Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación de los procesos del movimiento hospitalario del Sistema de Información Hospitalaria alas HIS.

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

Autor: Jorge Díaz Rodríguez

Tutores: Ing. Karel Gómez Velázquez

Ing. Reinel Muñoz Pérez

Ciudad de la Habana, junio de 2010

"Año del 52 de la Revolución"

ÍNDICE

INTRODUCCION 1

CAPITULO I: FUNDAMENTACION TEORICA 6

1.1 Sistemas de salud pública 6

1.2 Sistemas de Información Hospitalaria (HIS) 7

1.3 Antecedentes 7

1.3.1 Metis HIS 7

1.3.2 XHosp 8

1.3.3 HIS CNT Pacientes 8

1.3.4 Aplicación de Gestión de Guardias Médicas 9

1.3.5 Galen Hospital 9

1.4 Conceptos relacionados con el campo de acción 10

1.5 Tendencias y tecnologías actuales a considerar 10

1.5.1 Modelo Cliente - Servidor 10

1.5.2 Patrones de Arquitectura y Diseño 11

1.6 Tecnologías utilizadas en el proceso de desarrollo 12

1.6.1 Capa presentación 13

1.6.2 Capa de Negocio 14

1.6.3 Capa de Datos 15

1.6.4 Tendencias y metodologías horizontales 16

1.7 Herramientas 18

CAPÍTULO II: DESCRIPCION DE LA ARQUITECTURA 20

2.1 Requerimientos no funcionales 20

2.2 Descripción de la Arquitectura 24

2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados 25

2.4 Seguridad 25

2.5 Vista de Despliegue 26
2.6 Estrategias de codificación. Estándares y estilos a utilizar 27
CAPITULO III: DESCRIPCION Y ANALISIS DE LA SOLUCION PROPUESTA 33
3.1 Valoración crítica del diseño propuesto por el analista 33
3.1.1 Diagramas de clases del diseño 37
3.1.2 Diagramas de secuencia 39
3.2 Descripción de las nuevas clases u operaciones necesarias. 41
3.3 Modelo de Datos 45
3.4 Valoración de las técnicas de validación 47
3.5 Descripción de las tablas 47
3.6 Vista de Implementación 51
3.6.1 Diagrama de componentes 51
3.7 Tratamiento de excepciones 52
CAPITULO IV: MODELO DE PRUEBAS 54
4.1 Pruebas de caja negra 54
4.1.1 Descripción de los casos de prueba 56
CONCLUSIONES 64
RECOMENDACIONES 65
REFERENCIAS BIBLIOGRAFICAS 66
BIBLIOGRAFIA 69
GLOSARIO DE TERMINOS 72

ÍNDICE DE FIGURAS

Figura 2.1 Diagrama de Despliegue.....27
Figura 3.2 Diagrama de CD_GestionarOrdenTransferencia 37
Figura 3.3 DC_ProcesarTransferencia.....38
Figura 3.4 DS_CrearOrdenTransferencia 39
Figura 3.5 DS_EliminarOrdenTransferencia 39
Figura 3.6 DS_ModificarOrdenTransferencia 40
Figura 3.7 DS_VerOrdenTransferencia.....40
Figura 3.8 Modelo de Datos. 46
Figura 3.9 Modelo de Componentes. 52
Figura 4.1 Enfoque de diseño de pruebas de caja negra. 54

ÍNDICE DE TABLAS

Tabla 2.1 Notación de los controles 32
Tabla 3.1 CrearOrdenTransferenciaHospControlador 42
Tabla 3.2 VerOrdenTransferenciaHospControlador 43
Tabla 3.3 ModificarOrdenTransferenciaHospControlador 44
Tabla 3.4 EliminarOrdenTransferenciaHospControlador 45
Tabla 3.5 Admisión 48
Tabla 3.6 Egreso 49
Tabla 3.7 Orden de transferencia 50
Tabla 3.8 Transferencia 51
Tabla 4.1 CP Crear Orden de Transferencia 58
Tabla 4.2 SC Crear Orden Transferencia 61
Tabla 4.3 CP Eliminar Orden Transferencia 62
Tabla 4.4 SC Eliminar Orden Transferencia 63

INTRODUCCION

Con el aumento de la población mundial y la utilización de nuevas técnicas de diagnóstico médico, los sistemas sanitarios requieren de un manejo racional y eficaz. Las innovaciones tecnológicas, el aumento de la expectativa de vida, así como el incremento del nivel de información de la sociedad y por lo tanto una mayor exigencia de la misma, han influido en la necesidad de elaborar sistemas con mayor grado de eficiencia y eficacia.

En dependencia de la complejidad de las acciones preventivas, curativas o de rehabilitación, así como el grado de especialización de los servicios brindados, la atención médica se ha organizado en tres niveles: la Atención Primaria, la Atención Secundaria u Hospitalaria y la Atención Terciaria o Especializada. La Atención Secundaria es proporcionada fundamentalmente por hospitales, ya que también puede ser ofrecida en clínicas de especialidades. En el caso, en que una persona haya sido atendida en el nivel primario y el personal de salud que lo atendió considera que requiere de una atención de mayor complejidad, este paciente es remitido a una institución hospitalaria.

Los hospitales como centros principales de los sistemas sanitarios, 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 requerida. Como consecuencia de esto se ha visto un incremento paulatino del uso de la informática, ciencia que se encarga del procesamiento automático de la información, mediante dispositivos electrónicos y sistemas computacionales, con el objetivo de obtener una herramienta que permita recoger y tratar la información de forma tal que sea útil para la toma de decisiones, dando lugar a los Sistemas de Información Hospitalaria (HIS por sus siglas en inglés). [1][2]

Los Sistemas de Información Hospitalaria están orientados a satisfacer las necesidades de generación de la información, almacenamiento, procesamiento e interpretación de datos médico – administrativos de las instituciones hospitalarias, lo cual permite una mayor optimización de los recursos, tanto humanos como materiales. En la actualidad existe un gran número de naciones que han optado por la utilización de los HIS, existiendo entidades que se especializan en su desarrollo y comercialización, siendo sus precios excesivamente altos. [3]

En países del primer mundo como son Estados Unidos y el Reino Unido, se pueden encontrar gran número de centros de atención de salud que brindan sus servicios y gestionan la información mediante este tipo de sistemas. Por otra parte en América latina, existen varios países, que tienen entre sus principales metas: la informatización de la salud, pero para llevar a cabo dicha informatización es

necesario modernizar y actualizar las infraestructuras existentes de sus instituciones del nivel secundario o construyendo nuevas con el correspondiente equipamiento tecnológico.

Una institución hospitalaria se encuentra conformada por un conjunto de áreas o departamentos, los cuales interactúan entre sí de forma dinámica. Una de estas áreas es la Hospitalización, la cual es la encargada de atender a los pacientes cuando su estado de salud necesita una atención médica permanente, la cual es administrada por el personal asistencial de la misma, posibilitando de esta forma su pronta recuperación. [4]

Un paciente que se encuentre ingresado en el área de Hospitalización, está sujeto a movimientos hospitalarios, los cuales no son más que el conjunto de ingresos, egresos y traslados que se le realizan durante su estancia en la institución hospitalaria, evidenciándose en esta área, con la realización de la transferencia y el egreso, dichos movimientos son solicitados por el médico que lo atiende y procesados por los Técnicos de Estadísticas de Salud, del servicio en que se encuentra. [5]

La transferencia, se lleva a cabo cuando un paciente presenta síntomas que indican que necesita atención médica de un servicio en el cual no se encuentra hospitalizado y es necesario trasladarlo a dicho servicio, además puede realizarse de igual manera para llevarse a cabo movimientos hospitalarios dentro de un mismo servicio. Al realizarse en forma manual, las transferencias se retrasan considerablemente, lo cual atenta contra la salud del paciente y la calidad de la atención brindada, debido a que no llega con la rapidez requerida al servicio receptor.

Por otra parte, el egreso se produce cuando un paciente ha recuperado su estado de salud y no requiere de un monitoreo periódico, haya fallecido o por contra indicación médica. En cualquiera de las formas mencionadas anteriormente, el médico que llevó a cabo el egreso, debe realizar un resumen de la estancia del paciente en el servicio de hospitalización, con aquellos aspectos que considera son significativos, momento en el que además se conforma el Diagnóstico Clínico Final. Cabe destacar que al concluirse las transferencias o egresos, se realiza una actualización del estado de las camas dentro del área, liberándose las mismas al no tener pacientes asociados a ellas. [6]

Al ser registrada la información referente al egreso y a las transferencias en formato duro, esta información queda propensa su pérdida como consecuencia del posible deterioro del documento que lo contiene, ya sea a causa de las condiciones del lugar donde se almacene o debido a la frecuente consulta que se le realiza a dicha información. Es importante resaltar que llegará un momento, en el cual será muy difícil gestionar esta información debido al aumento del volumen de documentos almacenados. Además puede llegar a ocurrir, que las instituciones hospitalarias no cuenten con

información detallada de último minuto, sobre la disponibilidad de nuevas camas para la admisión de pacientes en los servicios. [7]

Otro de los procesos que se lleva a cabo en el área, es la planificación de la guardia médica, cuyo objetivo es el de mejorar el control del personal médico dentro de esta. El jefe de servicio es el responsable de llevarlo a cabo, el cual consiste en distribuir al personal médico de forma tal que siempre exista un médico en cada servicio, pendiente del estado de salud de los pacientes hospitalizados. Al ejecutarse este proceso de forma manual se hace muy difícil llevar a cabo una planificación que no coincida con otras actividades del médico en la institución hospitalaria, tales como: horarios de consulta, intervenciones quirúrgicas, reuniones, etc. [8]

Por lo descrito anteriormente, se determina que el ***Problema a resolver*** es: ¿Cómo facilitar la gestión relacionada con los procesos del movimiento hospitalario y planificación de la guardia médica en el área de hospitalización de las instituciones hospitalarias?

Teniendo como ***Objeto de estudio*** de la presente investigación: Proceso de gestión de información en el área de hospitalización de las instituciones hospitalarias. Enmarcado en el ***Campo de acción,*** de los Procesos de gestión del movimiento hospitalario y de la planificación de la guardia médica en el área de Hospitalización de las instituciones hospitalarias.

Para resolver el problema identificado se propone el siguiente ***Objetivo general***: Implementar los procesos de gestión del movimiento hospitalario y de la planificación de la guardia médica del área de hospitalización de las instituciones hospitalarias.

Para dar cumplimiento al objetivo planteado se definen las siguientes ***Tareas de la Investigación***:

1. Analizar los procesos de negocio relacionados con el movimiento hospitalario y la planificación de la guardia médica en el área de Hospitalización de las instituciones hospitalarias.
2. Evaluar las ventajas y desventajas de sistemas de informática médica que cuenten con funcionalidades para gestionar los procesos del movimiento hospitalario y planificación de la guardia médica para instituciones de salud.
3. Asimilar la arquitectura y pautas definidas por el Departamento de Sistema de Gestión Hospitalaria para el desarrollo de sus aplicaciones.

4. Obtener mediante el Proceso Unificado de Desarrollo, los artefactos correspondientes a los flujos de trabajo “Diseño”, “Implementación” y “Pruebas” relacionados con los procesos de gestión del movimiento hospitalario y planificación de la guardia médica.

Con la informatización de los procesos de gestión del movimiento hospitalario y la planificación de la guardia médica, que se llevan a cabo en el área de Hospitalización de las instituciones hospitalarias, se propiciarán un conjunto de beneficios; entre los cuales se pueden mencionar:

Para el paciente:

1. Recibir una atención de salud con mayor calidad, gracias a que la atención sanitaria se lleva a cabo de forma más rápida y se garantiza que siempre exista un médico pendiente del estado de salud de los pacientes.

2. Mayor rapidez en la atención médica, gracias a la disminución del tiempo que se demora la realización de las transferencias entre los servicios del área de Hospitalización.

Para el personal que labora en el área:

1. Disponer de un sistema que posibilite la planificación de las guardias médicas del área de Hospitalización, obteniéndose un mayor control de la distribución de la guardia médica del personal que labora en el área.

2. Conocer en cada momento de la disponibilidad real de camas por servicio, debido que cuando se realice cualquier movimiento hospitalario se garantizará la actualización del estado de las mismas.

3. Permitir que la información clínica que se genere durante el proceso de atención a los pacientes, sea almacenada de modo que se garantice su persistencia en el tiempo, con lo que se evita así su posible deterioro.

El presente documento se encuentra estructurado u organizado en cuatro capítulos, el primero de ellos **FUNDAMENTACION TEORICA**, aborda en el ambiente de desarrollo para la informatización de los procesos concernientes a la gestión del movimiento hospitalario y la planificación de la guardia médica en el área de Hospitalización, se enuncian además las tendencias actuales, tecnologías, metodologías y herramientas que fueron utilizadas para su desarrollo.

A continuación el segundo capítulo, **DESCRIPCION DE LA ARQUITECTURA** se centra en la estructuración del Modelo de Diseño el cual posibilitará la correcta construcción de la aplicación. En el

tercer capítulo, **DESCRIPCION Y ANALISIS DE LA SOLUCION PROPUESTA**, se describe el sistema en términos de componentes y subsistemas de implementación, obteniéndose una solución que posibilite una mayor eficiencia de los movimientos hospitalarios y la planificación de la guardia médica del área de hospitalización. Finalmente en el capítulo, **MODELO DE PRUEBAS**, se obtienen los artefactos propios del flujo de pruebas tales como los diseños de los casos de pruebas, listas de no conformidades y flujos de procesos de integración; obteniéndose una evaluación de la calidad de las funcionalidades implementadas.

CAPITULO I: FUNDAMENTACION TEORICA

El presente capítulo tiene como objetivo mostrar los diferentes elementos que brindan la base teórica y conceptual para el desarrollo de los procesos del movimiento hospitalario y planificación de la guardia médica en el área de Hospitalización. Se abordan los antecedentes existentes de sistemas que dan solución a los procesos antes mencionados, además de los principales conceptos relacionados con el dominio del problema y el campo de acción. Por último se exponen las características de las diferentes tecnologías y herramientas que se utilizaron en cuenta en el proceso de desarrollo.

1.1 Sistemas de salud pública

La salud pública está considerada como la disciplina encargada de la protección de la salud en el nivel poblacional. Busca mejorar las condiciones sanitarias mediante la promoción de estilos de vida saludables, las campañas de concienciación, la educación y la investigación. Entre las funciones de la salud pública, se encuentran la prevención epidemio - patológica (con vacunaciones masivas y gratuitas), la protección sanitaria (control del medio ambiente y de la contaminación), la promoción sanitaria (a través de la educación) y la restauración sanitaria (para recuperar la salud).

Es de vital importancia asegurar una adecuada gestión de los servicios de salud pública, lo cual posibilita que estos lleguen al mayor número de personas posibles, teniendo en cuenta que no se deberían hacer distinciones entre los habitantes de una misma región, debido a que se pueden brindar servicios de avanzada a varias personas y descuidar al resto de la población. En dependencia de la complejidad de las acciones sanitarias, debido al nivel de especialización de los servicios que se ofertan, los niveles de atención médica se han organizado en tres categorías, los cuales son: [10]

1. **Atención Primaria de Salud:** Asistencia Sanitaria esencial, basada en métodos y tecnologías prácticas, científicamente fundadas y socialmente aceptadas, puestas al alcance de todos los individuos y familias de la comunidad mediante su plena participación. Constituyendo la función central y el núcleo principal del sistema nacional de salud. Representa el primer nivel de contacto entre los individuos, la familia y la comunidad con el sistema nacional de salud. [9]
2. **Atención Secundaria u Hospitalaria:** Es proporcionada en instituciones hospitalarias y clínicas de especialidades, su función fundamental es tratar al hombre ya enfermo. En el caso de que una persona haya sido atendida en el nivel primario y se considere que requiere de una atención de mayor complejidad, ese paciente es remitido hacia un hospital, con el objetivo de llevar a cabo un monitoreo de su condición clínica.

