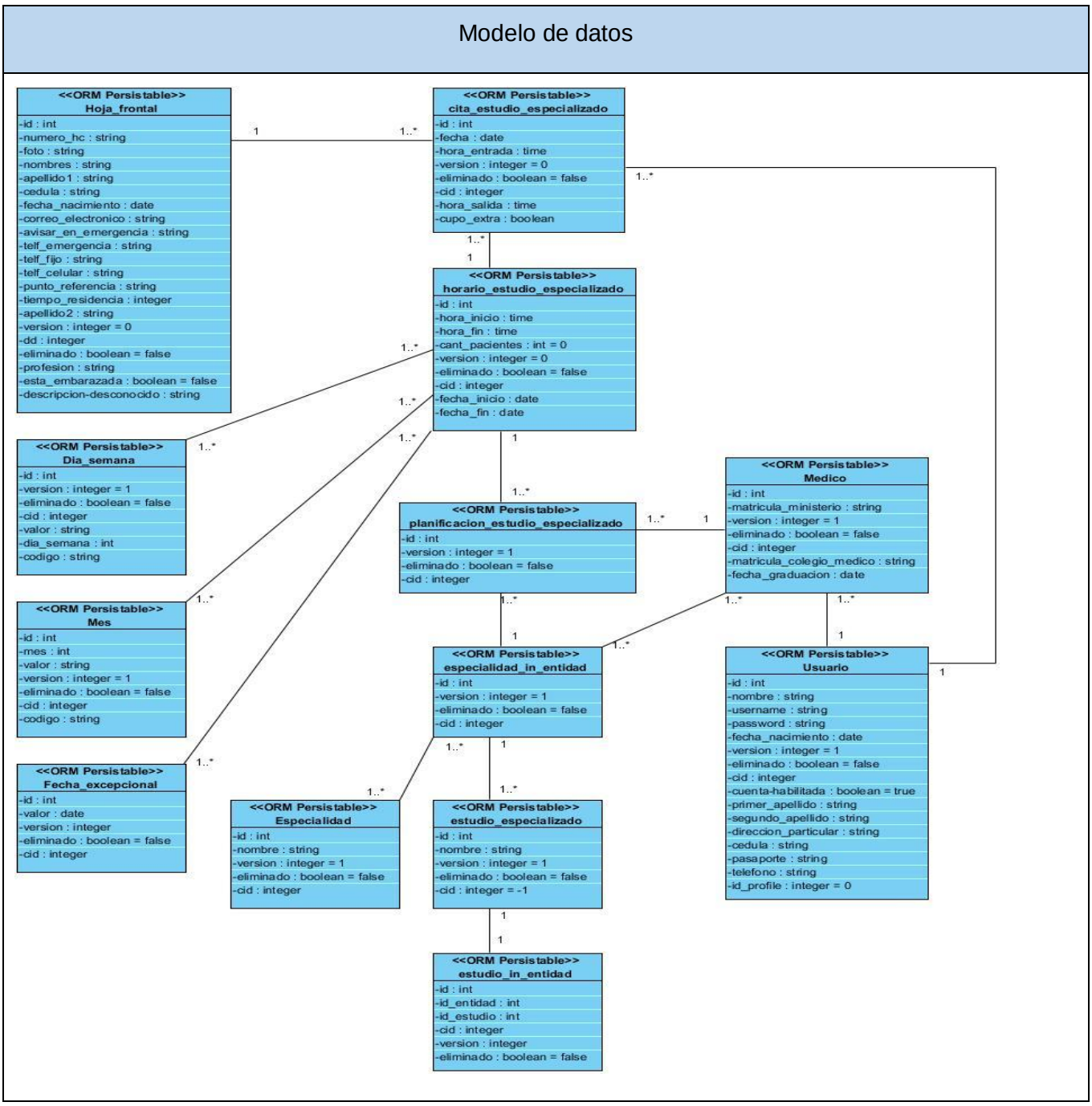


Figura 3.6 Modelo de datos.

3.4.1. Descripción de las tablas de la base de datos

Tabla Usuario:

Nombre: Usuario		
Descripción: Almacenar todos los usuarios del sistema		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del usuario
nombre	String	Nombre del usuario
username	String	Usuario
password	String	Contraseña
fecha_nacimiento	Date	Fecha de nacimiento
eliminado	Boolean	Si el usuario fue eliminado o no
cid	Integer	Para trabajar con la bitácora
cuenta_habilitada	Boolean	Si la cuenta fue habilitada o no
primer_apellido	String	Primer apellido
segundo_apellido	String	Segundo apellido
dirección_particular	String	Dirección particular
cedula	String	Número de identidad
pasaporte	String	Pasaporte
telefono	String	Teléfono

Tabla 3.5 Descripción de la tabla Usuario.

Tabla Médico:

Nombre: Medico		
Descripción: Almacenar los médicos que trabajan en la institución hospitalaria		
Atributo	**Tipo**	
id	int	Identificador del médico
matricula_ministerio	String	Matrícula que lo identifica en el ministerio
eliminado	Boolean	Si el médico fue eliminado o no
cid	Integer	Para trabajar con la bitácora
matricula_colegio_medico	String	Matrícula que lo identifica en el colegio
fecha_graduacion	Date	Fecha de graduación

Tabla 3.6 Descripción de la tabla Médico.

Tabla Planificación_estudio_especializado:

Nombre: Planificacion_estudio_especializado		
Descripción: Almacenar la planificación de citas para estudios especializados en el sistema		
Atributo	**Tipo**	**Descripción**
id	int	Identificador de la planificación
eliminado	Boolean	Si la planificación fue eliminada o no
cid	Integer	Para trabajar con la bitácora

Tabla 3.7 Descripción de la tabla Planificación_estudio_especializado.

Tabla Especialidad:

Nombre: Especialidad		
Descripción: Almacenar las especialidades que presentan las instituciones hospitalarias		
Atributo	**Tipo**	**Descripción**
id	int	Identificador de la especialidad
nombre	String	Nombre de la especialidad
eliminado	Boolean	Si la especialidad fue eliminada o no
cid	Integer	Para trabajar con la bitácora

Tabla 3.8 Descripción de la tabla Especialidad.

Tabla Cita:

Nombre: Cita		
Descripción: Almacenar las citas existentes que han sido planificadas a los médicos		
Atributo	**Tipo**	**Descripción**
id	int	Identificador de la cita
fecha	Date	Fecha en que se crea la cita
hora_entrada	Time	Hora de entrada de la cita
eliminado	Boolean	Si la cita está eliminada o no
cid	Integer	Para trabajar con la bitácora
hora_salida	Time	Hora de salida de la cita

Tabla 3.9 Descripción de la tabla Cita.

Tabla Horario_estudio_especializado:

Nombre: Horario_estudio_especializado		
Descripción: Almacenar los horarios de la planificación de citas para estudios especializados		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del horario
hora_inicio	Time	Hora de inicio del horario
hora_fin	Time	Hora de fin del horario
cant_pacientes	int	Cantidad de pacientes que almacena el horario
eliminado	Boolean	Si el horario fue eliminado o no
cid	Integer	Para trabajar con la bitácora
fecha_inicio	Date	Fecha de inicio del horario
fecha_fin	Date	Fecha fin del horario

Tabla 3.10 Descripción de la tabla Horario_estudio_especializado.

Tabla Hoja Frontal:

Nombre: Hoja_frontal		
Descripción: Almacenar los pacientes de las instituciones hospitalarias		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del paciente
numero_hc	String	Número de la historia clínica

foto	String	Foto del paciente
nombres	String	Nombre(s) del paciente
apellido1	String	Primer apellido del paciente
cedula	String	Número de identidad
fecha_nacimiento	Date	Fecha de nacimiento
correo_electronico	String	Correo electrónico del paciente
avisar_en_emergencia	String	Avisar en caso de emergencia
telf_emergencia	String	Teléfono en caso de emergencia
telf_fijo	String	Teléfono fijo
telf_celular	String	Teléfono celular
punto_referencia	String	Punto de referencia
tiempo_residencia	Integer	Tiempo en la residencia
apellido2	String	Segundo apellido del paciente
cid	Integer	Para trabajar con la bitácora
eliminado	Boolean	Si el paciente es eliminado o no
profesion	String	Profesión que ocupa el paciente
esta_embarazada	Boolean	Si el paciente está embarazada o no
descripcion_desconocido	String	Descripción en caso de ser desconocido

Tabla 3.11 Descripción de la tabla Hoja Frontal.