3. **Atención Médica Terciaria:** El nivel terciario debe abarcar alrededor del 5 % de los problemas de salud, relacionados con secuelas o aumento de las complicaciones de determinadas dolencias. Se brindan servicios de muy alta complejidad, con la óptima utilización de los recursos y medios existentes y el desarrollo de la investigación. A este nivel pertenecen los institutos y hospitales especializados.

1.2 Sistemas de Información Hospitalaria (HIS)

Debido a los grandes volúmenes de información que se generan en la atención secundaria, así como al grado de especialización de las acciones de salud, que en este nivel se llevan a cabo, se hace necesaria la utilización de los sistemas de información. Los cuales permiten recopilar, administrar y manipular un conjunto de datos, que conforman la información necesaria para la toma de decisiones en una organización. En resumen, es aquel conjunto ordenado de elementos (no necesariamente computacionales), que permiten manipular toda aquella información necesaria, para implementar aspectos específicos para la toma de decisiones.

Entre los sistemas de información se encuentran: los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés). Los cuales tienen como propósito, permitir la optimización de los recursos humanos y materiales, con el objetivo de satisfacer las necesidades de las áreas operativas, administrativas, clínicas y de investigación en las organizaciones de salud. Un HIS es un sistema modular, porque permite tener una base de datos de tipo demográfico, información de asegurados así como datos clínicos y estadísticos. Se caracteriza por utilizar una identificación numérica para cada uno de los pacientes. Además actúan como sistemas bases para todos los demás sistemas, ya sean clínicos, financieros o administrativos, relacionados con el paciente. [11]

1.3 Antecedentes

Como parte de la investigación, se ha llevado a cabo un estudio con el objetivo de conocer el estado del arte, obteniéndose como resultado un grupo de sistemas que poseen entre sus funcionalidades la gestión de los movimientos hospitalarios, así como la planificación de las guardias médicas; constituyendo este análisis un punto de partida que brinda una base, con el objetivo de llevar a cabo la investigación. A continuación se muestran las principales características de los sistemas valorados:

1.3.1 Metis HIS

El producto fue desarrollado por la empresa Uruguaya Apraful Software. Siendo una solución para entornos web, pensada para gestionar organizaciones de salud de manera integral, y permitir un control absoluto de las mismas. Este sistema es capaz de generar información sobre puntos tales

como: pacientes, áreas clínicas y administrativas además de la gerencia y dirección. A través del Módulo Sanatorio se registra el ingreso y el egreso de pacientes, también incluye la correspondiente codificación de las patologías al egreso. Además del ingreso de hojas de evolución, indicaciones médicas, controles de enfermería y todo tipo de documento que se quiera definir. Como producto final de la Historia Clínica de interacción. [12]

1.3.2 XHosp

Es un Sistema de Información Hospitalaria desarrollado por la empresa mexicana Virtus Group, con el objetivo de dar apoyo operacional, así como control administrativo integral a un hospital o clínica. Fue diseñado como un sistema modular y escalable, el cual se puede desplegar en la mayoría de las instituciones hospitalarias, desde pequeñas clínicas hasta grandes centros médicos. Esta aplicación lleva a cabo un control del egreso de los pacientes, registrando el tipo de egreso y los diagnósticos que se generaron. [13]

Al mismo tiempo hace posible llevar el control de las transferencias de los pacientes entre las diferentes áreas de hospitalización. Permitiendo la explotación de información médica con fines estadísticos, así como la información administrativa con fines de control. Además de aprovechar todas las ventajas de una aplicación diseñada y desarrollada para Windows: interfaz de usuario gráfica y amigable, capacidad de compartir la información con otras aplicaciones de Windows como Word, Excel o Access entre otras. Lo cual permite la utilización de prácticamente cualquier base de datos para el almacenamiento de la información del paciente. [14]

1.3.3 HIS CNT Pacientes

Es un Sistema de Información Hospitalaria producido por la compañía colombiana CNT Sistemas de información S.A., centrándose este sistema principalmente en la Historia Clínica del paciente. Posee sus funcionalidades agrupadas en varios módulos, en dependencia de los servicios ofrecidos a los pacientes, pudiéndose nombrar entre los módulos pertenecientes a este producto: Historia Clínica Dinámicas Hospitalaria, Enfermería, Hospitalización, así como Emergencias. [15]

Mediante el módulo de Hospitalización, el sistema administra eficientemente la gestión de las camas de la institución, y le permite a la enfermera o personal encargado, registrar las transferencias internas de los pacientes, así como dar el alta de los mismos, teniendo la posibilidad de cambiar el estado de las camas en que se encontraban aquellos pacientes a los cuales se les dio el alta del hospital. Además permite registrar los diferentes tipos de admisiones, y además posibilita la asignación de los tipos de estancia, con lo que se asigna la ubicación al paciente interno. Otras funciones del módulo,

permiten el registro de diagnósticos de salida, estado a la salida y los nacimientos. [16]

1.3.4 Aplicación de Gestión de Guardias Médicas

Este sistema fue creado por la empresa Open Sistemas, una empresa multinacional con presencia tanto en el continente europeo como en el americano, Es una herramienta que se integra al portal Univadis.net, aporta a los médicos que visitan dicho portal, un mecanismo para gestionar sus guardias en distintos hospitales, tramitando sus datos, los datos de hospitales, jornadas trabajadas, precio por hora y obtener partiendo de esta información, informes periódicos acerca de su trabajo. Esta aplicación fue llevada a cabo, como un contenedor configurable de información que se incluye en la estructura de un portal, bajo tecnología PHP, utilizando Apache como servidor web en una plataforma Linux, accediendo a sistemas de base de datos MySQL y sistemas de directorio openLDAP. [17]

1.3.5 Galen Hospital

Ha sido creado por la empresa de origen cubano SOFTEL, contando con funcionalidades para la gestión del registro de pacientes, hospitalización, gestión de medios diagnósticos y consultas. Además de incluir conexión con equipos de diagnóstico y autoanalizadores, así como la emisión de informes de resultados y estadísticas. Posee módulos que ejecutan los procesos informativos vinculados con la admisión y alta de los pacientes para cada episodio hospitalario. Además posibilita establecer un monitoreo sobre la información asociada a la estancia de cada paciente en el hospital. [18]

Entre los procesos que son gestionados por este sistema se pueden mencionar el hecho de incluir los procesos de ingreso, traslados y egresos, posibilitando además el control del uso de las camas, así como los servicios de hospitalización. Al mismo tiempo provee información estadística a partir del movimiento hospitalario. Pero no posee entre sus funcionalidades, la gestión de la planificación de la guardia médica en el área de Hospitalización de las instituciones hospitalarias. Este sistema ha sido diseñado para ser ejecutado en Windows. [19]

A partir del análisis de los sistemas antes expuestos, se puede llegar a la conclusión de que la gran mayoría de estos, son desarrollados con herramientas y tecnologías propietarias y mayormente aplicaciones de escritorio lo cual eleva el costo y dificulta el proceso de despliegue, al ser necesario instalarlo de forma individual en todas computadoras que van a utilizar dichos sistemas. Además cabe destacar, que no se ha encontrado un HIS que gestione todas las funcionalidades referentes a los movimientos hospitalarios: transferencias y egresos, y la planificación de las guardias médicas para el área de Hospitalización de las instituciones hospitalarias.

1.4 Conceptos relacionados con el campo de acción

Con el objetivo de proveer un mayor entendimiento del negocio vinculado a la solución definida, a continuación se relacionan un conjunto de conceptos inherentes al dominio del campo de acción:

Hospitalización: Área de las instituciones hospitalarias, en la cual un paciente es admitido cuando su estado de salud requiere de un monitoreo constante por parte de personal de salud que labora en a misma.

Egreso: Movimiento hospitalario que se lleva a cabo cuando el paciente haya recuperado su salud, fallecido o se lleve a cabo contra indicación médica. Además al realizarse el egreso, se genera un conjunto de documentos, en función al tipo de egreso realizado.

Transferencia: Movimiento hospitalario que se lleva a cabo cuando el paciente presenta problemas de salud que no se pueden solucionar en el servicio en el cual se encuentra y es necesario trasladarlo a otro, cabe destacar que es posible la realización de una transferencia que tenga como destino el mismo servicio en el cual se encuentra el paciente.

Guardia médica: Actividad realizada por el personal médico del área de hospitalización, consiste en distribuir al personal de forma tal que se garantice la presencia de al menos un galeno en cada servicio, el cual estará al tanto de la condición médica de los pacientes.

Médico: Profesional de la salud que se encarga de emitir un diagnóstico sobre la condición sanitaria del paciente y suministrarle el tratamiento adecuado, además de monitorear su estado de salud mientras se encuentre en el área.

Servicio: Servicio del área de Hospitalización en el cual el paciente fue admitido, al coincidir su patología con el área de atención en el cual está especializado el servicio en cuestión.

1.5 Tendencias y tecnologías actuales a considerar

A continuación se definen las tecnologías actuales que se han analizado, con el objetivo de desarrollar una aplicación que provea una adecuada solución del problema a resolver.

1.5.1 Modelo Cliente - Servidor

Un sistema distribuido se define como una colección de computadoras autónomas conectadas por una red, y con el software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de computación. El Modelo Cliente - Servidor de un sistema distribuido, es el modelo más conocido y más ampliamente adoptado en la actualidad,

brindando un enfoque efectivo y de propósito general con el objetivo de compartir tanto información como recursos.

Cabe destacar la existencia de un conjunto de procesos servidores, cada uno actuando como un gestor para una determinada colección de recursos de un tipo y una colección de procesos clientes. Por lo tanto, todos los recursos compartidos son mantenidos y manejados por los procesos servidores. Los procesos clientes llevan a cabo peticiones a los servidores cuando necesitan acceder a algún recurso. Si la petición es válida, entonces el servidor lleva a cabo la acción requerida y envía una respuesta al proceso cliente. [20]

El Modelo Cliente - Servidor presenta varias características, destacándose algunas como:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor o del tipo de equipo físico en el que se encuentra, así como tampoco su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

1.5.2 Patrones de Arquitectura y Diseño

A continuación se presentan los patrones de diseños que se aplicaron en el desarrollo del sistema.

1.5.2.1 Arquitectura en Capas

En la arquitectura en capas se pone de manifiesto la técnica para resolver problemas, conocida como “divide y vencerás”, ya que al separar el sistema en varias capas, se logra disminuir el grado de complejidad del trabajo a llevarse a cabo, fraccionándolo en problemas mucho más sencillos, debido a que se puede trabajar de forma independiente sobre cada una de las capas, sin tener que poseer un conocimiento profundo de todas ellas. La arquitectura en 3 capas es un ejemplo de este tipo de arquitectura, pues se pone de manifiesto la existencia de capas lógicas: la presentación, la de negocio y la de datos.

La capa de presentación es la encargada de permitir la interacción entre el usuario del sistema y el sistema en sí, mediante las interfaces de usuario, mientras que la capa de negocio es aquella encargada de gestionar las funcionalidades que forman parte de los procesos que la aplicación debe gestionar, al mismo tiempo la capa de datos se encarga de gestionar la persistencia de la información, comúnmente utilizando un sistema de base de datos.

1.5.2.2 Modelo – Vista – Controlador

La arquitectura Modelo – Vista – Controlador (MVC), fue creada con el objetivo de reducir el esfuerzo de la implementación de sistemas múltiples, que se encuentran sincronizados a datos comunes. Este modelo posee varias ventajas, entre las que se encuentran:

- Permitir la implementación de los componentes de forma separada, al existir una clara separación entre los mismos.
- Mayor adaptación al cambio, ya que al no depender el modelo de las vistas se hace mucho más fácil modificar el entorno visual sin ocasionar afectaciones al modelo.
- La conexión entre el modelo y sus vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

El modelo es el objeto que representa los datos del programa, maneja los datos y controla todas sus transformaciones además no tiene conocimiento específico de los controladores ni de las vistas, pues ni siquiera contiene referencias a ellos; es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas así como notificar a las vistas cuando el modelo cambia. Esto trae consigo una desventaja, debido a que al experimentar el modelo cambios con mayor frecuencia puede provocar un desbordamiento de requerimientos de actualización en las vistas. [21]

La vista es el objeto que maneja la presentación visual de los datos representados por el modelo, generando una representación visual del modelo y mostrando los datos al usuario. Las vistas interactúan con el modelo a través de una referencia al propio modelo. Mientras tanto que el controlador es el objeto que proporciona significado a las ordenes del usuario, actuando sobre los datos representados por el modelo. Cuando se realiza algún cambio, este objeto entra en acción, bien sea por cambios en la información del modelo o por alteraciones de la vista, posibilitando su interacción con el modelo de la misma forma que lo hacen las vistas, a través de una referencia al propio modelo. [22]

1.6 Tecnologías utilizadas en el proceso de desarrollo

En continuación se muestran las tecnologías utilizadas en el proceso de desarrollo, las cuales han sido agrupadas según la capa en la que están presentes.

1.6.1 Capa presentación

1.6.1.1 JSF

Java Server Faces (JSF), es un framework de desarrollo para aplicaciones Web en Java. Además es muy sencillo y flexible, siendo capaz de crear componentes propios o utilizar los componentes ya existentes de acuerdo a las necesidades del problema.

1.6.1.2 Richfaces

Es un framework de código abierto, que añade capacidad Ajax dentro de aplicaciones JSF sin recurrir a JavaScript. Este framework incluye ciclo de vida, validaciones, convertidores, así como la gestión de recursos estáticos y dinámicos. Sus componentes están construidos con soporte Ajax y poseen un alto grado de personalización de sus interfaces gráficas, que puede ser fácilmente incorporado dentro de las aplicaciones JSF. Los componentes de interfaz de usuario de Richfaces vienen preparados para su uso fuera del paquete, así los desarrolladores pueden ahorrar tiempo y disponer de las ventajas anteriormente mencionadas para la creación de aplicaciones Web.

1.6.1.3 Seam UI

Serie de controles Seam altamente integrables con JBoss Seam. Están dirigidos a complementar los controles JSF incorporados y los controles de otras bibliotecas externas. [23]

1.6.1.4 Facelets

Es un framework para el diseño de páginas Web centrado en la tecnología JSF, por lo cual se integran de manera muy factible, además es muy fácil de usar y configurar. Facelets posee características tales como: [24]

- Trabajo basado en plantillas.
- Fácil composición de componentes.
- Creación de etiquetas lógicas a la medida.
- Funciones para expresiones.
- Desarrollo amigable para el diseñador gráfico.
- Creación de librerías de componentes.

1.6.1.5 XHTML

Los documentos XHTML están basados en XML, una familia de módulos y tipos de documentos que reproduce, engloba y extiende el HTML 4.0. Los tipos de documentos de esta familia están diseñados fundamentalmente, para trabajar en conjunto con aplicaciones de usuario basados en XML. El XHTML trae consigo varias ventajas, entre las que se encuentran: [25]

- Los documentos XHTML son conformes a XML. Como tales, son fácilmente visualizados, editados y validados con herramientas XML estándar.
- Los documentos XHTML pueden escribirse para que funcionen igual, o mejor que lo hacían antes, tanto en las aplicaciones de usuario conformadas con HTML 4.0, así como en las nuevas aplicaciones conformes a XHTML.
- Los documentos XHTML pueden usar aplicaciones que se basen, ya sea en el Modelo del Objeto Documento de HTML o XML.