Tabla Estudio_especializado:

Nombre: Estudio_especializado		
Descripción: Almacenar los tipos de estudios especializados existentes en la entidad.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del tipo de estudio
nombre	String	Nombre del tipo de estudio especializado
eliminado	Boolean	Si el tipo de estudio especializado es eliminado o no
cid	Integer	Para trabajar con la bitácora
version	Integer	Para contabilizar las acciones realizadas sobre una tupla

Tabla 3.12 Descripción de la tabla Estudio_especializado.

Conclusiones parciales:

Como resultado del modelado de los flujos de diseño del sistema realizado en este capítulo, se logró identificar las clases fundamentales que deben ser definidas para el desarrollo de las funcionalidades relacionadas con la gestión de estudios especializados en el Sistema de Información Hospitalaria del CESIM, así como, los atributos y métodos que tienen las mismas para brindarle al desarrollador una idea clara de lo que se debe implementar. Además se obtuvo la realización de casos de uso por procesos donde se elaboraron los diagramas de clases de diseño.

CAPÍTULO 4. IMPLEMENTACIÓN DEL SISTEMA.

En el presente capítulo se introduce el flujo de trabajo de implementación. A partir del resultado del diseño se obtiene los diagramas de componente y despliegue; estos últimos, con la idea de mostrar las dependencias entre las partes del código y la estructura física del sistema en ejecución. Se describen las clases y subsistemas implementados en términos de componentes y se proporciona una detallada explicación de la solución implementada. Además se especifica la estrategia de codificación a seguir, así como, elementos de seguridad y captura de errores.

4.1. Modelo de Despliegue.

El Modelo de Despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene memoria y, a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. Representan un procesador o un dispositivo sobre el que se pueden desplegar los componentes. La relación entre un nodo y el componente que despliega puede mostrarse con una relación de dependencia. (33)

A continuación se presenta como queda estructurado el modelo de despliegue:

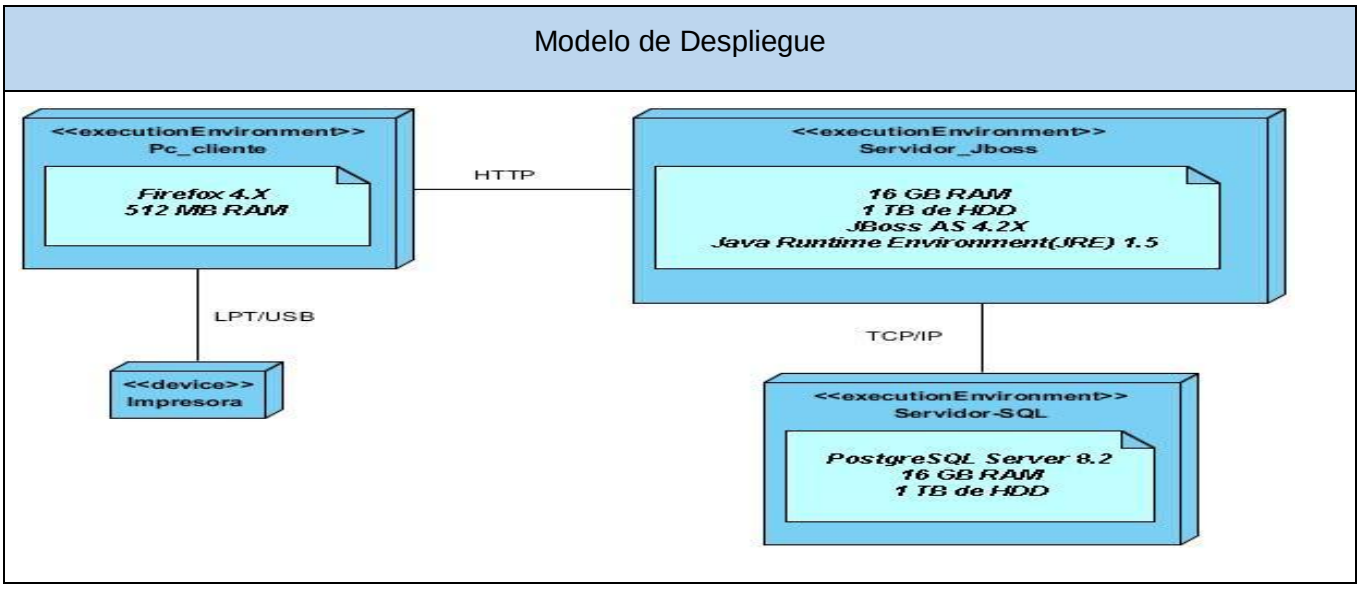


Figura 4.1 Modelo de Despliegue.

4.2. Diagrama de Componentes.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Estos muestran las opciones de realización como el código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. Estos tienen relaciones de traza con los elementos del modelo de diseño. De los estereotipos estándar que se aplican a los componentes según el lenguaje de modelado UML se emplean: library y file, los cuales representan, una biblioteca de objetos estática o dinámica y un documento que contiene código fuente, respectivamente. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente se refiere a los servicios ofrecidos por otro componente. (33)

A continuación se presenta como quedan estructurados estos componentes:

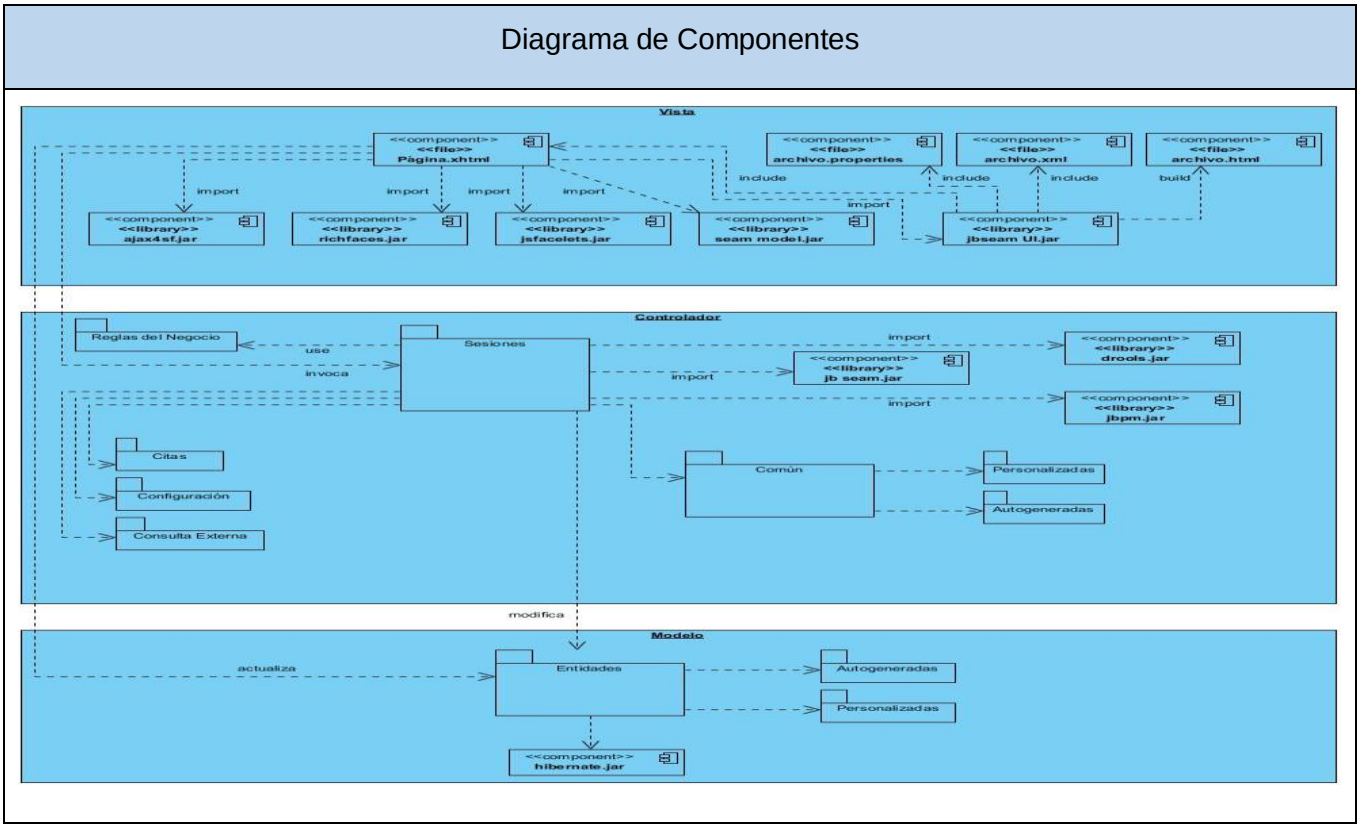


Figura 4.2 Diagrama de componentes.