1.6.1.6 Extended EL

Brinda una extensión al estándar conocido como lenguaje de Expresión Unificado (EL). Dicho estándar agrupa las características que posee JSF, su utilización posibilita una reducción de la cantidad de líneas de código en las páginas que se generen, con lo que trae consigo un aumento ostensible de la productividad, al disminuir la curva de aprendizaje necesaria para llevar a cabo cualquier desarrollo.

1.6.2 Capa de Negocio

1.6.2.1 JBoss Seam

Es un framework que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. Ha sido diseñado para simplificar al máximo el desarrollo de aplicaciones, basando el diseño en Plain Old Java Objects (POJOs) con anotaciones. Estos componentes se usan desde la capa de persistencia, hasta la capa de presentación, poniendo todas las capas en comunicación directa. Seam es el encargado de conectar a Enterprise JavaBeans 3 (EJB3) y Java Server Faces (JSF), ya que estos framework no han sido concebidos para trabajar juntos. Al mismo elimina la barrera existente entre estas tecnologías, permite usar EJBs directamente como "backing beans" de JSF y lanzar o escuchar eventos web. [26]

1.6.2.2 JBoss jBPM

La gestión de procesos de negocio BPM (Business Process Management) se encarga del modelado de este tipo de procesos y define el flujo de trabajo que rige su funcionamiento. Un flujo de trabajo modela de forma gráfica las relaciones entre las distintas tareas. JBoss jBPM es sistema de código abierto que gestiona estos flujos de trabajo y se encarga de automatizar la secuencia de acciones, actividades o tareas que componen el proceso. Esta automatización permite integrar los procesos de la empresa y rediseñarlos de acuerdo con ayuda de nuevas estrategias. [27]

1.6.2.3 Drools

Es un motor de reglas de negocio compatible con la especificación JSR-94 Rules Engine API. Siendo un componente que a partir de una información inicial y un conjunto de reglas, detecta qué reglas deben aplicarse en un instante determinado y cuáles son los resultados de esas reglas. Posee la ventaja de que si el negocio de una aplicación cambia, solo es necesario modificar las reglas del negocio, sin tener que realizarse la modificación del código, ni recompilar, ni detener la aplicación si se encuentra en ejecución. [28]

1.6.3 Capa de Datos

1.6.3.1 Java Persistence API (JPA)

Es el estándar para la persistencia de datos en Java, siendo el mismo un modelo de persistencia basado en clases simples, las cuales no dependen de un framework en específico, con el objetivo de mapear bases de datos relacionales en Java. Al crear el JPA, se combinaron ideas y conceptos de los principales frameworks de persistencia existentes, entre los que se destacan Hibernate, Toplink y JDO, además de las versiones anteriores de EJB; en la actualidad los frameworks mencionados con anterioridad cuentan con una implementación JPA.

El mapeo objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML; también pueden definirse transacciones como anotaciones JPA. Java Persistence API consta de tres áreas: [29]

- El Java Persistence API.
- El lenguaje de consulta.
- El mapeo de los metadatos objeto/relacional.

1.6.3.2 Enterprise JavaBeans (EJB3)

La especificación de Enterprise JavaBeans, define una arquitectura para un sistema transaccional de objetos distribuidos basados en componentes. La especificación posee un modelo de programación; convenciones o protocolos y un conjunto de clases e interfaces que crean el API EJB. Proporciona a los desarrolladores de Beans y a los vendedores de servidores EJB, un conjunto de contratos que definen una plataforma de desarrollo común. El objetivo de estos contratos es asegurar la portabilidad a través de los vendedores y el soporte de un rico conjunto de funcionalidades. [30]

1.6.3.3 Hibernate

Es una herramienta que realiza el mapeo entre el mundo orientado a objetos de las aplicaciones y el concepto asociado al modelo entidad - relación de las bases de datos utilizados en entornos Java. El término utilizado es ORM (Object Relational Mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional, a un modelo orientado a objetos y viceversa. Hibernate no solo realiza esta transformación, sino que proporciona capacidades para la obtención y almacenamiento de datos de la base de datos, reduciendo así el tiempo de desarrollo. [31]

1.6.4 Tendencias y metodologías horizontales

1.6.4.1 PostgreSQL

Es un sistema gestor de bases de datos objeto/relacional, que posee licencia BSD, la cual es una licencia original de una distribución de Software: *Berkeley Software Distribution* y permite el uso del código fuente en software no libre, por lo que su código fuente del PostgreSQL se encuentra disponible en la red, convirtiéndose en el gestor de bases de datos con código abierto más potente en la actualidad. Es un sistema multiplataforma, pudiendo ser utilizado en la mayoría de los sistemas operativos actuales, consta con varias versiones, cuya última versión, la 8.4 fue lanzada al mercado en junio del 2009, además de poseer con el apoyo de una comunidad de usuarios que colabora activamente con su desarrollo.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, *estabilidad, potencia, robustez, facilidad de administración e implementación de estándares* han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. [32]

1.6.4.2 Java

Es un lenguaje orientado a objetos, siendo creado por la empresa Sun Microsystems, con el objetivo de ser utilizado en la programación Web. Presenta varias semejanzas con lenguajes como C, C++ y C#, por lo que el programador que conozca estos lenguaje podrá entender Java con mayor rapidez. Con la programación en Java, se pueden crear los applets, aplicaciones especiales que se ejecutan dentro de un navegador, al ser cargada una página HTML en un servidor web, por lo general los applets son programas pequeños y de propósitos específicos. [33]

Java permite la modularidad por lo que se pueden hacer rutinas individuales que sean utilizadas por más de una aplicación, por ejemplo podemos contar con una rutina de impresión que puede servir para

un procesador de palabras, así como para una hoja de cálculo. Además de permitir el desarrollo de aplicaciones bajo el Modelo de Cliente - Servidor, como aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente y de esta forma logra distribuir el trabajo a realizar. [34]

1.6.4.3 JBoss Application Server

JBoss Application Server es uno de los servidores de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada J2EE, soporta todas las funcionalidades de J2EE 1.4, incluyendo servicios adicionales como clustering, caching y persistencia. JBoss es ideal para aplicaciones Java y aplicaciones basadas en la Web. También soporta Enterprise Java Beans (EJB) 3.0, y esto hace que el desarrollo de las aplicaciones sea mucho más fácil. [35]

1.6.4.4 Proceso Unificado de Desarrollo (RUP)

Es una metodología desarrollado y comercializado por Rational Software, una empresa perteneciente a IBM. Esta metodología posee varias características esenciales, las cuales están definidas como: [36]

- **Dirigido por casos de uso:** Se define un caso de uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los casos de uso representan los requisitos funcionales del sistema. Estos no solo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.
- **Proceso centrado en la arquitectura:** La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema, ayudando a determinar en qué orden se realizará. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo.
- **Proceso iterativo e incremental:** El equilibrio correcto entre los casos de uso y la arquitectura es algo muy parecido al equilibrio entre la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP, es tener un proceso iterativo e incremental, en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre casos de uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo.

1.6.4.5 Lenguaje Unificado de Modelado (UML)

El Lenguaje de Modelado Unificado (UML, Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientado a objetos, los cuales aparecen a fines de los 80 y principios de los 90, siendo como su nombre lo indica un lenguaje de modelado y no un método. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientado a objetos. Los autores de UML apuntan también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios. [37]

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño, indicando los pasos que se deben seguir para llegar a un diseño. La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo. [38]

1.6.4.6 Notación para el Modelado de Procesos de Negocio (BPMN)

El objetivo principal de la creación de BPMN es dar una notación rápidamente comprensible por todos los que intervienen en el desarrollo, desde el analista de negocio que hace el borrador inicial de los procesos, pasando por los desarrolladores técnicos responsables de implementar la tecnología que llevarán a cabo dichos procesos, llegando finalmente a las personas que gestionarán y monitorizarán procesos referidos anteriormente. BPMN define un Business Process Diagram (BPD, por sus siglas en inglés), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio, constituyéndose un modelo de procesos de negocio, como una red de objetos gráficos, que son actividades y controles de flujo que definen su orden de rendimiento. [39]

Un BPD está formado por un conjunto de elementos gráficos, estos habilitan el fácil desarrollo de diagramas simples que serán familiares para la mayoría de analistas de negocio (diagrama de flujo). Los elementos fueron elegidos para ser distinguibles los unos de los otros y para usar formas familiares para la mayoría de modeladores. Debe notarse que uno de los objetivos del desarrollo de BPMN es crear un mecanismo simple para crear modelos de procesos de negocio, y al mismo tiempo que sea posible gestionar la complejidad inherente en dichos procesos. [40]

1.7 Herramientas

Luego de un análisis de los elementos anteriormente expuestos, se ha definido como las herramientas a utilizar en el ambiente de desarrollo, las cuales se muestran a continuación:

- **Visual Paradigm 6.0**

  Visual Paradigm para UML, es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, además de generar código desde diagramas y generar documentación. [41]

- **Eclipse SDK 3.3.2**

  Eclipse SDK 3.3.2 es una plataforma gratuita e independiente, de código abierto para crear aplicaciones de cualquier tipo. Fue creada inicialmente por IBM y mantenido en la actualidad por el Proyecto Eclipse. Es totalmente extensible con módulos que aumentan su funcionalidad y que permiten que el entorno de desarrollo soporte otros lenguajes además de Java. Solo requiere tener instalado Java Runtime Environment 1.4.2 o superior. [42]

- **PostgreSQL**
  PostgreSQL como Sistema Gestor de Bases de Datos, cuyas características ya han sido abordadas a lo largo de este capítulo.

En este capítulo se hizo referencia a los principales conceptos vinculados con la gestión de los procesos de los movimientos hospitalarios, así como la planificación de la guardia médica en el área de Hospitalización de las instituciones hospitalarias. Realizándose un estudio de los antecedentes a nivel mundial que llevan a cabo estos procesos y se explican las tecnologías que se han definido para llegar a cabo el desarrollo del sistema.

CAPÍTULO II: DESCRIPCION DE LA ARQUITECTURA

A continuación se describen los requerimientos no funcionales que se han identificado a lo largo de la investigación y resulta imprescindible que posea el sistema que se desarrollará. Además de la descripción de la arquitectura definida, abordándose otros temas de interés, como la forma en que se manipulará la seguridad del sistema.

2.1 Requerimientos no funcionales

Los requerimientos no funcionales son cualidades o propiedades que el producto debe tener, es decir, restricciones en el producto que está siendo desarrollado. Estos no describen lo que el software hará, sino cómo lo hará. Son muy importantes debido a que permiten a los clientes valorar las características no funcionales del producto como: usabilidad, rendimiento y portabilidad, que junto a las funcionalidades esperadas del software ayudarán a marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. [43]

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

Fiabilidad

En los servidores de los hospitales se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos. Se garantizarán además, políticas de respaldo a toda la información, evitando pérdidas en caso de desastres ajenos al sistema.

Los estudios imagenológicos y otros datos que por su tamaño no se puedan replicar hacia el Centro de Datos, se almacenarán localmente en los hospitales; quedando la referencia a dicho estudio en el Centro de Datos, de tal forma que se pueda acceder a dichos estudios mediante una transmisión directa entre los hospitales, sin que medie para esto el Centro de Datos Nacional.

Las informaciones médicas relacionadas con los pacientes y que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas. 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. Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.

El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (log) de cada usuario en el sistema. Soportará el uso de firmas digitales para la transferencia de información cuya certificación sea imprescindible para validar el uso de la misma. Además implementará un control de cambios a determinados campos de información (seleccionados por su importancia), de forma tal que sea posible determinar cuáles han sido las actualizaciones que se le han realizado.

Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista. El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

Eficiencia

El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos. Además de minimizar el volumen de datos en las peticiones y optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos. Se deberá usar siempre que sea posible el patrón Singleton, destruir referencias que ya no estén siendo usadas, optimizar el trabajo con cadenas, entre otras buenas prácticas que ayudan a mejorar el rendimiento.

Soporte

- Seguridad de acceso y administración de usuarios

  Se mantendrá seguridad y control a nivel de usuario, garantizando su acceso 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 únicamente la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión. Registrándose 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 (log) de cada usuario. Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos. El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

- Monitoreo de funcionamiento

  Se permitirá administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación.

- Respaldo y recuperación de base de datos

  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.

- Auditoría

  Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema, para esto debe existir un registro de trazas que almacene todas las transacciones realizadas en el sistema, indicando para cada caso como mínimo: usuario que realizó la transacción, tipo de operación, fecha y hora en que se realizó la operación e información contenida en el registro modificado.

- Configuración de parámetros

  Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

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.

Interfaz

- Interfaces de usuario

  Las ventanas del sistema contendrán los datos claros y bien estructurados, además de permitir la interpretación correcta de la información. La interfaz contará con teclas de función y menúes desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario.

- Interfaces software

  Se interactuará con el sistema alas RIS para realizar solicitudes y obtener resultados de estudios radiológicos e imagenológicos.

Requerimientos de rendimiento

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Se respetarán las buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos.

Requerimientos de soporte

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP. Permitiéndose la administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación. Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema. Además de permitirse el establecimiento de parámetros de configuración del sistema y actualización de nomencladores.

Requerimientos de hardware

- Estaciones de trabajo

  En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS, las cuales necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, se para

Windows, Firefox 2 o versiones superiores para Linux. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz.

- 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.

  Servidor de Base de datos: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux.

  Servidor de Aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux.

  Servidor de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 2 GB de memoria y 2x72GB de disco y sistema operativo Linux.

Requerimientos de software

El sistema debe ejecutarse en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss y PostgreSQL).

2.2 Descripción de la Arquitectura

Una de las acciones que contribuyen a un buen desarrollo de software, es saber escoger sabiamente cual será la arquitectura que regirá el ambiente de trabajo. La cual provee de un esquema de referencia útil, con el que se guiará el desarrollo de un sistema informático. Por lo cual se ha definido como la arquitectura del sistema el patrón Modelo-Vista-Controlador, cuyas características generales han sido abordadas en el capítulo anterior.

La capa de presentación está desarrollada básicamente con JSF, usando la librería de componentes Richfaces 3.2.0 G.A., la cual se complementa fácilmente con el framework de integración Seam y permite generar vistas no necesariamente basadas en HTML (PDF, etc.), adiciona además controles out-of-the-box AJAX-ready y el framework de extensión AJAX para los controles JSF básicos Ajax4Jsf. Incluyendo conversión y validación de campos, establecimiento de reglas de navegación declarativas, la internacionalización y accesibilidad de la interfaz de usuario, además de un modelo orientado a eventos y combinado con Facelets, además de que nos brinda la capacidad añadida de la tecnología de plantillas de Facelets.

Seam es un poderoso y moderno framework creado para unificar todas las tecnologías estándares JSF, EJB3, JPA, además se hace uso de JBPM (se encarga de la administración de los procesos de negocio) y utilizando Drools como motor de reglas de negocio. Para el acceso a datos se usa la implementación de JPA de Hibernate 3.3, minimizando por un lado las configuraciones en XML sin chequeo de tipos y por otro lado usando los servicios del contenedor de EJB3 y/o los contextos de persistencias administrados por Seam, se elimina gran parte del código "infraestructural" en cuanto a transacciones, la transmisión del contexto de persistencia, etc, permitiendo además que se puedan establecer validaciones end-to-end gracias a los Hibernate Validators.