4.3. Tratamiento de errores.

Durante el tiempo de ejecución de un sistema pueden fracasar diferentes rutinas; es esto a lo que comúnmente se le llama excepción. Mediante el tratamiento de excepciones se restaura un estado en el que la rutina pueda seguir la ejecución, lo que permite obtener un sistema más robusto y fiable. En el sistema propuesto, el control de las excepciones se lleva a cabo a toda porción de código, donde pueda surgir alguna situación inesperada, especialmente donde se ejecutan sentencias que manipulan los datos que viajan desde y hacia la base de datos. También se controlan los errores que pueden surgir en la validación de datos provenientes de la interfaz de usuario, puesto que encierran una lógica compleja en cierta medida. Para el manejo de las excepciones o errores, en las clases controladoras de procesos, se utilizará el bloque try para detectar cuando ocurra algún fallo y mediante el catch se manejarán dichas excepciones, mediante mensajes que se muestran en la interfaz de usuario, por las facilidades que brinda el FacesMessages, componente del framework Seam. Ejemplo:

```
try
{
 //declaración que causa la excepción
}
```

Entre las llaves de try se escribe el código que hará funcionar el programa. Para capturar la excepción que puede generar este código se necesita otra instrucción llamada catch (capturar).

```
catch(NombredeExcepcion obj){
 //código para tratar el error
}
```

Entre las llaves de catch se escribe el código que se quiera para tratar el error. Existe además un archivo XML, denominado page.xml, que engloba la configuración de todos los mensajes que se deben mostrar por cada tipo de excepción, así como la página a la que el sistema redirecciona en caso de la aparición de un error sorpresivo. (34)

4.4. Seguridad.

Para lograr un sistema seguro se sostendrá un control a nivel de usuarios y contraseñas, permitiendo el acceso por tipo de usuario logrando así la visibilidad sólo a las áreas establecidas de acorde a la función que realizan. Las contraseñas solo podrán ser cambiadas por el usuario o por el administrador del sistema. Otra cuestión es lograr la fiabilidad en las estaciones de trabajo, para lograr esto se define un segundo nivel de seguridad a nivel de estaciones de trabajo lo que posibilita la ejecución sólo de las aplicaciones que hayan sido definidas para la estación en cuestión. El sistema además permitirá llevar una traza de todas las operaciones llevadas a cabo por cada usuario mediante un registro de actividades por usuario en todo momento.

Para lograr la fidelidad de los datos, todo el intercambio entre el sistema y otros sistemas que soliciten información desde cualquier hospital, se realizará de forma cifrada eliminando posibilidades de acceso o modificación de la misma.

4.5. Estrategias de codificación. Estándares y estilos a utilizar.

Los estándares de codificación son reglas que se aplican para logar uniformidad en el código producido por un grupo de desarrollo de un sistema. Estos reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Los estándares de codificación no destapan problemas existentes, evitan más bien que los errores ocurran, lo que permite obtener un código de alta calidad. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación, con vistas a generar un código de alta calidad, es de gran importancia para la calidad del software. La aplicación de estándares de codificación además posibilita que el software que se obtiene sea fácil de comprender y de mantener en el tiempo. (35)

Para definir una robusta estructura y organización del código, se deben definir algunos estándares para su posterior entendimiento y cumplir con las buenas prácticas establecidas en la Ingeniería de Software. A continuación se resumen algunas de las convenciones tomadas en relación a estos aspectos.

Idioma: Se debe utilizar como idioma el español, las palabras no se acentuarán.

Identación: Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento. Se recomienda dejar dos espacios en blanco desde la instrucción anterior

para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.

Comentarios, separadores, líneas, espacios en blanco y márgenes: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez. Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.

Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función. Se recomienda usar espacios en blanco entre los operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nomproducto.

Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción.

No se debe usar espacio en blanco:

- Después del corchete abierto y antes del cerrado de un arreglo.
- Después del paréntesis abierto y antes del cerrado.
- Antes de un punto y coma.

Variables y constantes: El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamellCasing. El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Ejemplo: apellidoPaciente

Clases y Objetos: El objetivo fundamental es nombrar las clases e instancias de forma estándar para todas las aplicaciones. Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Ejemplo: MiClase(). Para el caso de las instancias se comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.

Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación PascalCasing. Ejemplo: function BuscarUnidad(). Si son funciones

que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set. El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

Conclusiones parciales:

Los autores del trabajo concluyen que los resultados obtenidos durante la etapa de implementación de las funcionalidades relacionadas con la gestión de estudios especializados fueron satisfactorios. En el capítulo, se detallaron los diagramas de despliegue y de componentes lo que posibilitó una vez concluido el proceso de implementación del sistema que se contara con un producto que satisface las necesidades identificadas con anterioridad y totalmente funcional. El mismo se encuentra correctamente documentado y cumple los requerimientos de seguridad necesarios para garantizar la confidencialidad y privacidad de la información que se manipula en todo el sistema. En el proceso de codificación se cumplió con los estándares definidos, lo que permitió obtener un sistema fácil de mantener en el transcurso del tiempo.

CONCLUSIONES GENERALES

La detallada descripción de los procesos relacionados con los estudios especializados en las instituciones hospitalarias ha permitido obtener una mejor comprensión del problema e identificar las principales necesidades a resolver. La posterior definición de los requerimientos ha sido punto de partida en el proceso de desarrollo del sistema propuesto para lograr que el mismo cumpla con las funcionalidades que la aplicación debe brindar.

El análisis de los sistemas relacionados con el campo de acción evidenció que los mismos no cumplen con todos los requisitos funcionales necesarios ni permiten la integración con el Sistema de Información Hospitalaria del CESIM.

Para el desarrollo de funcionalidades se asimiló la arquitectura propuesta por el departamento de Sistemas de Gestión Hospitalaria. El diseño propuesto y las tecnologías empleadas se basaron en dicha arquitectura. Se utilizaron patrones de arquitectura que permiten el desarrollo independiente de las capas. La solución web aporta las ventajas de ser multiplataforma, multiusuario y las facilidades de despliegue y mantenimiento.

Como resultado de las etapas de diseño e implementación desarrolladas, se ha concebido un sistema que cumple con los requisitos propuestos definidos, basado en la gestión de base de datos utilizando tecnologías Web según el modelo cliente/servidor de tres capas, garantizado la seguridad e integridad del sistema. La implementación se basó en tecnologías de desarrollo disponibles sin costo y que aseguran el cumplimiento de los requerimientos.

Con el desarrollo del presente trabajo se alcanzó, satisfactoriamente, el objetivo propuesto: desarrollar las funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del CESIM, que permita su correcta utilización por parte de los especialistas de las instituciones hospitalarias, reafirmando una vez más la importancia de emplear las tecnologías informáticas con el fin de apoyar el sistema de salud por los beneficios que pueden aportar a este sector.

RECOMENDACIONES

Se recomienda:

- Crear desde el módulo Consulta Externa una hoja específicamente para los estudios especializados, donde se pueda registrar el informe de resultado para cada estudio especializado indicado al paciente.
- Permitir desde todas las hojas de consulta, la búsqueda de los informes de resultados de estudios especializados registrados a los pacientes.

REFERENCIAS BIBLIOGRÁFICAS

1. **ALBET.** *IH-SW-DE-024 ALAS-HIS_Consulta Externa y Citas_Glosario de términos.* La Habana : s.n., 2008.

2. **MARTÍNEZ DÍAZ, YOSMEL y TARAFA GUZMÁN, ALEJANDRO.** *Módulo Citas del Sistema de Información Hospitalaria alas HIS. Universidad de las Ciencias Informáticas.* La Habana : s.n., 2009.

3. **García Linares, A.J., Reche Martínez, D. y Richarte Reina, J.M.** Conganat. *x-HIS: Un nuevo concepto en Sistemas de Información Hospitalarios.* [En línea] 2003. [Citado el: 20 de Enero de 2014.] http://www.conganat.org/SEIS/inforsalud03/INFORSALUD2003_garciaa1.pdf.

4. *GALENHOS: Sistema Integrado de Gestión.* [En línea] 2009. [Citado el: 20 de Enero de 2014.] http://www.saludayacucho.gob.pe/web/galenhos-introduccion.

5. *Sigho, Sistema de Información para la Gerencia Hospitalaria.* [En línea] 2009. [Citado el: 10 de Junio de 2014.] http://sigho.ses-gro.gob.mx/.

6. *Sistema de Información de Salud alasHIS.* [En línea] 2010. [Citado el: 20 de Enero de 2014.] http://www.scmsi.es/scmsi/images/pdf/alas-his.pdf.

7. **Geeks, Juan Calle.** [En línea] 20 de Febrero de 2011. [Citado el: 20 de Enero de 2014.] http://www.slideshare.net/Decimo/arquitectura-3-capas.

8. *2.14 MVC.pdf.* [En línea] 10 de Febrero de 2012. [Citado el: 22 de Enero de 2014.] https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf..

9. *Cliente-Servidor.* [En línea] 10 de Febrero de 2011. [Citado el: 24 de Febrero de 2014.] http://es.kioskea.net/contents/148-entorno-cliente-servidor.

10. **García, Alejandro Pérez.** *desarrolloweb.com.* [En línea] 21 de Febrero de 2009. [Citado el: 24 de Febrero de 2014.] http://www.desarrolloweb.com/articulos/2380.php..

11. *JBoss RichFaces.* [En línea] 2009. [Citado el: 25 de Febrero de 2014.] http://labs.jboss.com/portal/jbossrichfaces.

12. **Community, JBoss.** *JBoss Ajax4jsf. Introducción.* [En línea] 2007. [Citado el: 25 de Febrero de 2014.] http://www.jboss.org/jbossajax4jsf/docs/devguide/en/html/Introduction.html.

13. *Open Source University Meetup.* [En línea] 28 de Enero de 2010. [Citado el: 26 de Febrero de 2014.] http://www.slideshare.net/josehpxxx/lenguaje-de-programacin-java-12493687.

14. *Seam - Contextual Components. Introduction to JBoss Seam.* [En línea] 2008. [Citado el: 26 de Febrero de 2014.] http://www.seamframework.org/Documentation..

15. **Dan, Allen.** *Seam in action. Manning Early Access Program.* s.l. : Manning Publications Co, 2008.

16. **ALBET.** *Documento de arquitectura de Software alas HIS.* La Habana : s.n., 2008.

17. *Natura Software.* [En línea] 2007. [Citado el: 27 de Febrero de 2014.] http://www.naturasoftware.com/main.php?f=tecnologia.

18. *Introducción a Enterprise Java Beans.* [En línea] 2010. [Citado el: 27 de Febrero de 2014.] http://www.fdi.ucm.es/profesor/jpavon/web/45-ejb.pdf.

19. **Lasterra, Enrique Rodriguez.** slideshare. *Introducción a Java Persistence API - Presentation Transcript.* [En línea] 2008. [Citado el: 27 de Febrero de 2014.] http://www.slideshare.net/jamslug/introduccin-a-java-persistence-api..

20. **Bauer Christian, King Gavin.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co, 2007.

21. **García, Joaquín.** *UML: Casos de Uso. Desarrollo de software orientado a objetos.* [En línea] Septiembre de 2008. [Citado el: 6 de Marzo de 2014.] http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.

22. **Girón, Frans.** *slideshare.* [En línea] [Citado el: 7 de Marzo de 2014.] http://www.slideshare.net/FransGirn/visual-paradign-24825147.

23. *Milestone consulting.* [En línea] 2010. [Citado el: 7 de Marzo de 2014.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.

24. **Álvarez, Jorge Cortés.** slideshare. *Metodología RUP.* [En línea] 2008. [Citado el: 12 de Marzo de 2014.] http://www.slideshare.net/cortesalvarez/metodologa-rup..

25. **Media, O'Reilly.** Inc. XML.com. [En línea] 2009. [Citado el: 15 de Marzo de 2014.] http://www.xml.com.

26. **autores, Colectivo de.** *El Lenguaje de Etiquetado Hipertextual Extensible .* [En línea] [Citado el: 25 de Marzo de 2014.] http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm.

27. **Chaves, Michael Arias.** *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* [En línea] 6 de Julio de 2008. [Citado el: 20 de Marzo de 2014.] http://www.intersedes.ucr.ac.cr/10-art_11.html.

28. *Especificaciones De Requerimientos.* [En línea] [Citado el: 25 de Marzo de 2014.] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

29. **Fernández Vilas, Ana.** *Comportamiento del sistema.* [En línea] 20 de Marzo de 2011. [Citado el: 4 de Abril de 2014.] http://tvdi.det.uvigo.es/~avilas/UML/node24.html.

30. **Cesar de la Torre Llorente, y otros.** *Guía de Arquitectura N-Capas orientada al dominio.* 2010.

31. *Software y Aplicaciones Web. UML Diagramas.* [En línea] 2008. [Citado el: 7 de Abril de 2014.] http://www.jtentor.com.ar/post/UML-Diagramas.aspx.

32. **Marqués, María Mercedes.** *Modelo de datos.* [En línea] 2005-2006. [Citado el: 12 de Abril de 2014.] http://www3.uji.es/~mmarques/f47/apun/node32.html..

33. *Modelo de Implementación:Diagramas de Componentes y Despliegue.* [En línea] 15 de Marzo de 2013. [Citado el: 16 de Abril de 2014.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

34. WEBTutoriales. *Tratamiento de errores en Java con try y catch.* [En línea] 24 de Noviembre de 2007. [Citado el: 20 de Abril de 2014.] http://www.webtutoriales.com/tutoriales/programacion/java/try-and-catch.37.html.

35. MSDN. *Revisiones de código y estándares de codificación.* [En línea] 23 de Marzo de 2009. [Citado el: 26 de Abril de 2014.] http://msdn.microsoft.com/es-es/library/aa291591(VS.71).aspx.

36. **Hernández, Rolando Alfredo y Coello, Zayda.** *El paradigma cuantitativo de la investigación científica.* s.l. : Editorial Universitaria, 2002.

BIBLIOGRAFÍA

1. **ALBET.** *IH-SW-DE-024 ALAS-HIS_Consulta Externa y Citas_Glosario de términos.* La Habana : s.n., 2008.

2. **MARTÍNEZ DÍAZ, YOSMEL y TARAFA GUZMÁN, ALEJANDRO.** *Módulo Citas del Sistema de Información Hospitalaria alas HIS. Universidad de las Ciencias Informáticas.* La Habana : s.n., 2009.

3. **García Linares, A.J., Reche Martínez, D. y Richarte Reina, J.M.** Conganat. *x-HIS: Un nuevo concepto en Sistemas de Información Hospitalarios.* [En línea] 2003. [Citado el: 20 de Enero de 2014.] http://www.conganat.org/SEIS/inforsalud03/INFORSALUD2003_garciaa1.pdf.

4. *GALENHOS: Sistema Integrado de Gestión.* [En línea] 2009. [Citado el: 20 de Enero de 2014.] http://www.saludayacucho.gob.pe/web/galenhos-introduccion.

5. *Sigho, Sistema de Información para la Gerencia Hospitalaria.* [En línea] 2009. [Citado el: 10 de Junio de 2014.] http://sigho.ses-gro.gob.mx/.

6. *Sistema de Información de Salud alasHIS.* [En línea] 2010. [Citado el: 20 de Enero de 2014.] http://www.scmsi.es/scmsi/images/pdf/alas-his.pdf.

7. **Geeks, Juan Calle.** [En línea] 20 de Febrero de 2011. [Citado el: 20 de Enero de 2014.] http://www.slideshare.net/Decimo/arquitectura-3-capas.

8. *2.14 MVC.pdf.* [En línea] 10 de Febrero de 2012. [Citado el: 22 de Enero de 2014.] https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf..

9. *Cliente-Servidor.* [En línea] 10 de Febrero de 2011. [Citado el: 24 de Febrero de 2014.] http://es.kioskea.net/contents/148-entorno-cliente-servidor.

10. **García, Alejandro Pérez.** *desarrolloweb.com.* [En línea] 21 de Febrero de 2009. [Citado el: 24 de Febrero de 2014.] http://www.desarrolloweb.com/articulos/2380.php..