2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados

En la actualidad la reutilización de código fuente se ha convertido en una de las prácticas más utilizadas por los desarrolladores de software. Esta práctica trae consigo varias ventajas a la hora de llevar a cabo la implementación de cualquier sistema, permitiendo economizar tiempo y disminuir la redundancia que se genera. La manera más común de llevar a cabo esta reutilización, es copiarlo total o parcialmente, pero esto trae consigo que sea muy trabajoso mantener varias 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 proceso de desarrollo que se lleva a cabo, se cuenta con la ventaja de que existan algunos componentes que se utilizan de manera general por todo los módulos del Sistema de Información Hospitalaria alas HIS, uno de los componentes más utilizados es la clase UbicacionesManager, la cual se emplea para el trabajo con las ubicaciones de las camas que participan en los movimientos hospitalarios.

Además se maneja la clase Bitácora, la cual se utiliza para controlar las trazas de las acciones que son llevadas a cabo por los usuarios, mientras que se hace uso de la clase ActiveModule, la cual brinda la información sobre que módulo y entidad se encuentra el usuario que está utilizando el sistema, sin olvidar a la clase User, la cual ofrece los datos del usuario que se encuentra autenticado, entre otro conjunto de funcionalidades.

2.4 Seguridad

Se hace imprescindible que el sistema cuente con un fuerte nivel de seguridad, ya que este tipo de aplicaciones manipula información sensible de los pacientes, la cual solo debe ser accedida por el personal autorizado. Con el este objetivo se ha definido un control de acceso a nivel de usuarios y

contraseñas, con lo cual el usuario solo tendrá acceso a los niveles que le competan. También se cuenta con la existencia de una bitácora, en la cual se almacena todas las operaciones llevadas a cabo por el usuario, quedando registrado la fecha, la hora, así como la actividad que llevo a cabo el usuario en cuestión.

Cabe destacar que todo tipo de autorización, desde la autorización a directorios, páginas, controles, opciones del menú, servicios del negocio, está basada en reglas del negocio, con lo cual dichas reglas no se encuentran en el código de la aplicación, lográndose de esta forma una total independencia con respecto a este y posibilitando que cualquier cambio de las reglas del negocio, no afecte en lo absoluto al código fuente. Lo anteriormente expuesto es posible gracias a la integración existente entre Seam Security Framework y el potente motor de reglas JBoss Rules.

2.5 Vista de Despliegue

La vista de despliegue permite representar la distribución física del sistema, así como los distintos nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tales como servidores, computadoras, además de dispositivos tales como impresoras y scanners. Siendo el diagrama de despliegue la una parte fundamental de esta vista. El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado, que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas, y las relaciones entre sus componentes.

La mayoría de las veces el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema. Aunque UML no es un lenguaje de especificación hardware de propósito general, se ha diseñado para modelar muchos de los aspectos del hardware de un sistema, a un nivel suficiente para que un ingeniero de software, pueda especificar la plataforma sobre la que se ejecutará el sistema.

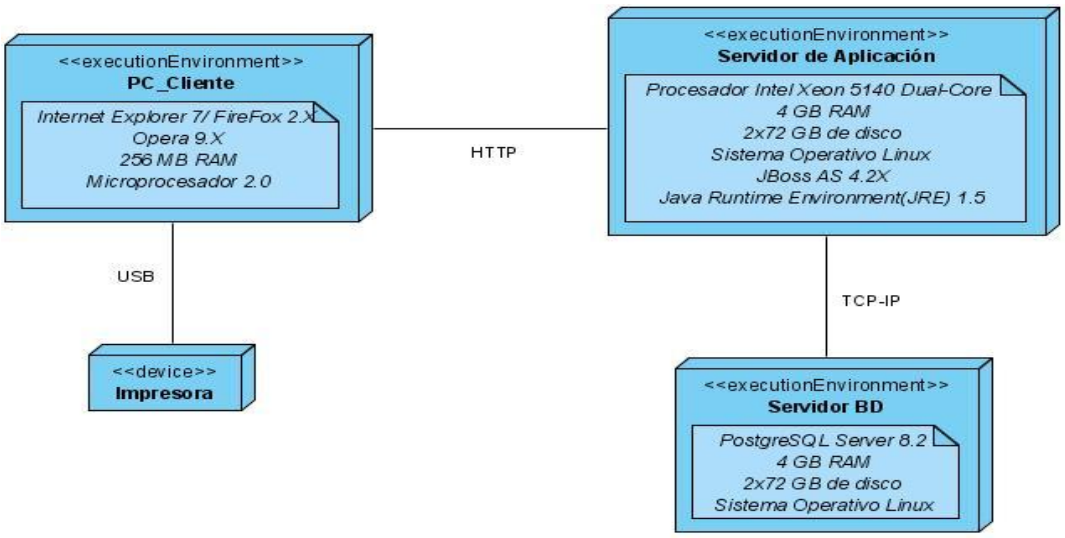

Figura 2.1 Diagrama de Despliegue

2.6 Estrategias de codificación. Estándares y estilos a utilizar

En general, los estándares de codificación son reglas que se deben seguir en el desarrollo de software, con el objetivo de posibilitar el entendimiento del código fuente generado, a otros programadores del mismo grupo de trabajo. Los estándares de codificación facilitan en gran medida el mantenimiento posterior de los sistemas que hayan sido creados basándose en ellos.

Identación

- Inicio y fin de bloque

  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.

- Aspectos generales

  El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la computadora o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar { en la línea de un código cualquiera, esto requiere una línea propia.

Comentarios, separadores, líneas, espacios en blanco y márgenes

- **Ubicación de comentarios**

  Al inicio de cada clase o función y al final de cada bloque de código.
  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.

- **Líneas en blanco**

  Se emplean antes y después de métodos, clases y estructuras.
  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.

- **Espacios en blanco**

  Entre operadores lógicos y aritméticos.
  Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo:

  producto = nomproducto

- **Aspectos generales**

  Sobre el comentario:
  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.

  Sobre los espacios en blanco:
  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

- **Apariencia de variables**

  Las variables tendrán un prefijo para el tipo de datos en minúscula.
  El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificara el tipo de datos al que se refiere (ver tabla 1.1), en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

  Ejemplo: sNombrePaciente.

- **Apariencia de constantes**

  Todas sus letras en mayúscula.
  Se deben declarar las constantes con todas sus letras en mayúscula.

- **Aspectos generales**

  Nombres de las variables y constantes.
  El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Clases y objetos

- **Apariencia de clases y objetos**

  Primera letra en mayúscula.
  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 comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.

- **Apariencia de atributos**

  Primera letra en minúscula.
  El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing*.

- **Apariencia de las funciones**

  Primera letra en mayúscula.

  Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación CamellCasing*. 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.

- **Declaración de parámetro en funciones**

  Agrupados por tipos.

  Poner los string 1 numéricos 2, además, agrupar según valores por defecto.

  Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen.

- **Aspectos generales**

  Sobre las clases, los objetos, los atributos y las funciones.

  El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

Bases de Datos, Tablas, esquemas y Campos

- **Apariencia de las vistas**

  Las 2 primeras letras representan el tipo. Todas las letras en minúscula.

  El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.

  Ejemplo:

  create view ‘vt_finanzas’.

- **Apariencia de los campos**

  Todas las letras en minúscula.

  El nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.

Ejemplo: ‘id_producto’.

- **Nombre de los campos**

  En caso de identificadores.
  Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo

  Ejemplo:

  id_municipio.

- **Sentencias SQL**

  Todas las letras en mayúscula.
  Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.

- **Aspectos generales**

  Sobre las BD, vistas, tablas atributos y procedimientos.
  El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con sólo leerlos se conozca el propósito de los mismos.

Controles

- **Apariencia de los controles.**

  Los controles tendrán un prefijo para el tipo de datos en minúscula.
  El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula, las cuales identificaran el tipo de datos al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

  Ejemplo: btnAceptar.

(*) Notación CamellCasing: Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamelCasing.

Control	Prefijo	Ejemplo
Botón	Btn	btnAceptar
Etiqueta	Lbl	lblNombre
Lista/Menú	Mn	mnPrincipal
Campo de Texto	Txt	txtFecha
Botón de Opción	Bpt	optSexo
Casilla de Verificación	Chx	chxBorrar
Casilla de Selección	Cbx	cbxSexo

Tabla 2.1 Notación de los controles

En este capítulo ha quedado definida la arquitectura del sistema, la cual es de vital importancia para finalizar del objetivo general de la investigación. Además se han expuesto los requerimientos no funcionales que deben cumplirse, para proveer un sistema de gran calidad, se ha dado una breve descripción de que pasos se guiarán para obtener un producto que sea lo más seguro, robusto y confiable posible. De igual forma, en el diagrama de despliegue ha quedado definida la propuesta de distribución física del sistema. Al mismo tiempo se ha expuesto el estándar de codificación con el objetivo de proveer una mayor comprensión del software, en vista a su futuro mantenimiento.

CAPITULO III: DESCRIPCION Y ANALISIS DE LA SOLUCION PROPUESTA

En el presente capítulo, se describe la solución propuesta, realizándose un análisis de la misma. Como parte de dicho análisis se lleva a cabo una valoración crítica del diseño presentado por el analista, se describirá las nuevas clases, el modelo de datos definido, así como una breve valoración de las técnicas de validación y la descripción de la vista de implementación.

3.1 Valoración crítica del diseño propuesto por el analista

Uno de los elementos principales de la ingeniería de software lo constituye el diseño, el cual tiene como objetivo principal la traducción de los requisitos, en una explicación que representa como llevar a cabo la implementación del sistema. Una de las ventajas que trae consigo la definición de un buen diseño, es el hecho de posibilitar la implementación del sistema sin equivocaciones. Además, el diseño describe las clases, la organización de las clases en paquetes y subsistemas, encontrándose entre sus artefactos el Modelo de Diseño.

El Modelo de diseño no es más que una abstracción de la implementación, siendo utilizada como entrada esencial del flujo de implementación, este tipo de modelo está organizado por un conjunto de subsistema que al integrarse en un todo, representa un sistema de diseño que denota el subsistema de nivel más alto del modelo, pudiendo contener varios artefactos como son: las clases, subsistemas, capsulas, protocolos, interfaces, entre otros que se puedan tener en cuenta, para el desarrollo del sistema.

En la concepción del diseño definido se requiere de la utilización de varios patrones de diseño, los cuales no son más que soluciones probadas, fruto de la experiencia de otros diseñadores de aplicaciones, constituyendo una buena práctica de programación el hecho de implementarlos, gracias al hecho de que brindan varias ventajas, al ser reusables, ya que se adaptan a varias circunstancias, obteniéndose de su utilización una un ahorro considerable de tiempo, al mismo tiempo posibilita la creación de un sistema más eficiente y dinámico.

En la etapa de implementación se hace necesario de un buen diseño de clases basado en patrones, el cual posibilita el aumento de la reutilización del código, con la consiguiente optimización del manejo de sus instancias. El framework Seam hace posible la persistencia de los datos, a través del uso de Hibernate, el cual permite abstraerse del gestor de bases de datos que se utilice, además de las ventajas que ofrece, al servir de puente entre el mundo relacional y la programación orientada a objetos, mediante el mapeo de las entidades del sistema.

Contándose entre uno de los principales componentes de la arquitectura el EntityManagerFactory, el cual se encarga de crear objetos EntityManager, siendo este componente el encargado de la implementación del patrón Abstract Factory, el cual permite el acceso a la base de datos. Por otra parte el mismo, es la interfaz principal del JPA, por lo cual es el responsable de llevar a cabo las operaciones del CRUD.

Hibernate utiliza además el patrón ActiveRecord, donde una fila de cualquier tabla de la base de datos se convierte en un objeto, de manera que se asocian filas únicas de la base de datos, con objetos del lenguaje de programación usado. Al mismo tiempo son implementados por Hibernate varios patrones de mapeo, contándose entre ellos a Identity Field, el cual guarda un campo identificador de la base de datos en un objeto para mantener la identidad entre un objeto en memoria y una fila de una tabla en una base de datos.

Otro de los patrones de comportamiento implementados por Hibernate es el Foreing Key Mapping, el cual mantiene una asociación entre objetos para referencias de llave foránea en tablas de la base de datos, lográndose de esta forma mapear relaciones de uno a muchos, y por último se utiliza Association Table Mapping, el cual permite mantener una asociación entre objetos para referencias de llave foránea en tablas de la de la base de datos, permitiendo mapear relaciones de muchos a muchos.

Por otra parte Hibernate utiliza además los patrones de comportamiento Identity Map y Lazy Load. Identity map es empleado con el objetivo de evitar tener en memoria dos representaciones distintas del mismo objeto en una transacción de negocio; por lo cual funciona como una caché de objetos de negocio.

Por su parte Lazy Load se encarga de solucionar los problemas de carga desproporcionada y las dependencias circulares, optimizando de esta forma la carga de información de la base de datos, manteniendo en memoria solamente a los datos que se estén utilizando. Cabe destacar otro patrón de gran importancia implementado por Hibernate, dicho patrón se nombra Query Object. El cual se encargar de interpretar la estructura de objetos y traducirlos a consultas SQL (Structured Query Language), facilitando de este modo el trabajo a los desarrolladores.

Como parte del diseño se recurrió a los patrones GRASP, los cuales se utilizan para la asignación de responsabilidades, obteniéndose como resultado un bajo acoplamiento en el sistema. Como parte de este patrón a cada clase se le asignó las tareas que podían ser realizadas, basándose en la información que dichas clases poseían, al mismo tiempo se les otorgó la habilidad de construir instancias de otras clases que fueran necesarias para llevar a cabo las responsabilidades

especificadas, trayendo como resultado la utilización de los patrones Experto y Creador, lográndose la conservación del encapsulamiento de la información.

En el diseño obtenido se evidencia el cumplimiento de los patrones de Bajo Acoplamiento y Alta Cohesión, los cuales se encargan respectivamente de las dependencias entre clases sean mínimas y que cada elemento lleve a cabo una única labor dentro de la aplicación, la cual no es realizada por ninguno de los otros elementos. Respondiendo a la arquitectura definida se llevo a cabo el modelo de diseño, cuyos criterios de empaquetamiento quedaron definidos por procesos y clases.

Cada proceso que se ha definido en el sistema se ha graficado en su paquete individual, utilizando las clases que se encuentran dentro del paquete del repositorio, las cuales a su vez se encuentran separadas en 3 subpaquetes:

Sesiones: Está compuesto por las clases controladoras autogeneradas, personalizadas y las controladoras propias del proceso. Siendo las autogeneradas creadas por el entorno de desarrollo, las personalizadas son aquellas que fueron modificadas y las últimas, aquellas que fueron creadas específicamente para el proceso.

Entidades: Está integrada por el conjunto de entidades autogeneradas que fueron obtenidas mediante el proceso de ingeniería inversa, utilizando el mapeo objeto – relacional de Hibernate. Además contiene las entidades personalizadas, las cuales son aquellas que han sido modificadas.

Vistas: Esta compuesta por el conjunto de clases que representan la interfaz de usuario.