11. *JBoss RichFaces.* [En línea] 2009. [Citado el: 25 de Febrero de 2014.] http://labs.jboss.com/portal/jbossrichfaces.

12. **Community, JBoss.** *JBoss Ajax4jsf. Introducción.* [En línea] 2007. [Citado el: 25 de Febrero de 2014.] http://www.jboss.org/jbossajax4jsf/docs/devguide/en/html/Introduction.html.

13. *Open Source University Meetup.* [En línea] 28 de Enero de 2010. [Citado el: 26 de Febrero de 2014.] http://www.slideshare.net/josehpxxx/lenguaje-de-programacin-java-12493687.

14. *Seam - Contextual Components. Introduction to JBoss Seam.* [En línea] 2008. [Citado el: 26 de Febrero de 2014.] http://www.seamframework.org/Documentation..

15. **Dan, Allen.** *Seam in action. Manning Early Access Program.* s.l. : Manning Publications Co, 2008.

16. **ALBET.** *Documento de arquitectura de Software alas HIS.* La Habana : s.n., 2008.

17. *Natura Software.* [En línea] 2007. [Citado el: 27 de Febrero de 2014.] http://www.naturasoftware.com/main.php?f=tecnologia.

18. *Introducción a Enterprise Java Beans.* [En línea] 2010. [Citado el: 27 de Febrero de 2014.] http://www.fdi.ucm.es/profesor/jpavon/web/45-ejb.pdf.

19. **Lasterra, Enrique Rodriguez.** slideshare. *Introducción a Java Persistence API - Presentation Transcript.* [En línea] 2008. [Citado el: 27 de Febrero de 2014.] http://www.slideshare.net/jamslug/introduccin-a-java-persistence-api..

20. **Bauer Christian, King Gavin.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co, 2007.

21. **García, Joaquín.** *UML: Casos de Uso. Desarrollo de software orientado a objetos.* [En línea] Septiembre de 2008. [Citado el: 6 de Marzo de 2014.] http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.

22. **Girón, Frans.** *slideshare.* [En línea] [Citado el: 7 de Marzo de 2014.] http://www.slideshare.net/FransGirn/visual-paradign-24825147.

23. *Milestone consulting.* [En línea] 2010. [Citado el: 7 de Marzo de 2014.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.

24. **Álvarez, Jorge Cortés.** slideshare. *Metodología RUP.* [En línea] 2008. [Citado el: 12 de Marzo de 2014.] http://www.slideshare.net/cortesalvarez/metodologa-rup..

25. **Media, O'Reilly.** Inc. XML.com. [En línea] 2009. [Citado el: 15 de Marzo de 2014.] http://www.xml.com.

26. **autores, Colectivo de.** *El Lenguaje de Etiquetado Hipertextual Extensible .* [En línea] [Citado el: 25 de Marzo de 2014.] http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm.

27. **Chaves, Michael Arias.** *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* [En línea] 6 de Julio de 2008. [Citado el: 20 de Marzo de 2014.] http://www.intersedes.ucr.ac.cr/10-art_11.html.

28. *Especificaciones De Requerimientos.* [En línea] [Citado el: 25 de Marzo de 2014.] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

29. **Fernández Vilas, Ana.** *Comportamiento del sistema.* [En línea] 20 de Marzo de 2011. [Citado el: 4 de Abril de 2014.] http://tvdi.det.uvigo.es/~avilas/UML/node24.html.

30. **Cesar de la Torre Llorente, y otros.** *Guía de Arquitectura N-Capas orientada al dominio.* 2010.

31. *Software y Aplicaciones Web. UML Diagramas.* [En línea] 2008. [Citado el: 7 de Abril de 2014.] http://www.jtentor.com.ar/post/UML-Diagramas.aspx.

32. **Marqués, María Mercedes.** *Modelo de datos.* [En línea] 2005-2006. [Citado el: 12 de Abril de 2014.] http://www3.uji.es/~mmarques/f47/apun/node32.html..

33. *Modelo de Implementación:Diagramas de Componentes y Despliegue.* [En línea] 15 de Marzo de 2013. [Citado el: 16 de Abril de 2014.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

34. WEBTutoriales. *Tratamiento de errores en Java con try y catch.* [En línea] 24 de Noviembre de 2007. [Citado el: 20 de Abril de 2014.] http://www.webtutoriales.com/tutoriales/programacion/java/try-and-catch.37.html.

35. MSDN. *Revisiones de código y estándares de codificación.* [En línea] 23 de Marzo de 2009. [Citado el: 26 de Abril de 2014.] http://msdn.microsoft.com/es-es/library/aa291591(VS.71).aspx.

36. **Hernández, Rolando Alfredo y Coello, Zayda.** *El paradigma cuantitativo de la investigación científica.* s.l. : Editorial Universitaria, 2002.

37. **Valencia, Maria Eugenia.** *Diagrama de Clases de Diseño. Escuela de Ingeniería de Sistemas y Computación.* [En línea] 2009. http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/DISCLASES_A12.pdf.

38. **Lovelle, Juan Manuel Cueva.** *Introducción a UML. Universidad de Oviedo.* 2003-2004.

39. El Mundo Informático. *PatronesGgraps. Patrones de software para la asignación General de Responsabilidad.* [En línea] 08 de Febrero de 2010. [Citado el: 10 de Junio de 2014.] http://jorgesaavedra.wordpress.com/category/patrones-grasp.

40. *Medicina, D. R. Facultad de. Sistema de Información Hospitalaria. Universidad Nacional Autónoma de México.* [En línea] 2008. http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.

41. *Jboss Enterprise Middleware.* [En línea] 2007. http://www.latam.redhat.com/pdf/jboss/JBoss_enterprise_300507_esp.pdf.

42. Hispano. *Asociación java. Java Hispano.* [En línea] 2002-2007. http://www.javahispano.org..

43. *Hat, Red. Richfaces Developer Guide.* [En línea] 2008-2009. http://www.jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html_single/index.html.

ANEXOS

Anexo 1. Guía de observación de la entrevista. Aspectos a tener en cuenta.

1. Correcta administración y seguridad de los procesos relacionados con los estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica con un nivel de detalle acorde con las necesidades existentes.
2. Los módulos Configuración y Citas del Sistema de Información Hospitalaria del Centro de Informática Médica no cuentan con una configuración detallada y precisa que permita una mejor gestión de los estudios especializados.
3. El módulo Consulta Externa del Sistema de Información Hospitalaria del Centro de Informática Médica no cuenta con una configuración detallada y precisa que permita una mejor gestión de los horarios a los especialistas para la realización de estudios especializados.
4. El módulo Consulta Externa del Sistema de Información Hospitalaria del Centro de Informática Médica no cuenta con una configuración detallada y precisa que permita una mejor gestión de las solicitudes para estudios especializados.
5. Para la navegación del usuario el Sistema de Información Hospitalaria del Centro de Informática Médica no cuenta con funcionalidades que permitan una mejor gestión de los procesos relacionados con los estudios especializados.
6. La gestión de los procesos relacionados con los estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica se realiza dependiendo de los módulos que administran esta información en el sistema.

Anexo 2. Guía con aspectos a considerar en las entrevistas.

Objetivo: Obtener criterios acerca de las funcionalidades correspondientes a la gestión de los procesos de estudios especializados en el Sistema de Información Hospitalaria del Centro de Informática Médica.

1. Características de los módulos Configuración, Citas y Consulta Externa que deben prevalecer en las nuevas funcionalidades a desarrollar.
2. Características que no se encuentran en las funcionalidades asociadas a la gestión de los procesos relacionados con los estudios especializados que actualmente existen en el Sistema de Información Hospitalaria del CESIM y que son necesarias agregarlas para el desarrollo de las nuevas funcionalidades.
3. Nuevos aspectos que deba manejar el sistema en los módulos relacionados con la gestión de estudios especializados.
4. Posible uso de las mismas tecnologías y herramientas usadas para desarrollar el Sistema de Información Hospitalaria del CESIM para gestionar las nuevas funcionalidades a agregar al sistema existente.
5. Deseo de mantener la misma gestión de horarios de los especialistas encargados de brindar la atención al paciente, para las citas relacionadas con los estudios especializados.
6. Incluir disponibilidad de equipos para la realización de estudios especializados, así como toda la gestión asociada a la reservación o manipulación de estos para realizar un estudio determinado.

Anexo 3. Proceso: Gestionar estudios especializados.

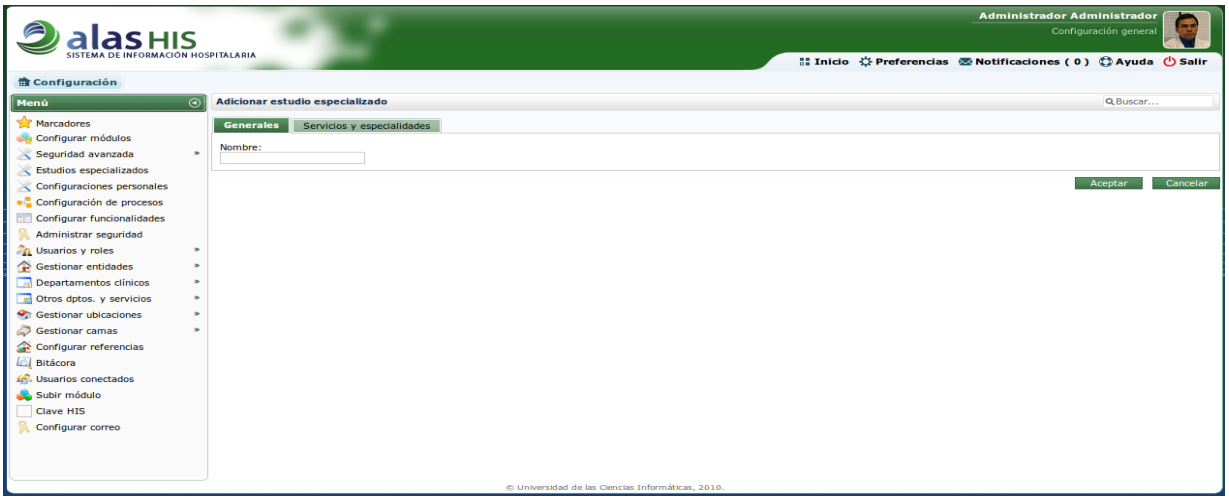


Imagen 1 Adicionar estudio especializado en el módulo Configuración.

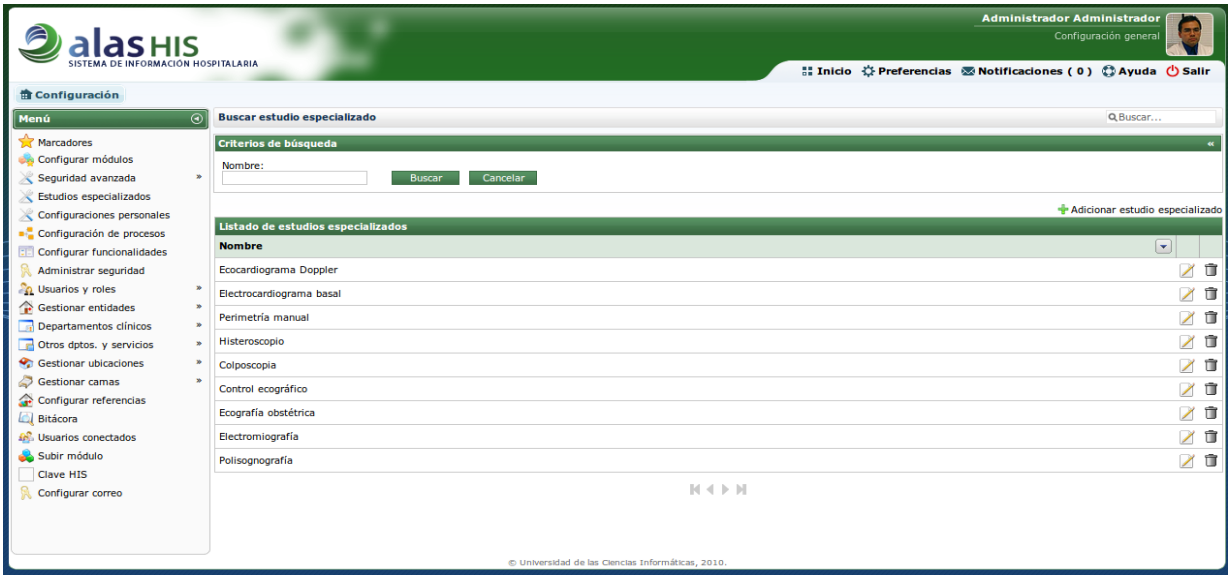


Imagen 2 Buscar estudio especializado en el módulo de Configuración.

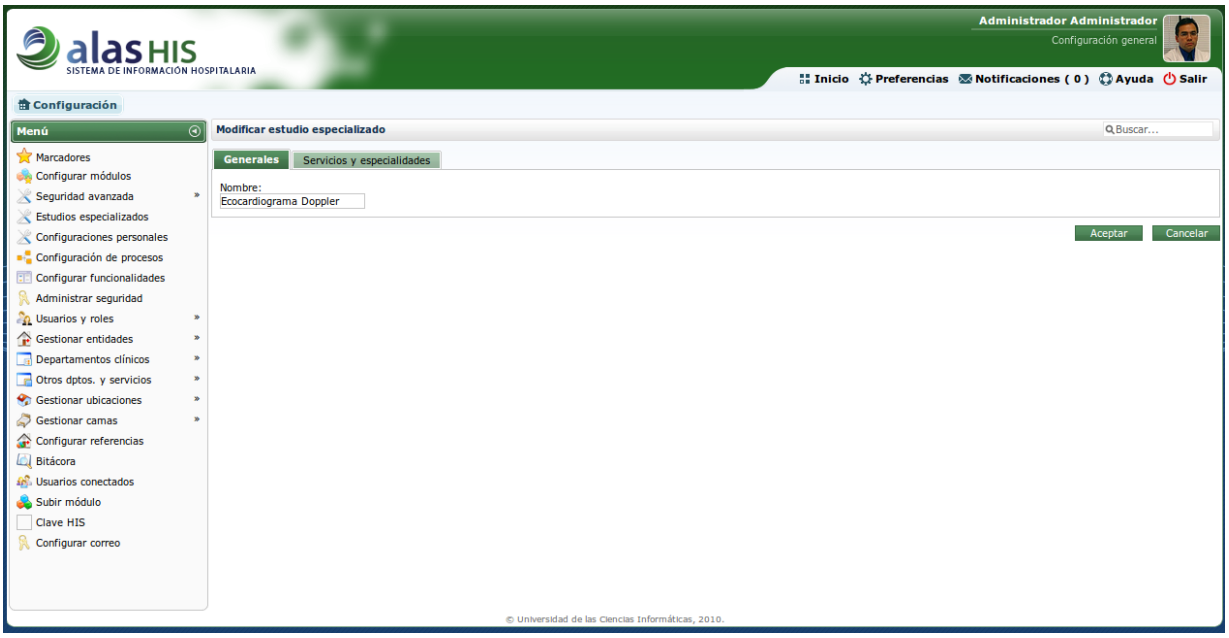


Imagen 3 Modificar estudio especializado en el módulo Configuración.