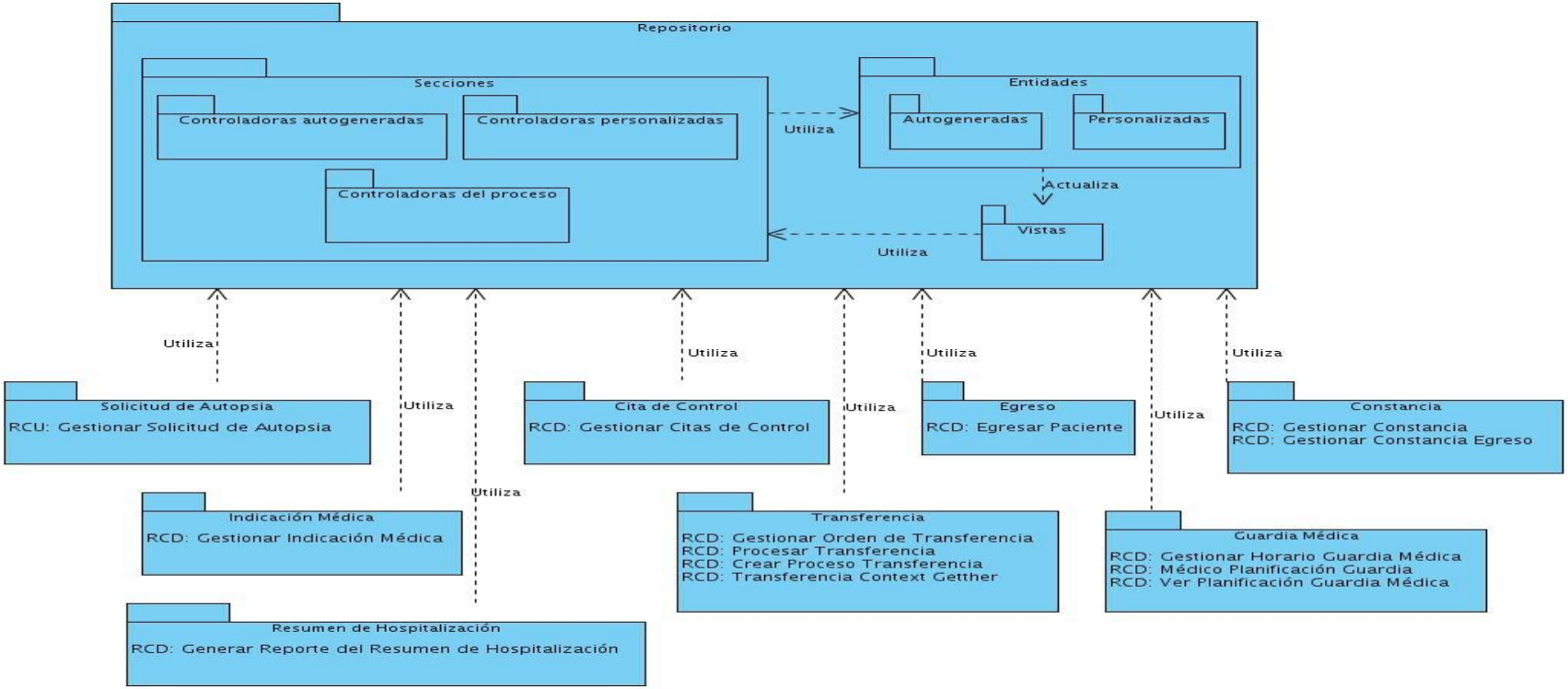

Figura 3.1 Diagrama de paquetes

Dentro de los elementos que componen el Modelo de Diseño se destacan los diagramas de clases del diseño y los diagramas de interacción. Los diagramas de diseño describen gráficamente las características de las clases de software y de las interfaces en un sistema. Mientras que las clases de interacción son aquellas que describen a un grupo de objetos que colaboran entre sí con el objetivo de lograr un objetivo común, evidenciándose además de a los objetos a los mensajes que dichos objetos se envían entre ellos mismos. A continuación se muestran varios diagramas de clases de diseño, además de un conjunto de diagramas de secuencia que evidencian la realización de algunos casos de uso.