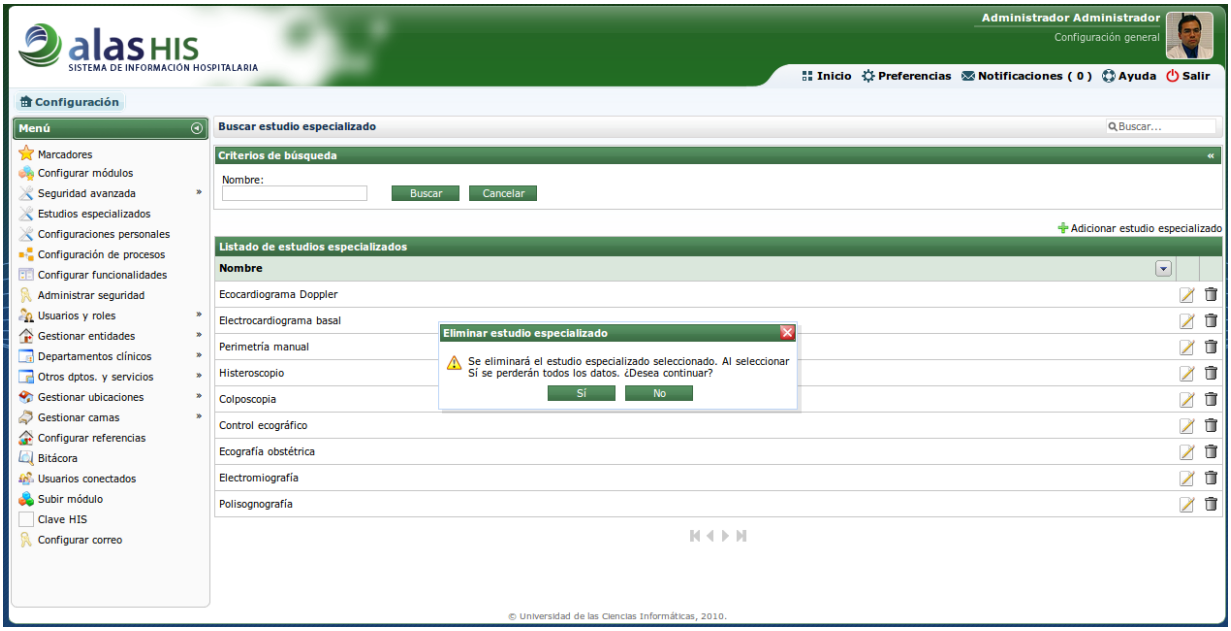


Imagen 1 Eliminar estudio especializado en el módulo Configuración.

Anexo 4. Proceso: Gestionar horarios para estudio especializados.

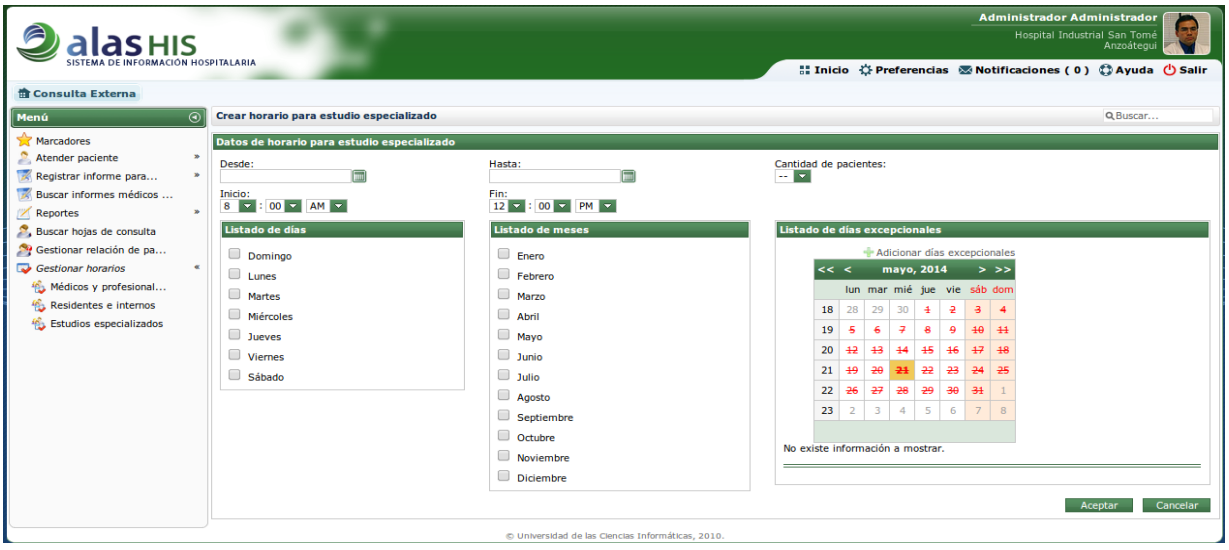


Imagen 2 Crear horario para estudio especializado en el módulo Consulta Externa.

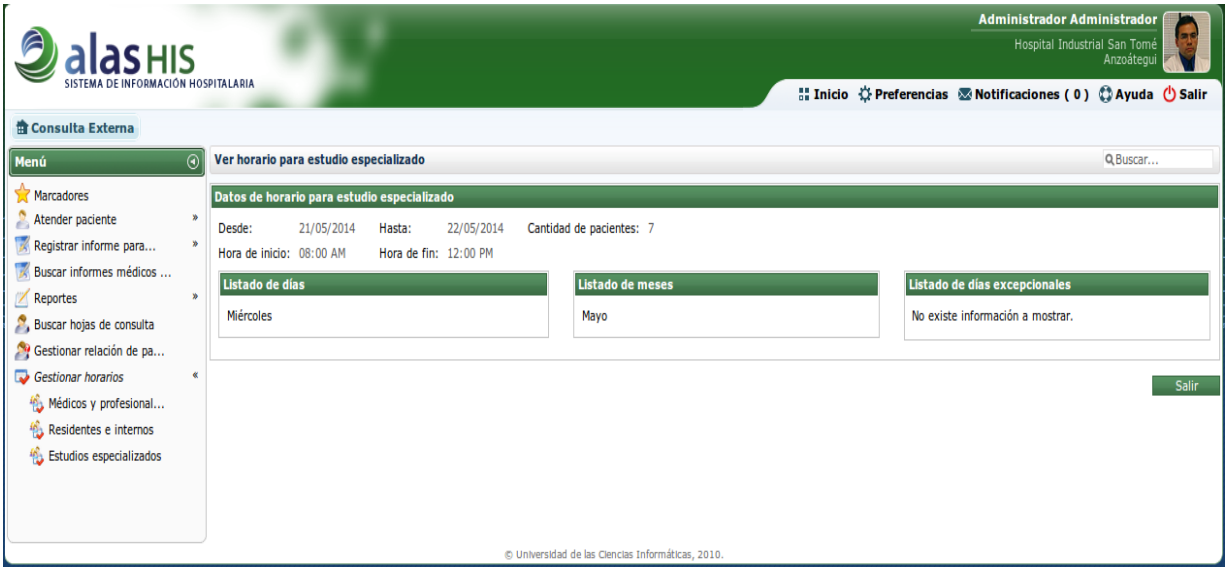


Imagen 3 Ver datos de horario para estudio especializado en el módulo Consulta Externa.

Anexo 5. Proceso: Asignar cita para estudio especializado en el módulo Consulta Externa.

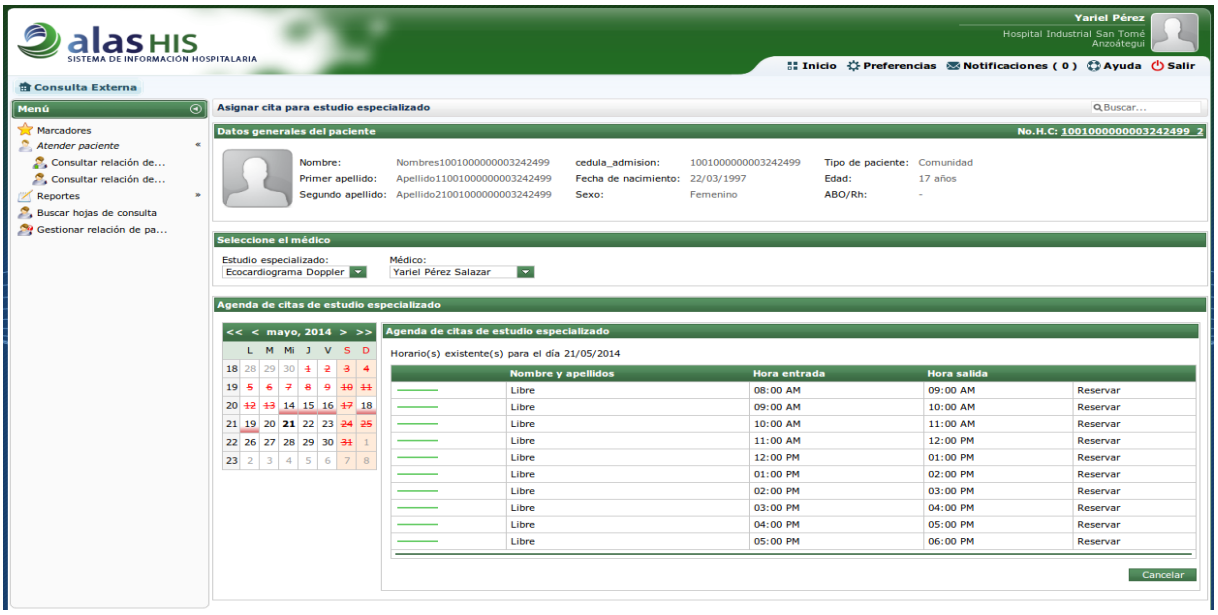


Imagen 4 Asignar cita para estudio especializado en el módulo Consulta Externa.

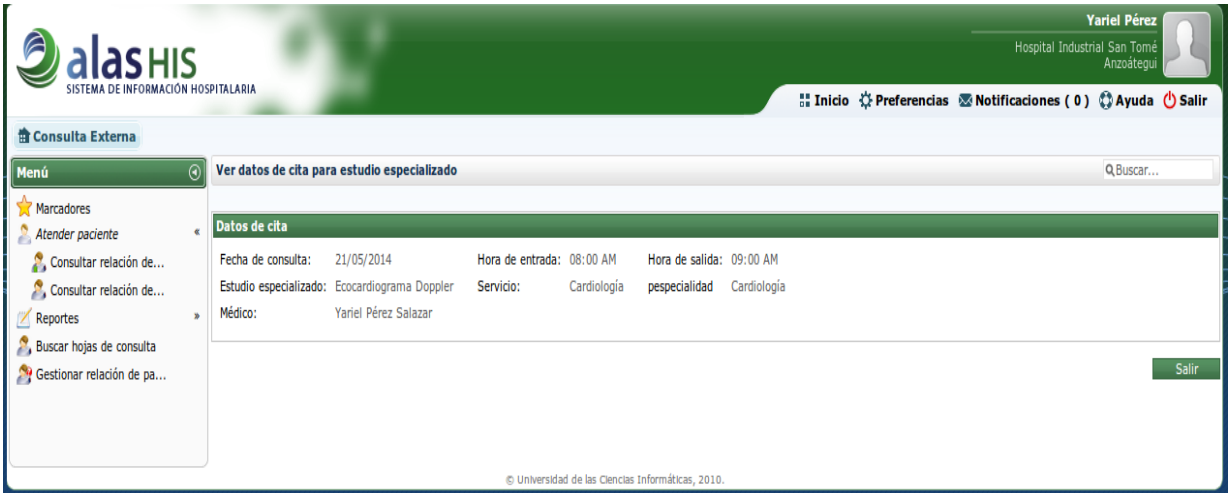


Imagen 5 Ver datos de cita para estudio especializado en el módulo Consulta Externa.

Anexo 6. Proceso: Asignar cita para estudio especializado en el módulo Citas.

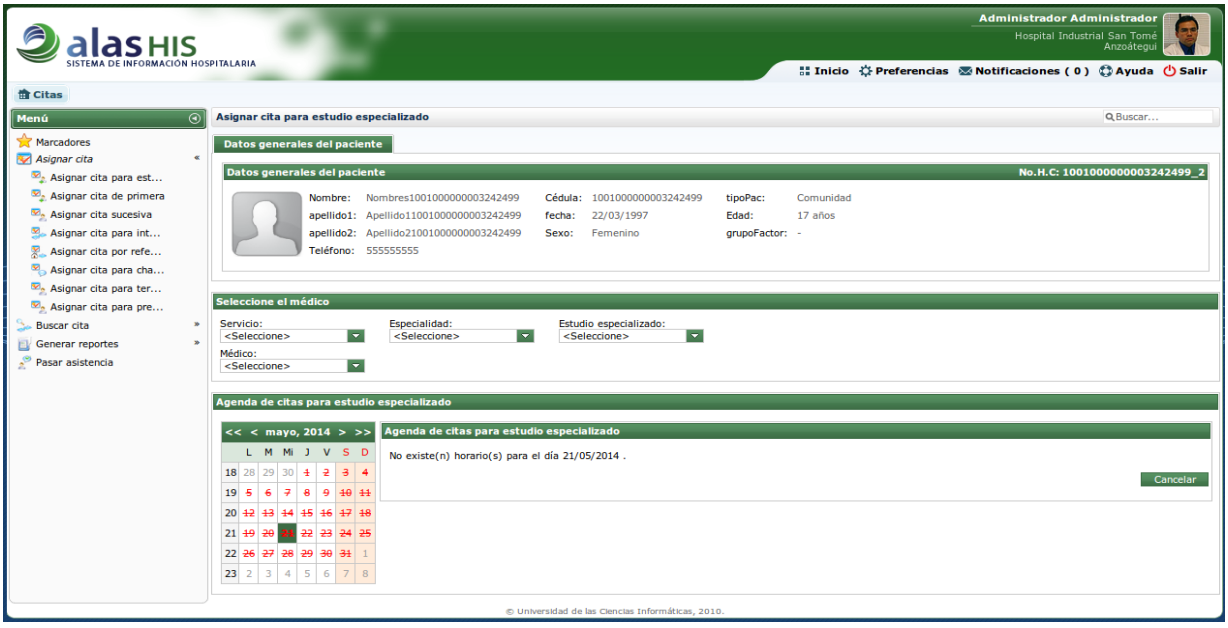


Imagen 6 Asignar cita para estudio especializado en el módulo Citas.

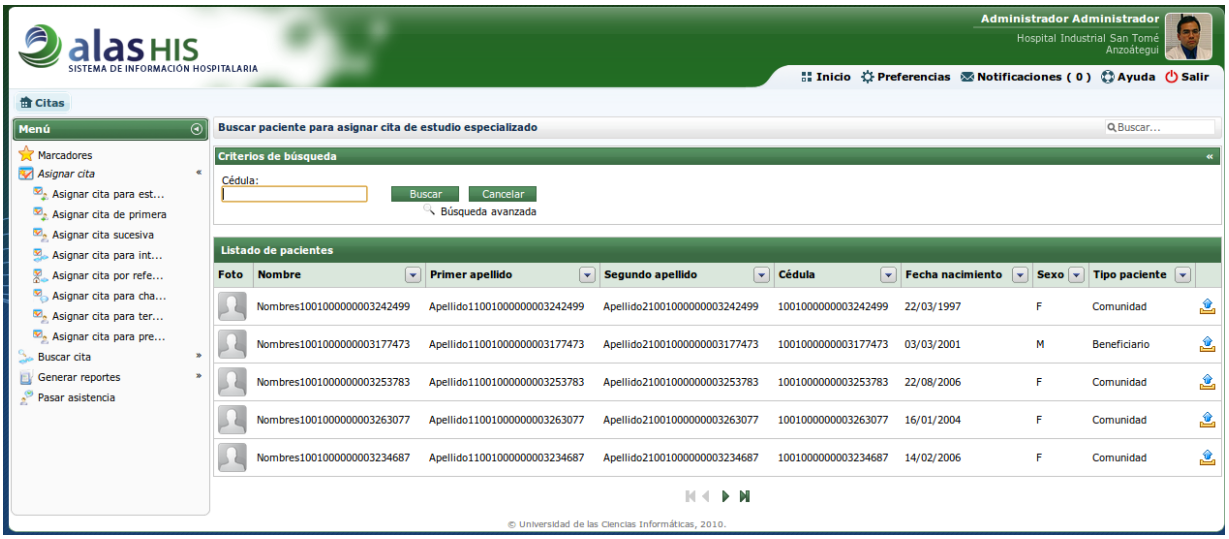


Imagen 7 Buscar paciente para asignar cita de estudio especializado en el módulo Citas.

GLOSARIO DE TÉRMINOS

HL7: Es una especificación para un estándar de intercambio de datos electrónicos en el área de la salud, con especial énfasis en las comunicaciones intra–hospitalarias en el área de la información clínica y administrativa.

PACS: Sistema para el almacenamiento y comunicación de imágenes médicas.

PDF (Portable Document Format): Es un formato de fichero que simula un documento impreso como una imagen electrónica que se puede leer, crear, navegar, imprimir, ente otros.

Plain Old Java Object (POJO): Enfatiza el uso de clases simples y que no dependen de un framework en especial.

Sistema de Información Radiológica (RIS-Radiological Information System): Es un sistema encargado de la gestión de la información generada y manipulada como resultado de los procesos de negocio de carácter radiológico (imagenológico).

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

SNOMED: Es la principal terminología clínica de referencia seleccionada para la Historia Clínica Digital del Sistema Nacional de Salud (SNS), lo que supone un primer paso fundamental hacia la interoperabilidad semántica de la información clínica del SNS.

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

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

IDE: Ambiente integrado de desarrollo (Integrated Development Environment). Es un conjunto de software que permite el desarrollo de aplicaciones.