3.1.1 Diagramas de clases del diseño

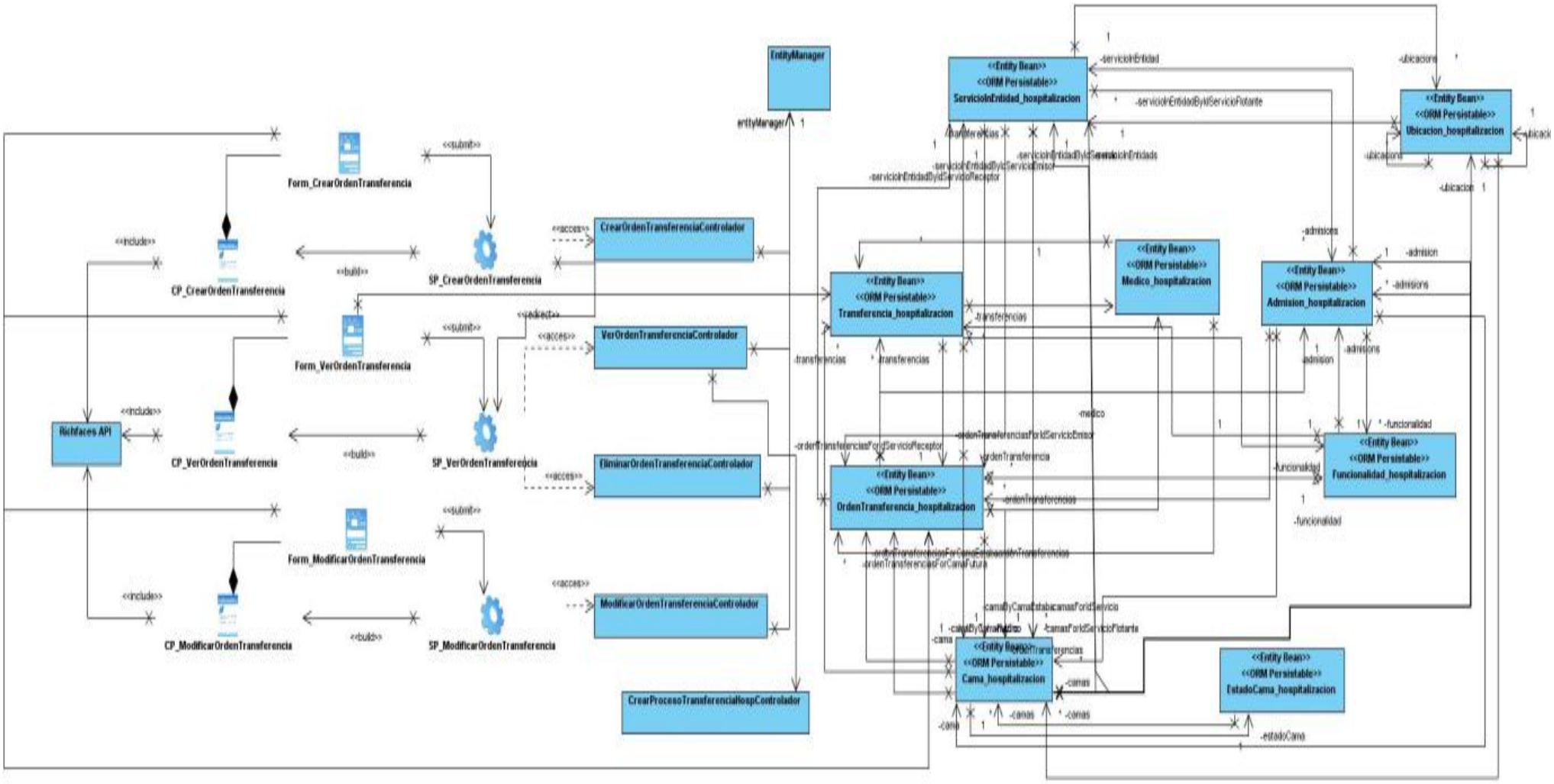

Figura 3.2 Diagrama de CD_GestionarOrdenTransferencia

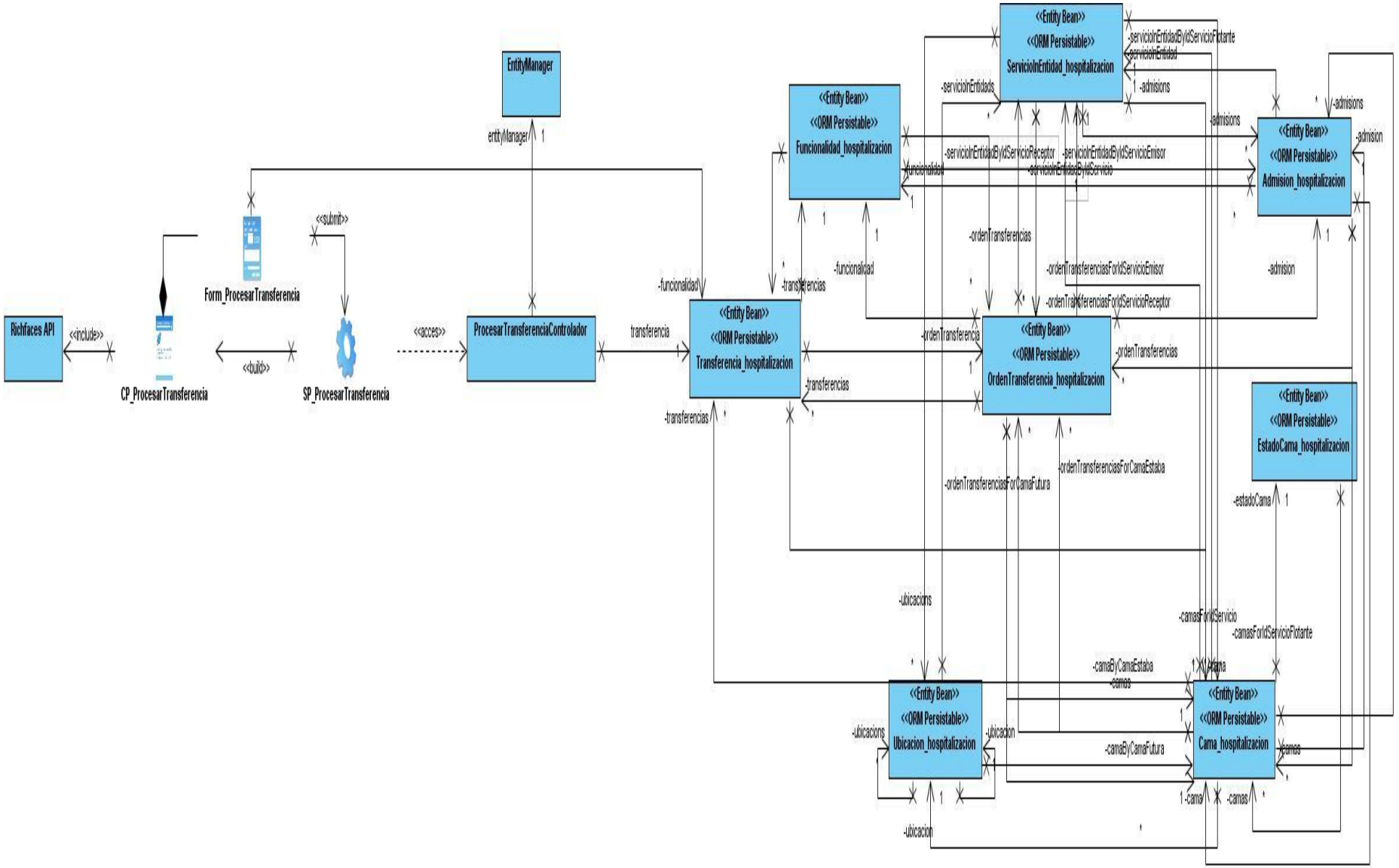

Figura 3.3 DC_ProcesarTransferencia

3.1.2 Diagramas de secuencia

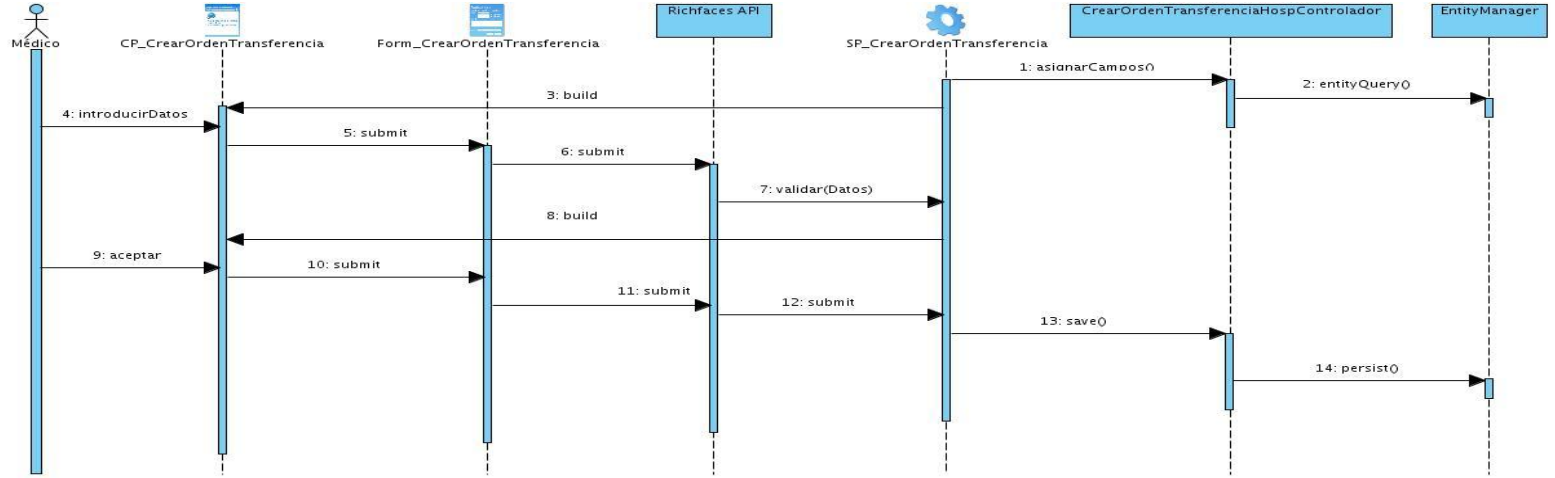

Figura 3.4 DS_CrearOrdenTransferencia

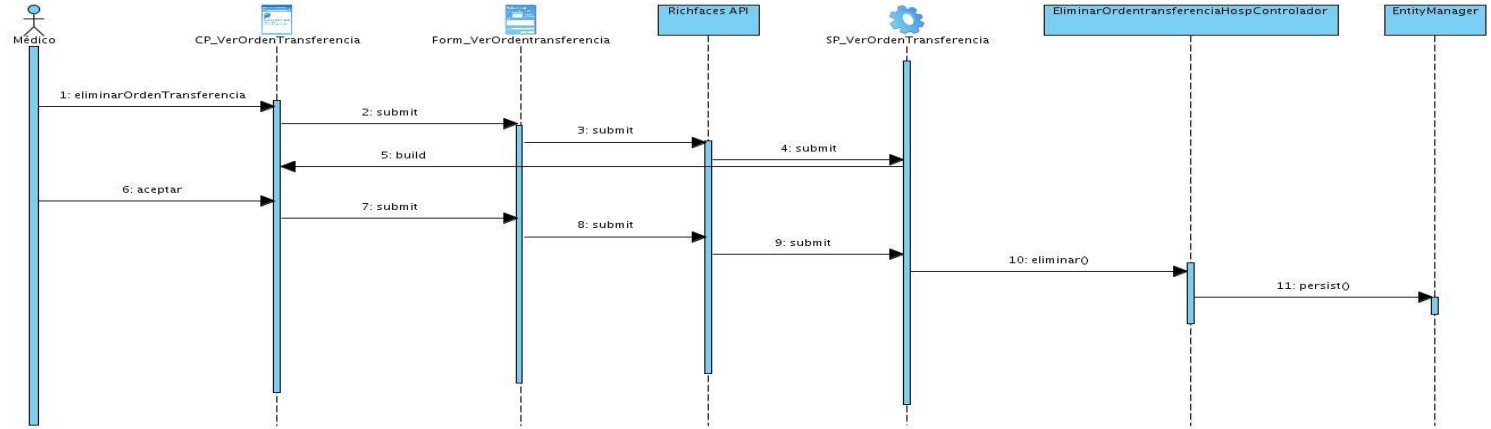

Figura 3.5 DS_EliminarOrdenTransferencia

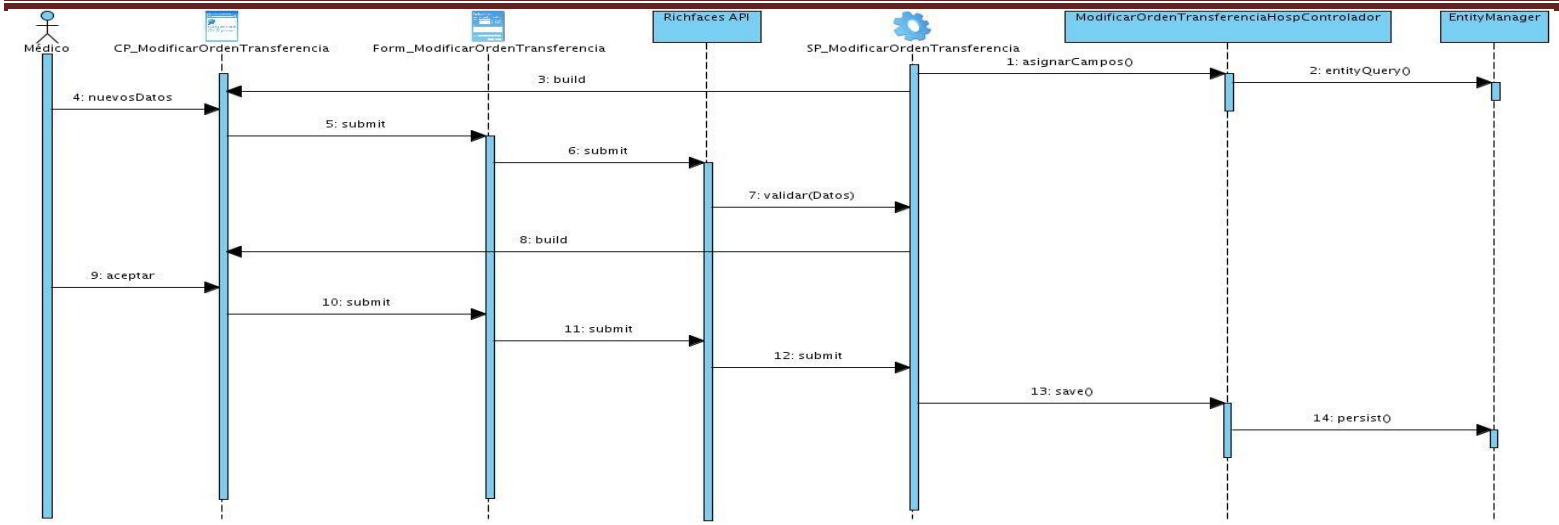

Figura 3.6 DS_ModificarOrdenTransferencia

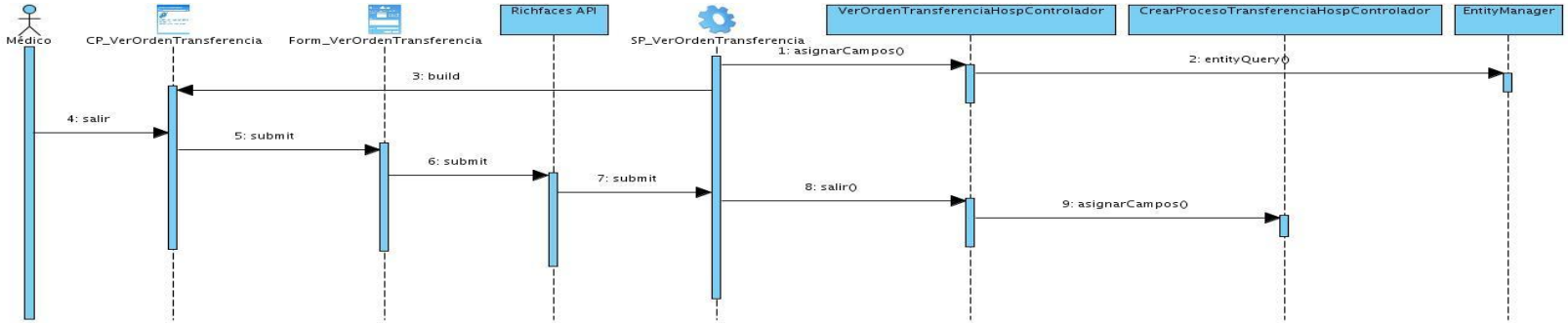

Figura 3.7 DS_VerOrdenTransferencia

3.2 Descripción de las nuevas clases u operaciones necesarias.

Nombre: CrearOrdenTransferenciaControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
medico	Medico_hospitalizacion
servicio	ServicioInEntidad_hospitalizacion
servicioEsta	ServicioInEntidad_hospitalizacion
ubicacion	Ubicacion_hospitalizacion
admision	Admision_hospitalizacion
camaByCamaFutura	Cama_hospitalizacion
camaByCamaEstaba	Cama_hospitalizacion
ordenTransferencia	OrdenTransferencia_hospitalizacion
fechaIngreso	Date
respuesta	String
mostrar	boolean
bloquearUbicaciones	boolean
fueraServicio	boolean
cama	boolean
cid	Integer
tabla	Hashtable<String,Ubicacion_hospitalizacion>
listaUbicaciones	List<String>
entityManager	EntityManager
facesMessages	FacesMessages
bitacora	IBitacora
listarPacientes_hospitalizacion	ListarPacientesControlador
camaHospList_custom	CamaCustomList_hospitalizacion
ubicacionesManager	UbicacionesManager
servicioSeleccionado	String
ubicacionSeleccionada	String
serviciosLista	List<String>
error	int
Para cada responsabilidad:	

Nombre:	asignarCampos()
Descripción:	Es el método que inicializa la clase.
Nombre:	dropEventListener(DropEvent event)
Descripción:	Se utiliza para parar los eventos que están a la espera.
Nombre:	save()
Descripción:	Este método es aquel que se usa para persistir la información en la base de datos.
Nombre:	cerrar()
Descripción:	Sirve para finalizar la clase.
Nombre:	ubicacionesListado()
Descripción:	Brinda el listado de las ubicaciones para un servicio dado.
Nombre:	listadoCamas()
Descripción:	Da como resultado, un listado de camas que están desocupadas, en una determinada ubicación.
Nombre:	seleccionarCama(Cama_hospitalizacion cama)
Descripción:	Se usa para obtener la cama que ha seleccionado el usuario.
Nombre:	setServicio(ServicioInEntidad_hospitalizacion servicio)
Descripción:	Permite asignarle un nuevo servicio al atributo servicio.
Nombre:	restaurar(int opcion)
Descripción:	Restaura varios atributos a sus valores por defecto.

Tabla 3.1 CrearOrdenTransferenciaHospControlador

Nombre: VerOrdenTransferenciaControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
ordenTransferencia	OrdenTransferencia_hospitalizacion
ubicacionesManager	UbicacionesManager
crearProcesoTransferenciaHosp	CrearProcesoTransferenciaHospControlador
id	int
cid	Integer
ubicacion	String
fueraServicio	Boolean
listarPacientes_hospitalizacion	ListarPacientesControlador

activeModule	IActiveModule
entityManager	EntityManager
facesMessages	FacesMessages
bitacora	IBitacora
error	int
conversacionActiva	Boolean
Para cada responsabilidad:	
Nombre:	asignarCampos()
Descripción:	Es el método que inicializa la clase.
Nombre:	salir()
Descripción:	Es el método que da por terminado la clase, y crea el proceso de transferencia.

Tabla 3.2 VerOrdenTransferenciaHospControlador

Nombre: ModificarOrdenTransferenciaControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
servicio	ServicioInEntidad_hospitalizacion
ubicacion	Ubicacion_hospitalizacion
camaByCamaFutura	Cama_hospitalizacion
camaByCamaEstaba	Cama_hospitalizacion
ordenTransferencia	OrdenTransferencia_hospitalizacion
fechaIngreso	Date
respuesta	String
mostrar	Boolean
bloquearUbicaciones	boolean
primera	Boolean
esCama	Boolean
fueraServicio	Boolean
cid	Integer
id	int
tabla	Hashtable<String,Ubicacion_hospitalizacion>
listaUbicaciones	List<String>

entityManager	EntityManager
facesMessages	FacesMessages
bitacora	IBitacora
listarPacientes_hospitalizacion	ListarPacientesControlador
camaHospList_custom	CamaCustomList_hospitalizacion
ubicacionesManager	UbicacionesManager
ubicacionSeleccionada	String
servicioSeleccionado	String
serviciosLista	List<String>
error	int
Para cada responsabilidad:	
Nombre:	asignarCampos()
Descripción:	Es el método que inicializa la clase.
Nombre:	dropEventListener(DropEvent event)
Descripción:	Se utiliza para parar los eventos que están a la espera.
Nombre:	save()
Descripción:	Este método es aquel que se usa para persistir la información en la base de datos.
Nombre:	cerrar()
Descripción:	Sirve para finalizar la clase.
Nombre:	ubicacionesListado()
Descripción:	Brinda el listado de las ubicaciones para un servicio dado.
Nombre:	listadoCamas()
Descripción:	Da como resultado, un listado de camas que están desocupadas, en una determinada ubicación.
Nombre:	seleccionarCama(Cama_hospitalizacion cama)
Descripción:	Se usa para obtener la cama que ha seleccionado el usuario.
Nombre:	setServicio(ServicioInEntidad_hospitalizacion servicio)
Descripción:	Permite asignarle un nuevo servicio al atributo servicio.

Tabla 3.3 ModificarOrdenTransferenciaHospControlador

Nombre: EliminarOrdenTransferenciaControlador
Tipo de clase: Controladora

Atributo	**Tipo**
ordenTransferencia	OrdenTransferencia_hospitalizacion
entityManager	EntityManager
facesMessages	FacesMessages
bitacora	IBitacora
cid	Integer
error	int
Para cada responsabilidad:	
Nombre:	eliminar()
Descripción:	Es el método que elimina una orden de transferencia.

Tabla 3.4 EliminarOrdenTransferenciaHospControlador

3.3 Modelo de Datos

Un Modelo de Datos no es más que un conjunto de conceptos que definen la estructura de una base de datos, ya sean los datos, las relaciones entre estos y las restricciones que rigen dichos datos, sirviendo de representación física de las tablas en una base de datos determinada. Uno de los aportes que brindan los modelos de datos, es el hecho de ofrecer la base conceptual para llevar a cabo el diseño de los sistemas que trabajan con bases de datos. Además cabe destacar que poseen un conjunto de operaciones básicas para la realización de consultas y actualización de los datos. [44]

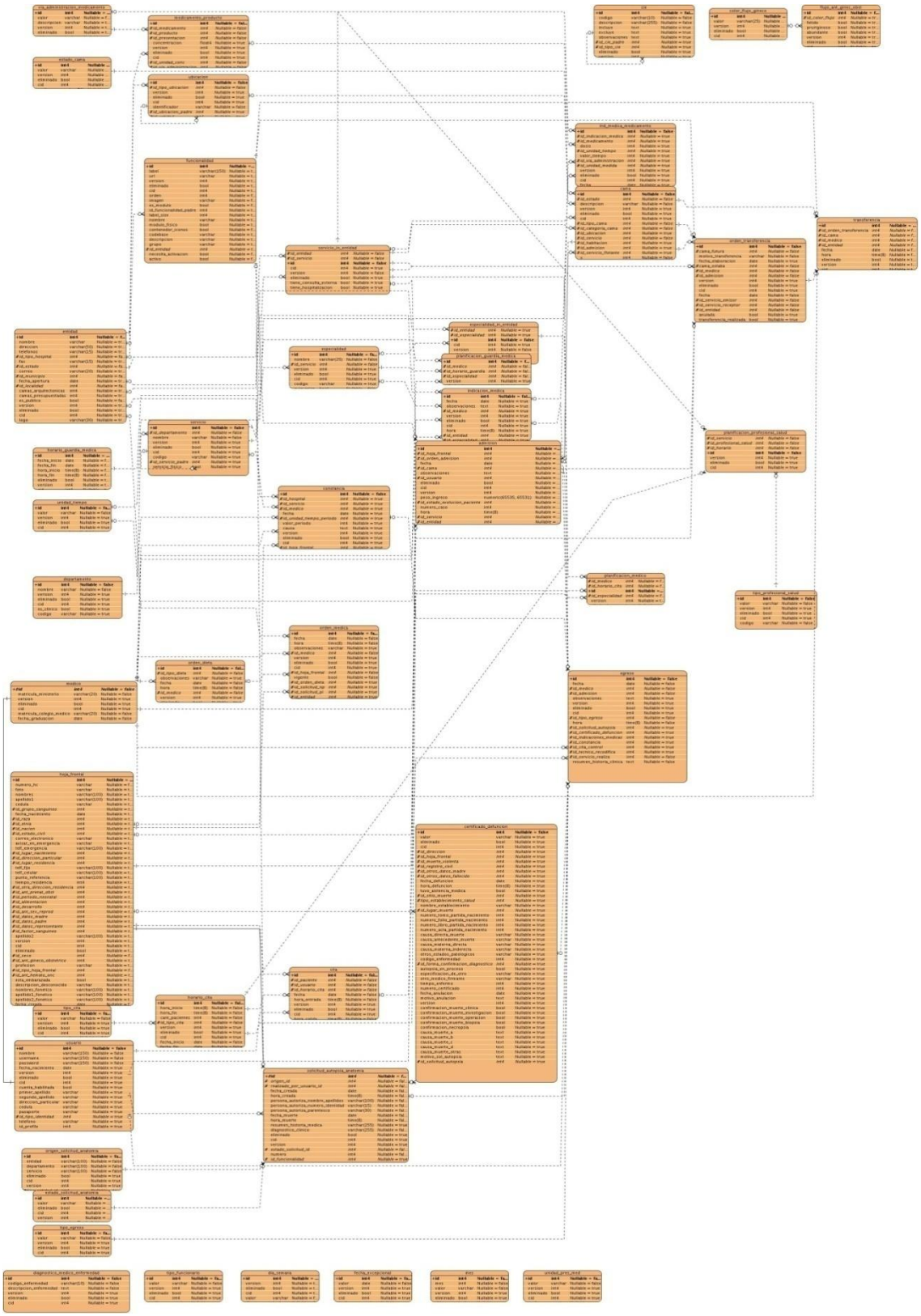

Figura 3.8 Modelo de Datos.

3.4 Valoración de las técnicas de validación

Normalmente al realizarse el desarrollo de cualquier aplicación, se lleva a cabo varias validaciones a los datos que son introducidos por los usuarios del sistema. Con el objetivo de facilitar este tipo de trabajo se propone la utilización del framework JSF, el cual permite definir las validaciones en la vista, pero esto trae consigo el hecho de que se repiten las validaciones una y otra vez. Este inconveniente se puede solucionar gracias a la utilización de Hibernate Validator, al definirse las validaciones en los objetos del negocio, y de esta forma dichas validaciones pueden ser llamadas en el momento necesario.

3.5 Descripción de las tablas

Nombre: admisión		
Descripción: Tabla para registrar los datos de admisión del paciente.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla (PK).
id_hoja_frontal	integer	Identificador de la tabla hoja_frontal (FK).
id_orden_admision	integer	Identificador de la tabla orden_admision (FK).
fecha	date	Fecha en que se llevo a cabo la admisión del paciente.
id_cama	integer	Identificador de la tabla cama (FK)
observaciones	text	Información sobre las observaciones realizadas al paciente.
id_usuario	integer	Identificador de la tabla usuario (FK).
eliminado	boolean	Permite verificar si la admisión ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.
version	integer	Atributo para persistir o actualizar la entidad.

peso_ingreso	numeric	Peso del paciente al ser ingresado.
id_estado_evolucion_paciente	integer	Identificador de la tabla estado_evolucion_paciente (FK).
numero_caso	integer	Número del caso médico.
hora	time(0)	Hora en que se llevó a cabo la admisión.
id_servicio	integer	Identificador de la tabla servicio_in_entidad (FK).
id_entidad	integer	Identificador de la tabla funcionalidad (FK).

Tabla 3.5 Admisión

Nombre: egreso		
Descripción: Tabla para registrar los datos del egreso del paciente.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla (PK).
fecha	date	Fecha del egreso.
id_medico	integer	Identificador de la tabla medico (FK).
id_admision	integer	Identificador de la tabla admision (FK).
observaciones	text	Observaciones sobre el paciente.
id_diagnostico	integer	Identificador de la tabla diagnostico_medico (FK).
version	integer	Atributo para persistir o actualizar la entidad.
eliminado	boolean	Permite verificar el egreso ha sido eliminado.
cid	integer	Identificador de modificaciones registradas en la bitácora.
id_tipo_egreso	integer	Identificador de la tabla tipo_egreso (FK).
hora	time(0)	Hora en que se realizó el egreso.
id_solicitud_autopsia	integer	Identificador de la tabla

		solicitud_autopsia_anatomia (FK).
id_certificado_defuncion	integer	Identificador de la tabla certificado_defuncion (FK).
id_indicaciones_medicas	integer	Identificador de la tabla indicacion_medica (FK).
id_constancia	integer	Identificador de la tabla constancia (FK).
id_cita_control	integer	Identificador de la tabla cita (FK).
id_diagnostico_princ_final	integer	Identificador de la tabla diagnostico_medico (FK).
id_tecnico_recodifica	integer	Identificador de la tabla usuario (FK).
id_servicio_realiza	integer	Identificador de la tabla servivio_in_entidad (FK).
resumen_historia_clinica	text	Resumen de la historia clínica del paciente.
id_registro_fallecimiento	integer	Identificador de la tabla fallecimiento (FK).

Tabla 3.6 Egreso

Nombre: orden_transferencia		
Descripción: Tabla para registrar los datos de la orden de transferencia.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla (PK).
cama_futura	integer	Identificador de la tabla cama (FK).
motivo_transferencia	varchar	Motivo por el cual se llevará a cabo la transferencia.
fecha_elaboracion	date	Fecha en que se elaboró la transferencia.
cama_estaba	integer	Identificador de la tabla cama (FK).
id_medico	integer	Identificador de la tabla medico (FK).
id_admision	integer	Identificador de la tabla admision (FK).
version	integer	Atributo para persistir o actualizar la entidad.

eliminado	boolean	Permite verificar si la orden de transferencia ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.
fecha	date	Fecha en que se creó la orden de transferencia.
id_servicio_emisor	integer	Identificador de la tabla servivio_in_entidad (FK).
is_servicio_receptor	integer	Identificador de la tabla servivio_in_entidad (FK).
id_entidad	integer	Identificador de la tabla entidad (FK).
anulada	boolean	Permite verificar si la orden de transferencia ha sido anulada.
transferencia_realizada	boolean	Permite verificar si la transferencia ha sido realizada.

Tabla 3.7 Orden de transferencia

Nombre: transferencia		
Descripción: Tabla para registrar los datos de la transferencia.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla (PK).
id_orden_transferencia	integer	Identificador de la tabla orden_transferencia (FK).
id_cama	integer	Identificador de la tabla cama (FK).
id_medico	integer	Identificador de la tabla medico (FK).
id_entidad	integer	Identificador de la tabla funcionalidad (FK).
fecha	date	Fecha en que se realizó la transferencia.
hora	time(0)	Hora en que se realizó la transferencia.
eliminado	boolean	Permite verificar si la orden de transferencia ha sido eliminada.
version	integer	Atributo para persistir o actualizar la

		entidad.
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 3.8 Transferencia

3.6 Vista de Implementación

La vista de implementación es aquella describe la organización del sistema en capas y subsistemas de implementación, describiendo la forma en que se realiza la asignación de paquetes y clases existentes en la vista lógica a los paquetes y módulos pertenecientes a la implementación, contándose entre uno de sus artefactos el diagrama de componentes.

3.6.1 Diagrama de componentes

Estos diagramas son utilizados con el objetivo de representar los aspectos de orden físico de un sistema, dichos diagramas poseen un grupo de elementos físicos que residen en un nodo, los cuales pueden ser: bibliotecas, documentos, archivos o ejecutables. Cabe destacar que este tipo de diagramas están compuestos por:

- Componentes
- Interfaces
- Relaciones de dependencia, generalización, asociación, realización.

Por lo tanto, a continuación se expone el modelo de componentes definido en la solución propuesta.

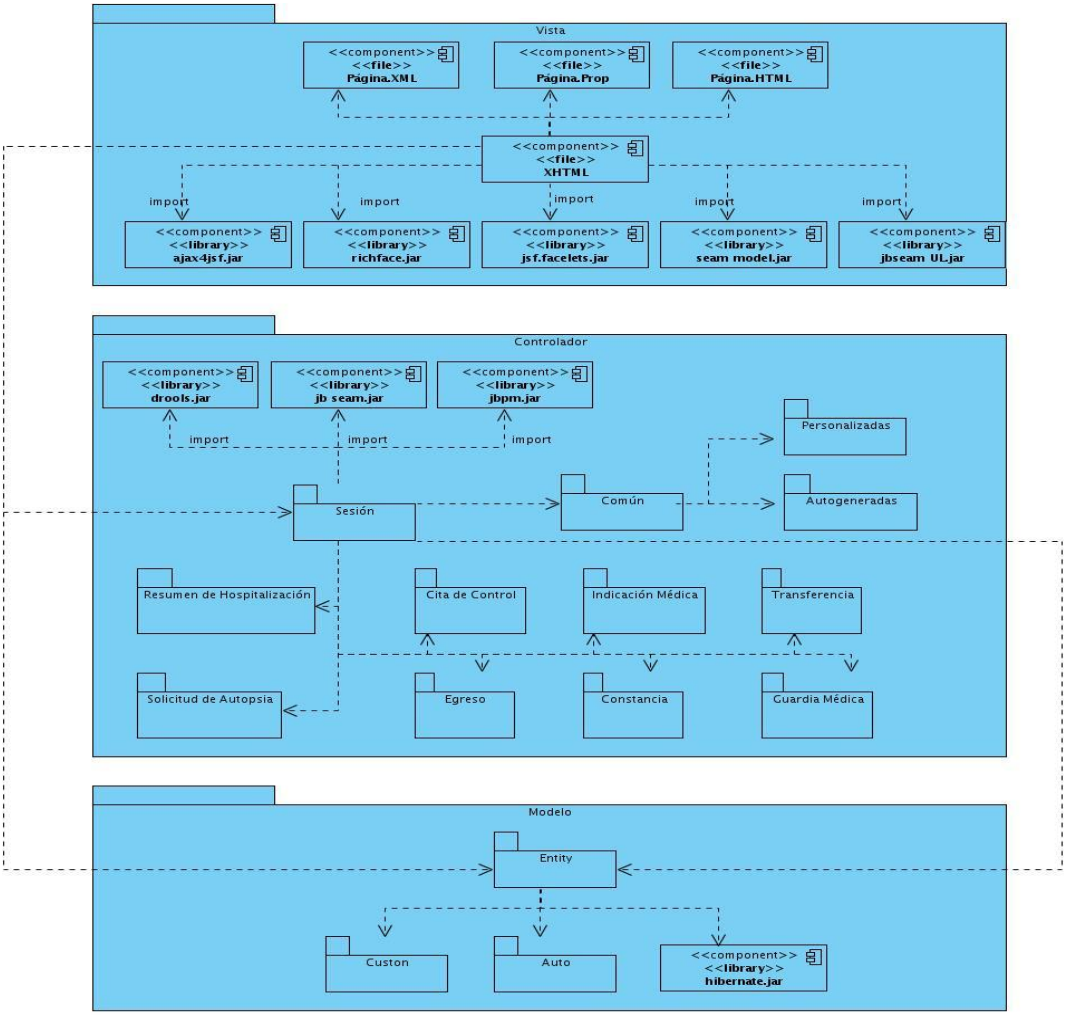

Figura 3.9 Modelo de Componentes.

3.7 Tratamiento de excepciones

Es necesario que es sistema sea capaz de lidiar con errores que se puedan dar en su ejecución, tal es el motivo de la utilización de las excepciones, las cuales se utilizan en varias regiones del código fuente, pero fundamentalmente en aquellas en donde se lleva a cabo la persistencia de la información. En algunos casos es necesario que al ocurrir una excepción, el sistema lleve a cabo un re-direccionamiento hacia otra página, lo cual es posible mediante la manipulación de los archivos cuya extensión es ".pages.xml". Con el objetivo de facilitar dicho trabajo con excepciones se hace uso del componente FacesMessages, el cual es parte del framework seam, dicho componente se encarga de la visualización los mensajes de error a través del objeto facesMessages, el cual se encuentra en las clases controladoras.

CAPITULO III: DESCRIPCION Y ANALISIS DE LA SOLUCION PROPUESTA

En el capítulo presente se llevó a cabo un análisis sobre los aspectos de mayor importancia del diseño e implementación del sistema. Se realizó una valoración del diseño propuesto por el analista del sistema, una descripción de varias clases controladoras, así como diagramas de clases de diseño y de secuencia, con el objetivo de brindar una mayor comprensión de la solución propuesta. Además de exponerse el Modelo de Datos y algunas descripciones de varias tablas de la base de datos, que son vitales para la aplicación.

CAPITULO IV: MODELO DE PRUEBAS

Introducción

A continuación en el presente capítulo se expondrá el Modelo de Prueba. Para dar cumplimiento a dicho objetivo, se realizará una caracterización de las pruebas de caja negra, al ser este tipo de método el que se ha definido a utilizar, definiéndose y describiéndose los casos de pruebas definidos.

Las pruebas de software constituye uno de los flujos de mayor importancia dentro de la ingeniería de software, gracias a estos procesos es que es posible la verificación de la calidad de los sistemas, al poder identificar fallos en la aplicación, ya sean de usabilidad o de implementación. Es de vital importancia para cualquier sistema, la ejecución de pruebas dentro de cada fase del desarrollo del producto, logrando de esta forma un sistema con más nivel de calidad, por lo que tendrá un mayor grado de aceptación en el mercado. Las pruebas de software se pueden dividir en dos tipos, las pruebas de caja blanca o de caja negra.

4.1 Pruebas de caja negra

Las pruebas de caja negra son aquellas que se centran en revisar las funcionalidades del sistema. Es decir, se lleva a cabo una revisión completa del sistema en búsqueda de diferentes tipos de errores, los cuales no son visibles al utilizarse pruebas de caja blanca. Entre los diferentes errores que los métodos de caja negra son capaces de detecta se encuentran:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Figura 4.1 Enfoque de diseño de pruebas de caja negra.

Con el objetivo de llevar a cabo este tipo de pruebas, se utilizan varias técnicas, entre ellas, encontrándose las que se exponen a continuación:

- Técnica de la partición de equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del análisis de valores límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de grafos de causa-efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica ha sido escogida para realizar estas pruebas es la de partición de equivalencia, la cual consiste en dividir el dominio de entrada de una aplicación en clases de datos, derivando a partir de los mismos los casos de prueba, representando cada una de estas clases de equivalencias un grupo de estados válidos e inválidos para las condiciones de entrada.

Un caso de prueba no es más que un conjunto de condiciones, bajo las cuales se introducen datos con el objetivo de obtener varios resultados, permitiendo determinar si se ha cumplido satisfactoriamente el desarrollo de las funcionalidad que se ha estado probando, cabe destacar que este tipo de pruebas no repara en la estructura interna del software, sino que el mismo realice la tarea para la cual fue diseñado. Se puede saber si un caso de prueba es aceptable, si el mismo presenta una alta probabilidad de detectar un error que no haya sido encontrado hasta el momento.

4.1.1 Descripción de los casos de prueba

Escenarios del crear Orden de transferencia	Descripción de la funcionalidad	Flujo Central
EC 1: Crear Orden de transferencia	Crear una Orden de transferencia satisfactoriamente.	Se selecciona la opción Crear Orden de transferencia. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se crea la Orden de transferencia. Muestra la interfaz Ver detalles de Orden de transferencia. Ver caso de prueba: **Ver detalles de Orden de transferencia**.
EC 2: Cancelar operación	Cancelar la opción de Crear Orden de transferencia.	Se selecciona la opción Crear Orden de transferencia. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de	Se selecciona la opción Crear Orden de transferencia.

	haber incompletos muestra un indicador sobre los campos incompletos.	Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.
EC 4: Sin camas disponibles	Luego de haber seleccionado el servicio receptor, el sistema consulta la disponibilidad de camas, de no existir camas disponibles muestra un mensaje de información.	Se selecciona la opción Crear Orden de transferencia. Se introduce el servicio receptor. Muestra un mensaje de información y permite continuar creando la Orden de transferencia.
EC 5: Cambio de servicio en cama actual	Crea una Orden de transferencia aunque no existan camas disponibles, manteniendo la cama actual del paciente pero asignándola al servicio receptor, quedando el paciente fuera de servicio.	Se selecciona la opción Crear Orden de transferencia. Se introducen los datos. Se selecciona la opción Aceptar. Muestra un mensaje de información y brinda la posibilidad de Aceptar crear la Orden de transferencia en la cama actual del paciente.
EC 6: Servicio incompleto	Luego de haber introducido los datos, el sistema los verifica y valida, de no haber introducido el servicio muestra un mensaje de información.	Se selecciona la opción Crear Orden de transferencia. Se introducen los datos sin el servicio. Se selecciona la opción Aceptar. Muestra un mensaje de información.

EC 7: Paciente no encontrado	Luego de haber introducido los datos, el sistema los verifica y valida, de haber egresado o transferido al paciente que se le creará la Orden de transferencia, se muestra un mensaje de información.	Se selecciona la opción Crear Orden de transferencia. Se introducen los datos incorrectos. Se selecciona la opción Aceptar. Muestra un mensaje de información.

Tabla 4.1 CP Crear Orden de Transferencia

Id del escenario	EC 1	EC	EC	EC	EC	EC	EC 7
Escenario	Crear Orden de transferencia satisfactoriamente	Cancelar operación	Existen datos incompletos	Sin camas disponibles	Cambio de servicio en cama actual	Servicio incompleto	Paciente no encontrado
Variable 1 (Motivo de transferencia) Editable	V		I				
Variable 2 (Servicio receptor)						I	
Variable 3 (Sala destino)							

Variable 4 (Cama destino)							
Botón 1 (Aceptar)	NA		NA				
Botón 1 (Cancelar)		NA					
Botón 2							
Respuesta del Sistema	Crea la Orden de transferencia y se muestra la pantalla ver detalles.	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.	Muestra el mensaje: "No hay camas disponibles para el servicio solicitado"	Muestra el mensaje "Se creará la orden de transferencia con la cama que posee actualmente. ¿Desea continuar?" Se selecciona Aceptar y se muestra la	Muestra el mensaje: "Debe seleccionar un servicio"	Muestra un mensaje de información: "El paciente no se encuentra en el servicio.". **Forma de probar el escenario:** 1ro: Un probador debe ejecutar el

					pantalla ver detalles		escenario EC 1 del presente DCP para los datos correspondientes. 2do: Otro probador debe ejecutar el mismo escenario EC 1 del presente DCP, antes de que el probador del caso 1ro seleccione Aceptar. 3ro: Otro probador debe ejecutar el EC 1 del DCP **Procesar Orden de transferencia**

							aceptando la transferencia solicitada por el 2do probador, de forma tal que el paciente que el probador del paso 1ro le está creando la Orden cambie de servicio. 4to: El primer probador Acepta crear la Orden de transferencia.
Resultado de la Prueba							

Tabla 4.2 SC Crear Orden Transferencia

Escenarios del eliminar Orden de transferencia	Descripción de la funcionalidad	Flujo Central
EC 1: Eliminar Orden de transferencia satisfactoriamente	Eliminar una Orden de transferencia.	Se muestra el mensaje: “Se eliminará la Orden de transferencia seleccionada. Al seleccionar Sí se perderán todos los datos. ¿Desea continuar?”. Selecciona la opción Si. Elimina la Orden de transferencia.
EC 2: Cancelar operación	Cancelar la opción de eliminar Orden de transferencia.	Se muestra el mensaje: “Se eliminará la Orden de transferencia seleccionado. Al seleccionar Sí se perderán todos los datos. ¿Desea continuar?”. Se selecciona la opción de No. Se regresa a la vista anterior.

Tabla 4.3 CP Eliminar Orden Transferencia

Id del escenario	EC 1	EC 2
Escenario	Eliminar Orden de transferencia	Cancelar operación

	satisfactoriamente	
(Si)	NA	
(No)		NA
Respuesta del Sistema	Elimina la Orden de transferencia.	Regresa a la vista anterior.
Resultado de la Prueba		

Tabla 4.4 SC Eliminar Orden Transferencia

Conclusiones

En el presente capítulo, ha quedado plasmado que estrategias se definieron con el objetivo de constatar que el sistema que se obtenga como resultado de la presente investigación, sea una aplicación robusta y confiable. Mostrándose al mismo tiempo algunos ejemplos de casos de pruebas que fueron utilizados.

CONCLUSIONES

Una vez desarrollada la presente investigación se concluye lo siguiente:

- Los sistemas analizados no cumplen en su totalidad con los requerimientos funcionales esperados. Además que los mismos han sido desarrollados con herramientas privativas.
- Se determinaron las tecnologías, herramientas y patrones a utilizar, las cuales permitieron la construcción de un sistema robusto y flexible, adaptable a diferentes entornos de despliegue.
- La utilización de pautas para el proceso de desarrollo, garantizó uniformidad y homogeneidad en los artefactos obtenidos a partir de este.
- La utilización de las funcionalidades correspondientes a los procesos del movimiento hospitalario y planificación de la guardia médica, permitirá mejorar la realización de los mismos en el área de hospitalización de las instituciones hospitalarias.
- La ejecución de pruebas de caja negra garantizó la calidad necesaria para la futura explotación y piloto de las funcionalidades implementadas.

RECOMENDACIONES

Se recomiendan los siguientes elementos:

- Implementar nuevas definiciones del proceso de transferencias, donde se consideren otros flujos de actividades y actores para su realización.
- Reutilizar en la implementación del Módulo de Agenda Médica, las funcionalidades asociadas con la gestión de guardias médicas del Módulo de Hospitalización.

REFERENCIAS BIBLIOGRAFICAS

1. Definición de Informática, 2009-11-10, Página Web, Disponible en: http://definicion.de/informatica/
2. Sistema de Información Hospitalaria. México D.F.: Universidad Autónoma de México. D. R. Facultad de Medicina, 2003. p 7. Disponible en: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf
3. Ídem. 2
4. Devesa, Arianna; Muñoz, Reiniel. (2009) Módulo Hospitalización del Sistema de Información Hospitalaria Alas HIS, Facultad 7. Universidad de las ciencias Informáticas
5. Ídem 4
6. Ídem 4
7. Ídem 4
8. Ídem 4
9. García Salabarria, Dr. Joaquín. (2006) ¿Sobrevivirán los niveles de atención a la revolución de la salud pública cubana? [Versión Electrónica]. Revista Cubana de Salud Pública, 32, 0864-3466
10. Ídem 4
11. Ídem 4
12. MetisHis, 2010-2-13, Pagina Web, Disponible en: http://www.apraful.com.uy/soluciones/metis-his.html
13. xHosp - Sistema integral para la administración hospitalaria, 2010-2-13, Pagina Web, Disponible en: http://www.virtus.com.mx/xhosp/index.html
14. Ídem. 13
15. CNT Pacientes HIS Edition, 2010-2-13, Pagina Web, Disponible en: http://cnt.com.co/PAGINA/index.asp?id=178
16. Ídem 15
17. Aplicación de Gestión de Guardias Médicas, 2010-2-13, Pagina Web, Disponible en: http://www.opensistemas.com/soluciones/muestra_caso_exito/caso_exito/8/
18. SOFTEL, Propuesta para la informatización de Centros de Atencion Médica, 2007, Disponible en: http://www.softel.cu/doc/CentrosAtencionMedica.pdf
19. Ídem 18
20. Rojo, J. Oscar. Introducción a los sistemas distribuidos, 2010-2-18, Pagina Web, Disponible en: http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos

21. Microsoft patterns and practices. Model-View-Controller. MSDN. 2009. Disponible en: http://msdn.microsoft.com/en-us/library/ms978748.aspx
22. Ídem 21
23. Ídem 4
24. Facelets. 2010-2-19. Página Web. Disponible en: http://www.synaptic-it.com/blog/?p=8
25. XHTML™ 1.0: El Lenguaje de Etiquetado Hipertextual Extensible. 2010-2-19. Página Web. Disponible en: http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm
26. ¿Qué es JBoss Seam?. 2010-2-20. Página Web. Disponible en: http://es.debugmodeon.com/articulo/que-es-jboss-seam
27. JBoss jBPM. 2010-02-18. Página Web. Disponible en: http://tratandodeentenderlo.blogspot.com/2009/06/jboss-jbpm.html
28. Drools I. Introducción a los motores de reglas de negocios. 2010-02-18. Página Web. Disponible en: http://rincew.blogspot.com/2005/11/drools-i-introduccin-los-motores-de.html
29. Rondon Grados, Luis. JPA - Java Persistence API. 2010-02-19. Página Web. Disponible en: http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html
30. JavaBeans Enterprise. 2010-02-17. Página Web. Disponible en: http://www.programacion.com/java/tutorial/javabeans/
31. Crespo, Cesar. Introducción a Hibernate. 2010-02-17. Página Web. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate
32. Pecos, Daniel. PostGreSQL vs. MySQL. 2010-02-19. Página Web. Disponible en: http://danielpecos.com/docs/mysql_postgres/x15.html
33. Ángel Álvarez, Miquel. Qué es Java. 2010-02-16. Página Web. Disponible en: http://www.desarrolloweb.com/articulos/497.php
34. Ídem 33
35. JBoss Application Server 4.2. 2010-02-15. Página Web. Disponible en: http://www.osalt.com/es/jboss
36. Rational Unified Process (RUP). 2010-02-18. Disponible en: http://www.assembla.com/spaces/isoii2010/documents/aSTUYcXmCr3RlVeJe5aVNr/download/Introducci%C3%B3naRUP.doc
37. González Cornejo, José Enrique. ¿Qué es UML?. 2010-02-14. Página Web. Disponible en: http://www.docirs.cl/uml.htm
38. Ídem 37
39. Introducción a BPMN. 2010-02-17. Página Web. Disponible en: http://www.aprendergratis.com/introduccion-a-bpmn.html

40. Ídem 39
41. Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) 6.0. 2010-02-19. Página Web. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/
42. Descripción de Eclipse SDK 3.3.2. 2010-02-13. Página Web. Disponible en: http://www.boxsoftware.net/programas/eclipse-sdk-3-3-2.asp
43. Ídem 4
44. Marqués Andrés, M. M. Sistemas de Bases de Datos. Página Web. Diusponible en: http://www3.uji.es/~mmarques/f47/apun/node32.html.

BIBLIOGRAFIA

1. Ángel Álvarez, Miquel. Qué es Java. 2010-02-16. Página Web. Disponible en: http://www.desarrolloweb.com/articulos/497.php
2. Aplicación de Gestión de Guardias Médicas, 2010-2-13, Pagina Web, Disponible en: http://www.opensistemas.com/soluciones/muestra_caso_exito/caso_exito/8/
3. Blackshell, bitácora sobre Software Libre, redes, desarrollo y sistemas. 2010-06-04, Página Web, Disponible en: http://blackshell.usebox.net/archivo/297.php
4. CNT Pacientes HIS Edition, 2010-2-13, Pagina Web, Disponible en: http://cnt.com.co/PAGINA/index.asp?id=178
5. Crespo, Cesar. Introducción a Hibernate. 2010-02-17. Página Web. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate
6. Definición de Informática, 2009-11-10, Página Web, Disponible en: http://definicion.de/informatica/
7. Devesa, Arianna; Muñoz, Reiniel. (2009) Módulo Hospitalización del Sistema de Información Hospitalaria Alas HIS, Facultad 7. Universidad de las ciencias Informáticas
8. Descripción de Eclipse SDK 3.3.2. 2010-02-13. Página Web. Disponible en: http://www.boxsoftware.net/programas/eclipse-sdk-3-3-2.asp
9. Drools I. Introducción a los motores de reglas de negocios. 2010-02-18. Página Web. Disponible en: http://rincew.blogspot.com/2005/11/drools-i-introduccin-los-motores-de.html
10. Facelets. 2010-2-19. Página Web. Disponible en: http://www.synaptic-it.com/blog/?p=8
11. García Salabarria, Dr. Joaquín. (2006) ¿Sobrevivirán los niveles de atención a la revolución de la salud pública cubana? [Versión Electrónica]. Revista Cubana de Salud Pública, 32, 0864-3466.
12. González Cornejo, José Enrique. ¿Qué es UML?. 2010-02-14. Página Web. Disponible en: http://www.docirs.cl/uml.htm
13. Introducción a BPMN. 2010-02-17. Página Web. Disponible en: http://www.aprendergratis.com/introduccion-a-bpmn.html
14. JavaBeans Enterprise. 2010-02-17. Página Web. Disponible en: http://www.programacion.com/java/tutorial/javabeans/
15. JBoss Application Server 4.2. 2010-02-15. Página Web. Disponible en: http://www.osalt.com/es/jboss
16. JBoss jBPM. 2010-02-18. Página Web. Disponible en: http://tratandodeentenderlo.blogspot.com/2009/06/jboss-jbpm.html

17. Marqués Andrés, M. M. Sistemas de Bases de Datos. Página Web. Diusponible en: http://www3.uji.es/~mmarques/f47/apun/node32.html.
18. MetisHis, 2010-2-13, Pagina Web, Disponible en: http://www.apraful.com.uy/soluciones/metis-his.html
19. Microsoft patterns and practices. Model-View-Controller. MSDN. 2009. Disponible en: http://msdn.microsoft.com/en-us/library/ms978748.aspx
20. Obregón Martin, Liuvyn. IH-SW-DR-090 ALAS-HIS_Requerimientos suplementarios. UCI : s.n., 2008
21. Pecos, Daniel. PostGreSQL vs. MySQL. 2010-02-19. Página Web. Disponible en: http://danielpecos.com/docs/mysql_postgres/x15.html
22. Pressman, Roger S. Ingeniería de software, un enfoque práctico. 2005
23. ¿Qué es JBoss Seam?. 2010-2-20. Página Web. Disponible en: http://es.debugmodeon.com/articulo/que-es-jboss-seam
24. Rational Unified Process (RUP). 2010-02-18. Disponible en: http://www.assembla.com/spaces/isoii2010/documents/aSTUYcXmCr3RlVeJe5aVNr/download/Introducci%C3%B3naRUP.doc
25. Rojo, J. Oscar. Introducción a los sistemas distribuidos, 2010-2-18, Pagina Web, Disponible en: http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos
26. Rondon Grados, Luis. JPA - Java Persistence API. 2010-02-19. Página Web. Disponible en: http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html
27. Sistema de Información Hospitalaria. México D.F.: Universidad Autónoma de México. D. R. Facultad de Medicina, 2003. p 7. Disponible en: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf
28. SOFTEL, Propuesta para la informatización de Centros de Atencion Médica, 2007, Disponible en: http://www.softel.cu/doc/CentrosAtencionMedica.pdf
29. Velázquez Carralero, Alejandro Mario. IH-SW-DR-091 ALAS-HIS_Documento de Arquitectura del Sistema. UCI : s.n., 2008
30. Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) 6.0. 2010-02-19. Página Web. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/
31. xHosp - Sistema integral para la administración hospitalaria, 2010-2-13, Pagina Web, Disponible en: http://www.virtus.com.mx/xhosp/index.html

32. XHTML™ 1.0: El Lenguaje de Etiquetado Hipertextual Extensible. 2010-2-19. Página Web. Disponible en: http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm

GLOSARIO DE TERMINOS

AJAX: Técnica para el desarrollo Web que posibilita la creación de aplicaciones interactivas.

API (Application Programming Interface): Conjunto de funciones y procedimientos que poseen algunas librerías con el objetivo de ser utilizadas por otro software como una capa de abstracción.

Bean: Componente de un software, que tiene la particularidad de ser reutilizable.

CRUD: Es el acrónimo de Crear, Obtener, Actualizar y Borrar (Create, Retrieve, Update y Delete en inglés). Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un sistema de software.

Dirección IP: es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz de un dispositivo dentro de una red que utilice el protocolo IP (Internet Protocol).

Egreso: Movimiento hospitalario que se lleva a cabo cuando el paciente haya recuperado su salud, fallecido o se lleve a cabo contra indicación médica. Además al realizarse el egreso, se genera un conjunto de documentos, en función al tipo de egreso realizado.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

GRASP: Son patrones generales de software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns".

Guardia médica: Actividad realizada por el personal médico del área de hospitalización, consiste en distribuir al personal médico de forma tal que se garantice la presencia de al menos un galeno en cada servicio, el cual estará al tanto de la condición médica de los pacientes.

Hospitalización: Área de las instituciones hospitalarias, en la cual un paciente es admitido cuando su estado de salud requiere de un monitoreo constante por parte de personal de salud de dicha área.

HTML: Siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

ICEFaces: Biblioteca de componentes JSF Ajax.

JavaScript: Lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web.

Médico: Profesional de la salud que se encarga de emitir un diagnóstico sobre la condición sanitaria del paciente y suministrarle el tratamiento adecuado, además de monitorear la condición sanitaria del mismo mientras se encuentre en el área.

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

Servicio: Servicio del área de Hospitalización en el cual el paciente fue admitido, al coincidir su patología con el área de atención en el cual está especializado el servicio en cuestión.

Transferencia: Movimiento hospitalario que se lleva a cabo cuando el paciente presenta problemas de salud que no se pueden solucionar en el servicio en el cual se encuentra y es necesario trasladarlo a otro servicio, cabe destacar que es posible la realización de una transferencia que tenga como destino el mismo servicio en el cual se encuentra el paciente.

UI: Interfaz de usuario.

XML (Extensible Markup Language): Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium.