Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Implementación de las hojas de anestesia estándar y cardiovascular del Quirófano del Sistema de Información Hospitalaria alas HIS

Autores: Carlos A. Montano Guevara

Roberto Rovira Roble

Tutores: Ing. Yoandy González Martínez

Ing. Vladimir Guerra Miller

Ciudad de la Habana, Julio de 2010

"Año del 52 de la Revolución"

DATOS DE CONTACTOS:

Tutores:

Ing. Yoandy González Martínez, Instructor recién graduado en el año 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Profesor de la Facultad # 7. Ha impartido las asignaturas de Gráficos por Computadoras y Preparación para la Prueba de Nivel de Programación. Se ha desempeñado como jefe del módulo Bloque Quirúrgico en el área temática de gestión hospitalaria.

Correo electrónico: ygonzalezm@uci.cu

Ing. Vladimir Guerra Miller, graduado en el año 2008 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Profesor de la Facultad # 7. Ha impartido las asignaturas de Algoritmización, Introducción a la programación y práctica profesional. Se ha desempeñado como jefe del módulo Visor de Historia Clínica en el área temática de gestión hospitalaria.

Correo electrónico: vguerra@uci.cu

AGRADECIMIENTOS:

De Carlos:

Quiero agradecer especialmente a mis padres por su apoyo y educación en todo momento de mi vida, guiarme hacia el camino correcto, ofrecerme amor, confianza y dedicación. Sin ellos no hubiese podido alcanzar esta meta.

A mi hermana.

A toda mi familia que me ha impulsado a ser cada día mejor.

A mi novia por su comprensión y apoyo incondicional.

A mis amigos que me han apoyado en momentos malos y buenos.

A los tutores y profesores del proyecto por toda la ayuda.

A mi compañero de tesis que siempre supo sobreponerse a las dificultades de la tesis.

De Roberto:

A mis padres por ser quienes me han guiado por el camino correcto y por depositar tanta confianza en mi.
A mis abuelos y tíos por estar siempre a mi lado cuando los necesito.

A mis primos, especialmente a Gaby por ser como un hermano para mí.

A mi novia por su amor, y por estar a mi lado en los momentos buenos y malos.

Al piquete querido de amigos de Chaparra:el Pabli, el Chuchi, el Isra, el Yuset y el Javy por regalarme su amistad y estar siempre ahí cualesquiera que sean los tiempos.

A mis compañeros de Universidad por haber compartido con ellos tantos momentos inolvidables durante estos años.

A mi compañero de tesis por su dedicación, y por hacer junto conmigo que este trabajo saliera adelante.

A mis tutores y profesores que contribuyeron a la realización de este trabajo.

DEDICATORIA:

De Carlos:

A mis padres por todo el esfuerzo, cariño y comprensión que siempre me han dado, que gracias a ellos estoy cumpliendo un sueño.

A mi hermana por su apoyo y cariño.

A mi novia por su amor.

A mis abuelos, abuelas, tíos y tías que me han ayudado y brindado su mano en todo momento.

De Roberto

A mi padre porque sé que estuvo orgulloso de mi y lo estará por siempre donde quiera que se encuentre, por ser mi meta, mi guía y por inculcarme esta educación por lo cual siempre le voy a estar agradecido.

A mi madre por el amor que cada día me profesa, por su apoyo, por su comprensión, por sus consejos, por ser aquella mano sabia que me levanta cuando estoy caído.

A mi hermana por quererme tanto, y por regalarme tanta felicidad cuando estoy a su lado.

A mi abuela Yeya por enseñarme tantas cosas de la vida y ser para mi segunda madre.

RESUMEN:

Actualmente muchas instituciones hospitalarias no cuentan con un sistema que gestione todo el flujo de información que en estas se genera, afectando así la capacidad de respuesta y la calidad de los servicios que se brinda a los pacientes. El objetivo de este trabajo es agregar los procesos de las hojas de anestesia estándar y cardiovascular del módulo Bloque Quirúrgico al Sistema de Información Hospitalaria alas HIS desarrollado en la Universidad de las Ciencias Informáticas.

Para su realización se utilizaron las herramientas establecidas por el Departamento de Sistema de Información Hospitalaria. Para el análisis y diseño se usó el Proceso Unificado de Desarrollo (RUP), metodología basada en el Lenguaje Unificado de Modelado (UML). Como lenguaje de programación orientado a objetos del lado del servidor Java, Eclipse como Entorno de Desarrollo Integrado, PostgreSQL 8.3 como Sistema Gestor de Bases de Datos, así como el uso de Hibernate como herramienta de mapeo relacional para la persistencia de los datos y el framework Seam para la lógica del negocio, entre otras tecnologías.

Con la agregación de estas funcionalidades al módulo de Bloque Quirúrgico se espera agilizar el proceso de atención al paciente a partir de los beneficios que ofrecen las tecnologías de la información. Ya que se podrán controlar los datos del paciente antes, durante y después del acto quirúrgico, lo que favorecerá el conocimiento del médico. Así como a reducir los costos de inversión que podría tener el país si adquiere un sistema de este tipo.

ÍNDICE

INTRODUCCIÓN *1*

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA *1*

1.1 CONCEPTOS BÁSICOS ASOCIADOS AL DOMINIO DEL PROBLEMA 1
1.2 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN 2
1.3 TENDENCIAS Y TECNOLOGÍAS ACTUALES A CONSIDERAR 7

CAPÍTULO 2: DESCRIPCIÓN DE LA ARQUITECTURA *17*

2.1 REQUISITOS NO FUNCIONALES 17
2.2 DESCRIPCIÓN DE LA ARQUITECTURA. FUNDAMENTACIÓN. 20
2.3 ANÁLISIS DE COMPONENTES REHUSADOS 22
2.4 ESPECIFICACIÓN DE LOS TÉRMINOS DE SEGURIDAD 25
2.5 DIAGRAMA DE DESPLIEGUE 26
2.6 ESTRATEGIAS DE CODIFICACIÓN. ESTÁNDARES Y ESTILOS A UTILIZAR 27

CAPITULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA. *34*

3.1 MODELO DE DISEÑO 34
3.2 DIAGRAMAS DE CLASES DEL DISEÑO 37
3.4 MODELO DE DATOS 48
3.5 DESCRIPCIÓN DE LAS TABLAS 50
3.6 VALORACIÓN DE LAS TÉCNICAS DE VALIDACIÓN. 51
3.7 VISTA DE IMPLEMENTACIÓN 52
3.8 DESCRIPCIÓN DE LOS ALGORITMOS NO TRIVIALES A IMPLEMENTAR. ANÁLISIS DE COMPLEJIDAD DE LOS MISMOS 54
3.9 SELECCIÓN DE LAS ESTRUCTURAS DE DATOS APROPIADAS PARA LA IMPLEMENTACIÓN DE ESTOS ALGORITMOS. 54
3.10 DESCRIPCIÓN DE LAS CLASES QUE SE UTILICEN PARA REPRESENTAR COMPUTACIONALMENTE DICHA ESTRUCTURA. .. 55

CAPÍTULO 4: MODELO DE PRUEBA *57*

4.1 PRUEBA DE CAJA NEGRA 57

CONCLUSIONES *62*

RECOMENDACIONES *63*

REFERENCIAS BIBLIOGRÁFICAS *64*

BIBLIOGRAFÍA *66*

Introducción

En pleno siglo XXI el conocimiento es la principal fuente de poder y la computación está estrechamente vinculada a su adquisición y desarrollo. Aparejado a ello existe un vertiginoso avance en las Tecnologías de la Información y las Comunicaciones (TIC) y con ellas la informática, impulsa el desarrollo de aplicaciones que aceleran y modernizan varios sectores de la sociedad.

Uno de los campos de la actividad humana que se ha visto beneficiado por el proceso de informatización es el área de la salud, pues brinda soluciones factibles a la necesidad global de compartir la información y el conocimiento entre el personal de la salud, la comunicación social y divulgación, la producción de publicaciones científicas y técnicas, medios audiovisuales, multimedia y software, programas educativos en televisión, así como servicios de redes de comunicación. (1)

Con el objetivo de agilizar el proceso de atención al paciente y elevar la calidad de los servicios médicos, surgieron en los años 70 los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés), sistemas de información integrados, diseñados para manejar los aspectos administrativos, financieros y clínicos de un hospital (2). Los HIS pueden estar compuestos por una gran variedad de subsistemas que se encargan de gestionar la información en las diferentes áreas del hospital, lo que permite la optimización de los recursos humanos y materiales.

Desde el mismo inicio del triunfo de la Revolución Cubana, una de las estrategias seguidas por el gobierno revolucionario y el Ministerio de Salud Pública (MINSAP), fue el estudio y procesamiento de los hechos vitales y sanitarios dentro del Sistema Nacional de Salud (SNS), inicialmente de forma manual y después con equipos de cómputo. En años posteriores se introdujeron las primeras mini computadoras cubanas y se construyó el primer centro de cálculo en salud pública en el Instituto de Oncología y Radiobiología. (3)

Con el objetivo de garantizar la calidad en la prevención, cuidado, rehabilitación de los pacientes y especializar los servicios de salud, el SNS cubano comprende tres niveles de atención médica organizados en: Atención Médica Primaria, Atención Médica Secundaria y Atención Médica Terciaria. El eje fundamental y centro del proceso de informatización del sector lo constituye el paciente, quien será el principal beneficiado al garantizar las aplicaciones, la calidad, oportunidad y consistencia de la información, lo que incrementará la efectividad y eficiencia de los procesos relacionados con la salud, que

en última instancia gravitarán en un incremento continuo y sostenido de la calidad en la atención médica. (4)

El país comienza a llevar a cabo diversas acciones con el propósito de utilizar los recursos y la información de salud disponible. Lo que permite enlazar a todo el sistema de salud para dar una respuesta más eficiente en la esfera de la información científica a los profesionales, técnicos de la salud y a la situación sanitaria del país. Por lo anteriormente expuesto en 1992 se crea la Red Telemática de la Salud, INFOMED, dentro de la estructura del Centro Nacional de Información de Ciencias Médicas con una acertada visión de la influencia que las Tecnologías de la Información y las Comunicaciones ejercerán en la esfera de la información y el conocimiento.

Durante los últimos años un grupo de instituciones cubanas y el propio MINSAP han desarrollado sistemas encaminados a lograr la informatización de la salud. En todos los casos el objetivo ha sido proveer al SNS de información confiable, consistente y oportuna para la toma de decisiones y el mejoramiento de los procesos médicos asistenciales, garantizando de esta manera el incremento en la calidad y seguridad de la atención médica a la población. Dentro de este grupo de instituciones del Ministerio de la Informática y Comunicaciones que se dedican al desarrollo de dichas aplicaciones básicas para la informatización del sector de la salud; se encuentran empresas como Desoft, Softel, PcMax, Sys, INFOMED, CEDISAP y la Universidad de Ciencias Informáticas (UCI) representada por la empresa Albet.

En el país existen problemas con la gestión de información de los procesos que se realizan tanto para la atención al paciente como para la administración de los recursos. Estos problemas ocurren en algunas instituciones hospitalarias que no automatizan sus procesos o no cuentan con un HIS que integre todas las áreas del hospital, lo que afecta la capacidad de respuesta de este sector a la población atendida. En estas instituciones el proceso manual que se lleva a cabo para la recopilación de información, dificulta el adecuado flujo de la misma. Además, el almacenamiento de la información en formato físico, no brinda la seguridad y conservación de los datos clínicos del paciente, así como también demora la obtención de las historias clínicas que son tan importantes para que los servicios prestados tengan mayor calidad.

Un área importante de las instituciones hospitalarias es el Bloque Quirúrgico, donde son atendidos los pacientes que requieren de un procedimiento quirúrgico para su tratamiento clínico. En esta área funcionan servicios de cirugía, donde son consultados, elaborados y modificados documentos por parte del personal médico en función de la gestión de la información vinculada a los procesos quirúrgicos, entre

los que se encuentran, la historia clínica y la distribución de quirófanos por servicios. Los documentos se gestionan en formato físico, corriendo el riesgo de dañarse o extraviarse, atrasando el funcionamiento de los servicios quirúrgicos. Se llevan a cabo procesos manuales que afectan la estandarización de los datos, debido a problemas de comunicación e imprecisiones que dificultan el adecuado flujo de información dentro del Bloque Quirúrgico y con los departamentos de Consulta Externa, Hospitalización y Emergencia que solicitan los servicios quirúrgicos.

La historia clínica es el documento que refleja la situación y evolución médica de un paciente a lo largo del proceso asistencial. Específicamente en el Bloque Quirúrgico, se anexan a la historia clínica registros tales como, la hoja de anestesia, la hoja de evolución médica y la nota operatoria por lo que la búsqueda de la información contenida en estos registros también se dificulta dada la forma en que se estructura una historia clínica en formato físico. La hoja de anestesia es un documento muy importante en el transcurso del funcionamiento del servicio que brinda el Bloque Quirúrgico, la cual se llena de forma manual. Esta hoja transitará por todos los departamentos del Bloque Quirúrgico por los que haya necesitado pasar el paciente.

El acto quirúrgico supone procedimiento invasivo al organismo y afecta tres niveles Biópsico-Sociales: preoperatorio, transoperatorio y posoperatorio, que no es más que el antes, durante y el después de la intervención quirúrgica. Los datos obtenidos en estos tres niveles así como los obtenidos en la Consulta Preanestésica son registrados en la Hoja de Anestesia garantizando un control del estado del paciente antes, durante y después de la intervención quirúrgica.

El proceso de gestión de información de forma manual sobre la hoja de anestesia implica que: se pierda o deteriore ya que es un documento anexado a la historia clínica y transitará con la misma por todos los departamentos del Bloque Quirúrgico por los que haya transitado el paciente, que no sean legibles los datos que en la misma se registran ya que es llenada por distintos médicos en diferentes consultas, todo lo anterior unido al escaso espacio que existe en los campos de texto para la escritura.

Estos procesos gestionados de manera manual afectan la estandarización de los datos debido a problemas de comunicación e imprecisiones que dificultan el adecuado flujo de la información. Además, la hoja de anestesia está concebida en formato físico y pueden existir retrasos en la obtención de información que se necesite consultar, así como puede provocar deterioro de los formatos o la pérdida de los mismos.

Debido a la necesidad existente de mejorar las condiciones de las instituciones hospitalarias como actores fundamentales del sistema de salud, se puso en práctica la creación de una solución informática integrada al desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs). La UCI y específicamente el Departamento de Gestión Hospitalaria del Centro Especializados en Soluciones de Informática Médica es la encargada de la producción de un HIS, el cual estará compuesto por módulos que representan las diferentes áreas de los hospitales.

Considerando lo analizado anteriormente, sobre la situación actual de los servicios quirúrgicos en los hospitales, se plantea como **Problema a Resolver**: ¿Cómo facilitar la gestión de información en los procesos relacionados con la hoja de anestesia en el área de Bloque Quirúrgico de las instituciones hospitalarias?

Se define como **Objeto de Estudio** de la investigación: Proceso de gestión de información en el Bloque Quirúrgico. El **Campo de Acción** está comprendido en: Proceso de gestión de información en el área de Bloque Quirúrgico específicamente para la hoja de anestesia en las instituciones hospitalarias.

Para resolver el problema identificado se propone el siguiente **Objetivo General**: Desarrollar dentro del módulo Bloque Quirúrgico del Sistema de Información Hospitalaria alas HIS los procesos correspondientes a la hoja de anestesia, que facilite la gestión de información en esta área de las instituciones hospitalarias.

Para dar cumplimiento al objetivo trazado, deben desarrollarse las siguientes **Tareas**:

- Analizar los procesos de negocio asociados al área de Bloque Quirúrgico de las instituciones hospitalarias.
- Evaluar las tendencias actuales en el mundo de los Sistemas de Información Hospitalaria.
- Aplicar las pautas de desarrollo de los entregables y diseño definidas en el Departamento de Sistemas de Gestión Hospitalaria.
- Obtener los artefactos correspondientes a los flujos de trabajo de “Análisis y Diseño”, “Implementación” y “Prueba”.
- Implementar los procesos de negocio relacionados con la hoja de anestesia del módulo de Bloque Quirúrgico.

- Obtener acta de liberación otorgada por calidad interna, de la documentación y de las funcionalidades implementadas de los procesos de negocio relacionados con las hojas de anestesia.

Capítulo 1: Fundamentación teórica

En este capítulo se hace referencia a los principales sistemas automatizados que gestionan la información de los procesos elementales del Bloque Quirúrgico. El estudio de estos sistemas permite valorar las características funcionales y no funcionales de estas soluciones a nivel nacional e internacional. Se realiza además un estudio de las tendencias, tecnologías, herramientas y metodologías actuales, con el fin de proponer las más adecuadas a la solución del problema, a partir de la asimilación de la arquitectura definida por el Departamento de Gestión Hospitalaria del Centro Especializados en Soluciones de Informática Médica.

1.1 Conceptos básicos asociados al dominio del problema

En aras de tener un mejor dominio del marco teórico relacionado con el problema a solucionar, se describen los principales conceptos asociados al campo de acción.

El Bloque Quirúrgico es el área centralizada en la que se genera toda la actividad quirúrgica del hospital. Esta actividad transcurre en la unidad quirúrgica o quirófano que es el sitio en donde se llevan a cabo las intervenciones quirúrgicas electivas y de urgencia por lo que representa un servicio del departamento de cirugía.

Una intervención quirúrgica es un proceso que forma parte de la medicina y tiene por objetivo curar las enfermedades mediante la operación del paciente. La operación no es más que la ejecución de diversos actos curativos sobre el cuerpo vivo, como extirpar, amputar, resecar, implantar, corregir y coser, órganos, miembros o tejidos, con ayuda de un conjunto de productos y utensilios quirúrgicos que se integran como una unidad, denominados kits de cirugía.

La intervención quirúrgica debe realizarse por un cirujano, médico especialista en un servicio quirúrgico, pues son los que están legalmente autorizados a practicar la cirugía. Los cirujanos de los servicios quirúrgicos son los encargados de realizar la solicitud de intervención quirúrgica, documento que recoge los datos generales del paciente, el diagnóstico, además se registra el procedimiento quirúrgico a realizar, se especifica el personal que debería participar y en caso de requerir equipos especiales o implantación de dispositivos, también se especifican.

El anestesiólogo es un especialista, su función principal es suministrarle las drogas al paciente. Una de las tareas es monitorear el estado de salud del paciente antes, durante y después del acto quirúrgico. La

anestesia constituye el producto que se le suministra al paciente para que durante el transcurso de la operación no sienta dolor debido a la falta o privación general o parcial de la sensibilidad artificialmente producida. El anestesiólogo se encarga de crear la Hoja de anestesia que es el documento donde se registra la información a partir de la Consulta Preanestésica y que culmina en el proceso posoperatorio luego de pasar por el preoperatorio y transoperatorio.

La consulta preanestésica debe ser realizada en un local provisto de recursos que permitan la recogida de datos como peso talla, examen físico, entre otros; con suficiente privacidad y brinde un ambiente propicio para que el enfermo se sienta seguro. El médico debe tener una participación activa y registrará en la Hoja anestésica, que acompañará a la historia clínica, todo lo que considere de mayor interés y pudiera repercutir en el transcurso de la anestesia que se seleccione. Se revisará toda la documentación que acompañará a la historia clínica, incluyendo el consentimiento del enfermo. Todo ello ayudará a identificar y valorar riesgos.

El preoperatorio es el período en el cual se le suministra al paciente la anestesia preoperatoria y si no existe ninguna complicación se somete a la cirugía propuesta. El Transoperatorio es el período en el cual es aplicada la anestesia operatoria y se monitorea al paciente constantemente hasta finalizar la cirugía. El posoperatorio es el período de recuperación del paciente donde se le realiza un seguimiento para analizar su evolución mientras abandona los efectos transitorios de la anestesia.

La hoja de anestesia es un documento muy importante en el transcurso del funcionamiento del servicio que brinda el Bloque Quirúrgico. El acto quirúrgico supone un procedimiento invasivo al organismo y afecta tres niveles biópsico- sociales, preoperatorio, transoperatorio, posoperatorio. Todos los datos obtenidos en estos niveles son registrados en la hoja de anestesia, así como los registrados en la Consulta Preanestésica.

1.2 Sistemas automatizados existentes vinculados al campo de acción

1.2.1 Sushrut: C-CAD (Centre for development of advanced computing) Sushrut es un sistema de información hospitalaria (HIS) y se ha desarrollado con el objetivo de racionalizar el flujo de tratamiento de un paciente en el hospital. Presenta una arquitectura cliente-servidor de base de datos. El módulo de Operación contiene información sobre la disponibilidad de todos los quirófanos, equipo y herramientas. La planificación de operaciones es la función principal de este módulo a través de la cual se puede aprobar, cancelar o reprogramar la operación. Se genera un expediente de operación con los principales datos una

vez acabada la misma. Permite la entrada y validación de un registro detallado de la operación, la anestesia y el mantenimiento del posoperatorio. (5)

1.2.2 Care2X: Integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Presenta una arquitectura cliente servidor. Su diseño también puede manejar los servicios no médicos o de funciones tales como la seguridad y mantenimiento. El sistema contiene el módulo quirófanos que se encarga de las siguientes funcionalidades de las salas de operaciones:

- Documentar los procedimientos quirúrgicos (cirugía, anestesia, enfermería, materiales y medicinas)
- Planificador de actividades del quirófano
- Funciones de búsqueda y archivo
- Clasificación internacional de enfermedades
- Planificador de operaciones quirúrgicas (6)

1.2.3 BQO (Bloque Quirúrgico Oftalmológico): Es una aplicación realizada por el Área Temática Gestión Hospitalaria. El sistema permitirá a los usuarios controlar toda la información referente a un paciente que será intervenido quirúrgicamente por causas oftalmológicas, se registran además breves datos referentes a la anestesia a suministrar, antes y después de ser operado. (7)

Presenta una arquitectura cliente servidor. Entre sus principales funciones se encuentran:

- Registrar los datos necesarios del paciente a medida que pasa por las diferentes consultas.
- Visualizar los datos registrados en la historia clínica siempre que se atienda a un paciente.
- Realizar búsquedas de paciente, consultas, informes operatorios y anuncios.
- Registrar los datos necesarios del paciente durante la intervención quirúrgica.
- Crear y modificar el anuncio operatorio.
- Crear y modificar el informe operatorio.

1.2.4 GoWin: Es el paquete de software desarrollado por Valen Computer que está compuesto por todos y cada uno de los módulos necesarios para una gestión sanitaria integral. GoWin Qui es un sistema de información del quirófano que ha sido desarrollado e integrado en el Sistema de Información Hospitalaria (HIS) para poder gestionar todas las funciones que ha de realizar el personal de quirófano, comenzando con la hoja de trabajo, pasando por la citación, las validaciones y finalizando en la hoja quirúrgica, que generará los informes correspondientes.

Algunos módulos de Gowin Qui - Sistema de información del quirófano:

Módulo de hoja quirúrgica: El módulo de la hoja quirúrgica permite almacenar todos los datos que hacen referencia a la intervención quirúrgica, la codificación, registro de enfermería y registro de prótesis.

Módulo de parametrización: El módulo de parametrización permite la configuración del sistema. Se definen las tablas necesarias y se configura el sistema para su funcionamiento según las especificaciones de cada organización. (9)

1.2.5 LCS (Life Care Systems) - Sistema de Información de Gestión Hospitalaria: Se ocupa de la totalidad de las principales áreas funcionales de la moderna especialidad multi-hospital. Proporciona fácil acceso a información para tomar mejores decisiones a tiempo y puede ser fácilmente adaptado a las necesidades de cualquier hospital. Los detalles de preanestesia, tipo de anestesia, las investigaciones necesarias, de evaluación de riesgos, preoperatorio e instrucciones se mantienen en el sistema y también ofrece facilidades para almacenar detalles sobre la conducta de la anestesia. Destreza para almacenar los detalles de lo ocurrido durante la cirugía, como las constantes vitales, pulso, fluidos, la disponibilidad de drogas en cualquier momento y después de que la cirugía ha terminado. Las complicaciones se añaden en la nota de cirugía. Se registran los cuidados de enfermería para los cuidados posoperatorios y detalles tales como investigaciones, recuperación de las condiciones, problemas o dolor. (10)

1.2.6 Optim OTM: VANGUARD ofrece a los hospitales soluciones de software modulares e integradas para una amplia planificación y documentación de todos los procesos y flujos de producción relevantes para el Bloque Quirúrgico. Presenta una arquitectura cliente servidor. VANGUARD Optim OTM ofrece a los hospitales un sistema de planificación y documentación en tiempo real para el Bloque Quirúrgico que es fácil de integrar en su actual rutina de trabajo. Mediante un cómodo registro por código de barras se documentan todos los procesos en el Bloque Quirúrgico, pudiendo disponerse inmediatamente de ellos para los más diversos requerimientos de trazabilidad y gestión.(11)

1.2.7 Soarian™ MedSuite: Organiza el trabajo en 4 direcciones, ellas son la gestión administrativa del paciente, la gestión clínica del paciente, la gestión de medicamentos y materiales, y el laboratorio. Presenta una arquitectura cliente servidor. El módulo Quirófano es usado para procesar y administrar la información y servicios de los quirófanos de un hospital. A través de este módulo, el sistema permite consultar y procesar todos los servicios de cirugía requeridos. Los servicios de cirugía pueden estar vinculados a agendas que se configuran independientemente y se asocian a uno o varios quirófanos. Este

módulo permite solicitar los instrumentos al sector de esterilización así como llevar un control de stock de instrumentos del quirófano. Permite también capturar detalles pre y posoperatorios, se podrán definir las plantillas con las cuales el personal del Quirófano registrará esta información. (12)

1.2.8 Hosix-V: Es un sistema de gestión e información hospitalaria flexible, integrado y modular, que abarca todas las áreas de actividad de un hospital y pretende, a través de una utilización fácil, rentabilizar los recursos existentes para organizar el trabajo desarrollado diariamente. Presenta una arquitectura cliente servidor. El módulo de "quirófanos" se encarga de mantener un control informatizado de los tiempos de ocupación de los quirófanos, de los servicios que los utilizan, de las técnicas a utilizar, de las personas que participan, para conseguir la optimización de la programación de intervenciones tanto de pacientes internos como externos. Entre sus principales características se encuentran:

- Registro de las actividades realizadas en las operaciones quirúrgicas desde su programación, transcurso y posoperatorio, permitiendo el registro y control de actividades quirúrgicas tanto programadas como por la vía de urgencia.
- Seguimiento y registro de las actividades del paciente realizadas durante su estancia en el área quirúrgica, así como la inclusión en su historial clínico.
- Seguimiento y registro de todo aquel personal sanitario que va a participar en la intervención, por ejemplo, cirujanos, anestesistas, etc.

1.2.9 Opera: Es un sistema de CHCA Computer Systems Inc. que brinda a los profesionales de la salud y administradores todas las informaciones clínicas y financieras de cada intervención quirúrgica, a programar, real o ya realizada. Opera le permite gestionar su planificación quirúrgica al facilitarle amplios datos clínicos y operativos, además accede a documentar información sobre el estado quirúrgico en la zona de cuidados y compartir dicha información con todos los miembros que intervienen en el proceso de atención al paciente.

Opera soporta la toma de decisiones y la administración general del bloque operatorio, gracias a la producción de una información real y precisa. La solución informática Opera incluye los módulos para la planificación de las intervenciones, los exámenes de pre admisión, la administración de los recursos, la documentación del tiempo de los procedimientos por cada intervención (Pre operatorio, Operatorio, sala de recuperación, Posoperatorio), las notas clínicas de la intervención, el resumen de todo el procedimiento, la gestión del material y la administración de la esterilización. (13)

Nombre del Sistema	País	Empresa	Multipla-taforma	Bloque Quirúrgico		Tipo de Licencia		Tipo de Aplicación	
				Módulo	Sistema	Libre	Privativo	Web	Desktop
Sushrut	India	C-CAD	X	X			X		X
Care2X	Alemania	CARE	X		X	X		X	
BQO	Cuba	UCI	X		X		X	X	
GoWin	España	Valen Computer		X			X		X
LCS	India	Futurism		X			X	X	
Optim OTM	Francia	Vanguard			X		X	X	
Soarian™ MedSuite	Argentina	Siemens		X			X	X	
Hosix-V	España	Sivsa		X			X	X	
Opera	Canadá	CHCA			X		X		X

Tabla 1.2 Análisis de HIS en el mundo

Observaciones generales de los sistemas estudiados.

Los HIS mencionados anteriormente, en su mayoría no son multiplataforma y operan en sistemas operativos privados lo cual resulta un inconveniente para insertarse en el mercado del software. Se puede constatar además que en su mayoría no es software libre, condición que limita al país que adquiera el sistema para conseguir futuras mejoras a las funcionalidades que brinda, debido a las restricciones y alto coste de sus licencias.

La gran mayoría de estas soluciones representan al área quirúrgica como un módulo que se integra con las demás áreas del hospital lo que permite una mejor atención al paciente, aunque existen casos de sistemas especializados que solo se encargan de la gestión de información en esta área. Se ve un predominio de las aplicaciones web, las cuales se basan en una arquitectura cliente-servidor, ratificando que esta es una tendencia en el desarrollo de los sistemas actuales, aunque hay algunas que son aplicaciones de escritorio las que se verían dificultadas con su despliegue y soporte.

Care2x por su parte ofrece la ventaja de ser software libre y una aplicación web, pero el sistema gestor de base de datos MySQL que soporta la información gestionada por el sistema, no cuenta con la escalabilidad necesaria para el incremento de los datos inherentes en una aplicación de este nivel. Además, no abarca todas las funcionalidades que se pretenden desarrollar para que el sistema sea configurable a los procesos quirúrgicos de cualquier institución hospitalaria.

En el caso de BQO es una aplicación a la cual se puede tener acceso al código fuente y documentación por lo que se podría integrar como un servicio quirúrgico del módulo de Bloque Quirúrgico para desarrollarlo sobre software libre y en función de la integración con otros módulos del alas HIS, obtener las funcionalidades del BQO que son comunes al entorno de trabajo de los restantes servicios quirúrgicos y así poder adaptar el futuro sistema, a cualquier institución hospitalaria independientemente de los servicios quirúrgicos que disponga.

1.3 Tendencias y tecnologías actuales a considerar

Para desarrollar un HIS es imprescindible la utilización de una metodología de software para la documentación del gran volumen de información generada, así como el empleo de tecnologías y herramientas de desarrollo avanzadas y libres de restricciones. A continuación se definen las características de las tecnologías a considerar para el desarrollo del sistema.

Una aplicación web se puede definir como una aplicación en la cual un usuario por medio de un navegador realiza peticiones a una aplicación remota accesible a través de Internet (o a través de una intranet) y que recibe una respuesta que se muestra en el propio navegador. (14)

Estos sistemas informáticos se basan en la arquitectura cliente-servidor donde el navegador representa al cliente el cual interpreta el código del lenguaje etiquetado de hipertexto (HTML, por sus siglas en inglés) para formar la página web que se mostrará ante una petición que realice el usuario al servidor web mediante el protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés). Su principal ventaja es el acceso a la información desde cualquier máquina que se conecte con el servidor y así de esta forma es innecesario incurrir en gastos de alta tecnología para la instalación, actualización o mantenimiento del programa en los clientes.

Un web browser o navegador es una aplicación que opera a través de Internet, interpretando archivos y sitios web desarrollados a menudo en código HTML que contienen información y contenido en hipertexto de todas partes del mundo. El navegador tiene el expreso propósito de mejorar esta experiencia, a través de la incorporación de funcionalidades que agilicen la navegación, o bien, que ofrezcan la información en la mejor calidad disponible. Otra esencial función e interés de los navegadores es asegurar una experiencia segura al usuario, protegiéndolo de errores, virus y otros elementos nocivos que pueden hallarse en la web y afectar la computadora que realiza la navegación. (15)

Un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos. Los archivos para cada sitio de Internet se almacenan y se ejecutan en el servidor. Hay muchos servidores en Internet y muchos tipos de servidores, pero comparten la función común de proporcionar el acceso a los archivos y servicios. En la web, un servidor web es un ordenador que usa el protocolo HTTP para enviar páginas web o páginas HTML al ordenador de un usuario cuando él las solicita. Los servidores se conectan a la red mediante una interfaz que puede ser una red verdadera o mediante conexión vía línea telefónica o digital. (16)

1.3.1 Sistemas distribuidos. Modelo Cliente Servidor

Los sistemas distribuidos han surgido a partir del desarrollo de las redes para brindar servicio a numerosas prestaciones. Esta tendencia se ha acelerado por el desarrollo de software para sistemas

distribuidos, diseñado para soportar el desarrollo de aplicaciones distribuidas. Este software permite a los ordenadores coordinar sus actividades y compartir los recursos del sistema: hardware, software y datos. Esta perspectiva lleva a dos modelos de sistemas distribuidos: el modelo cliente-servidor y el modelo basado en objetos. (17)

El modelo cliente-servidor es una arquitectura donde se encuentran distribuidos los usuarios del sistema, donde el cliente que es la vista del programa que maneja el usuario es el que realiza peticiones al servidor que es el encargado de darle respuesta a las acciones que realiza el cliente. Este modelo se recomienda, en particular, para redes que requieran un alto grado de fiabilidad. Las principales ventajas son:

- Recursos centralizados: debido a que el servidor es el centro de la red, puede administrar los recursos que son comunes a todos los usuarios, por ejemplo: una base de datos centralizada se utilizaría para evitar problemas provocados por datos contradictorios y redundantes.
- Seguridad mejorada: ya que la cantidad de puntos de entrada que permite el acceso a los datos no es importante.
- Administración al nivel del servidor: ya que los clientes no juegan un papel importante en este modelo, requieren menos administración.
- Red escalable: gracias a esta arquitectura, es posible quitar o agregar clientes sin afectar el funcionamiento de la red y sin la necesidad de realizar mayores modificaciones. (18)

La arquitectura de software define, de manera abstracta, los componentes, sus interfaces y la comunicación ente ellos. La arquitectura por capas separa en niveles lógicos la organización del sistema basado en un modelo cliente-servidor para compartir la información existente en varios nodos físicos. Descompone la aplicación en capas independientes y ordenadas jerárquicamente, cada nivel o capa usa los servicios de la capa inmediatamente inferior y ofrece servicios a la capa inmediatamente superior. Las capas se utilizan en las aplicaciones web para optimizar su desarrollo y de esta forma separar las funcionalidades en distintos niveles para reducir las dependencias que dificultan la construcción y mantenimiento del sistema. (19)

Capa de Presentación: esta capa se encarga de validar la accesibilidad e interacción del usuario con el sistema para el intercambio de información.

Capa de Negocio o Intermedia: esta capa se encarga de la automatización los procesos y reglas del negocio para responder las peticiones de los usuarios que interactúan con la capa de presentación.

Capa de acceso a datos: esta capa se encarga de la persistencia de los datos que la capa intermedia procesa de la información gestionada por el usuario. (20)

La arquitectura se basa en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software. Dicha arquitectura establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita alcanzar los objetivos del sistema.

1.3.2 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva porque ya ha resuelto un problema similar anteriormente de manera satisfactoria y que es reusable porque se puede aplicar a muchos problemas de diseño en diferentes circunstancias. (21) La reutilización del diseño reduce los esfuerzos de desarrollo y mantenimiento. Además, el uso de patrones permite que entre el equipo de desarrolladores exista una comunicación fluida y precisa de las ideas esenciales sobre el diseño de la aplicación.

1.3.3 Patrones de arquitectura

En el desarrollo de aplicaciones informáticas se utilizan los llamados patrones de arquitectura o estilos arquitectónicos que definen las reglas generales de organización en partiendo de un patrón y las restricciones en la forma y la estructura fundamental de los sistemas. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas. (22)

Modelo Vista Controlador: (MVC, por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Son muchas las empresas que deciden pasar sus aplicaciones a la arquitectura modelo vista controlador para documentar más fácilmente el código, ahorrar espacio, y en caso de no disponer de diseñadores web, poder contratar los servicios de un diseñador que no sepa mucho de programación que les haga las vistas.

- El Modelo es todo acceso a datos, y las funciones que llevan lo que llaman “lógica de negocio”, o sea, datos y reglas de negocio. Lleva un registro de las vistas y controladores del sistema.
- La Vista, en una aplicación web, es el HTML y lo necesario para convertir datos en HTML. O sea, muestra la información del modelo al usuario. Tiene un registro de su controlador asociado.

- El Controlador une la vista y el modelo. Por ejemplo, son las funciones que toman los valores de un formulario, consultan la base de datos (a través del modelo) y producen valores, que la vista tomará y convertirá en HTML. En resumen, gestiona las entradas del usuario. Estas acciones pueden suponer peticiones al modelo o a las vistas. (23)

1.3.4 Metodologías de desarrollo de software

Las metodologías surgen por la necesidad de documentar proyectos de alta complejidad. Como el desarrollo del software es riesgoso y difícil de controlar, para evitar la insatisfacción de los clientes es preciso fundamentar el proceso de desarrollo en una metodología los más flexible posible a los requerimientos del sistema y que cumpla con los objetivos del proyecto. A la vez se debe configurar de acuerdo con los recursos disponibles, el proceso a seguir para poner en práctica la metodología.

1.3.4.1 Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés)

Metodología que utiliza el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) para generar todos los esquemas de un software, es un proceso dirigido por los casos de usos o funcionalidades que el sistema debe cumplir para satisfacer las necesidades del cliente, centrado en la arquitectura que se propone para el diseño de la aplicación e iterativo e incremental a lo largo del ciclo de vida del software.

RUP describe cómo utilizar arquitecturas basadas en componentes. El proceso de software debe focalizarse en el desarrollo temprano de una arquitectura robusta antes de comprometer recursos para el desarrollo en gran escala. RUP describe cómo diseñar una arquitectura flexible, que se acomode a los cambios, comprensible intuitivamente y promueve una efectiva reutilización de software. Soporta el desarrollo de software basado en componentes: módulos no triviales que completan una función clara.

1.3.4.2 Lenguaje Unificado de Modelado

UML es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. (24) El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientado a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios. (25)

Tecnologías y herramientas utilizadas en el proceso de desarrollo

1.3.4.3 Java

Java es un lenguaje de programación simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. (26) Una característica distintiva de Java, es su capacidad multiplataforma. Esto lo consigue, no solo a nivel de código fuente, sino también a nivel de código compilado. Java podrá ejecutarse en cualquier sistema operativo que tenga una máquina virtual Java compatible. (27)

1.3.4.4 Java EE

Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de N niveles, distribuida, ejecutándose sobre un servidor de aplicaciones. Ofrece nuevas y actualizadas funciones como Enterprise JavaBeans (EJB) Tecnología 3.0, Tecnología Java Server Faces (JSF). (28)

1.3.4.5 XHTML

Familia de módulos y tipos de documentos que reproduce, engloba y extiende HTML 4.0. Los tipos de documentos de la familia XHTML están basados en XML, y diseñados fundamentalmente para trabajar en conjunto con aplicaciones de usuario basadas en XML. Como tales, son fácilmente visualizados, editados y validados con herramientas XML estándar. Estos documentos XHTML pueden escribirse para que funcionen igual o mejor que lo hacían antes, tanto en las aplicaciones de usuarios conformes a HTML 4.0 como en las nuevas aplicaciones conformes a XHTML 1.0. (29)

1.3.4.6 JSF

Es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. Incluye un conjunto de interfaces de programación de aplicaciones (API, por sus siglas en inglés) para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para la internacionalización y accesibilidad. Existen dos librerías de etiquetas personalizadas para Java Server Pages que permiten expresar una interfaz Java Server Faces dentro de una página JSP. Un

modelo de eventos en el lado del servidor y administración de estados. La tecnología Java Server Faces simplifica la construcción de interfaces de usuario. (30)

1.3.4.7 Java Server Faces Expression Language

Lenguaje de expresión que provee JSF, utilizado en las páginas de las aplicaciones web con el fin de acceder a las entidades java.

1.3.4.8 Facelets

Facelets es un framework ligero que permite el uso de plantillas en aplicaciones JSF. Sus principales ventajas son:

- Construcción de interfaces basadas en plantillas.
- Rápida creación de componentes por composición.
- Fácil creación de funciones y librerías de componentes. (31)

1.3.4.9 Richfaces

Richfaces es un framework de código abierto que añade capacidad Ajax dentro de aplicaciones JSF existentes sin recurrir a JavaScript. Richfaces incluye ciclo de vida, validaciones, conversores y la gestión de recursos estáticos y dinámicos. Los componentes de Richfaces están construidos con soporte Ajax y un alto grado de personalización del “look-and-feel” que puede ser fácilmente incorporado dentro de las aplicaciones JSF. (32)

1.3.4.10 Ajax4jsf

Es una librería open source que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript. Mediante este framework se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automático y controlar los eventos de usuario. (33)

1.3.4.11 Jboss Seam

Seam es un framework open source desarrollado por la empresa Jboss, con el fin de unir diferentes tecnologías y estándares de Java en un solo framework, a la vez que añade algunas funcionalidades no contempladas por ellos. Está basado en la arquitectura Modelo Vista Controlador (MVC). Integra la capa de presentación (JSF) con la capa de negocios y persistencia (Enterprise JavaBeans EJB3). Con Seam basta agregar anotaciones propias de éste a los objetos entidad y session de EJB, logrando con esto escribir menos código Java y XML.

Otra característica importante es que se pueden hacer validaciones en los POJOs (Plain Object Java) como además manejar directamente la lógica de la aplicación y de negocios desde las sessions beans. Seam también se integra perfectamente con otros frameworks como: RichFaces, ICEFaces (soportan AjaX) MyFaces, Hibernate y Spring (34). Además integra las tecnologías como JavaScript Asíncrono y XML (ÁJAX), Java Server Faces (JSF), Enterprise Java Beans (EJB3), Java Persistence (JPA) y Business Process Management (BPM). (35)

1.3.4.12 Seam User Interface (UI)

Conjunto de controles JSF modificados que se integran con Seam. (36)

1.3.4.13 EJB3

Enterprise JavaBeans (EJB) es la tecnología del lado del servidor para la arquitectura de componentes de la Plataforma Java, (Java EE). La tecnología EJB permite el desarrollo rápido y simplificado de aplicaciones distribuidas, transaccionales, seguras y portátiles basadas en tecnología Java. EJB 3.0 define la nueva API EJB simplificado dirigido a la facilidad de desarrollo e incluye el nuevo Java Persistence API para la gestión de la persistencia. (37)

1.3.4.14 JPA

Java Persistence API (JPA) es el estándar para la gestión de la persistencia y provee facilidades para el mapeo objeto / relacional a desarrolladores de aplicaciones, haciendo uso del modelo de dominio de Java para administrar bases de datos relacionales. JPA es parte de la plataforma Java EE. (38)

1.3.4.15 JBoss Server

JBoss Application Server es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada Java EE, soporta todas las funcionalidades de Java EE 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 de los empresarios sean mucho más simples. (39)

1.3.4.16 Hibernate

Hibernate es una herramienta de mapeo objeto/relacional y un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se pueden generar bases de datos en cualquiera de los entornos soportados. Es de código abierto, lo que supone, que ha sido lanzado bajo una licencia LGPL. (40)

1.3.4.17 Sistemas Gestores de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es una herramienta que permite a los usuarios crear y mantener una base de datos, por lo que es un software de propósito general que facilita el proceso de definir, construir y manipular los datos almacenados. Estos sistemas se especializan en la validación de datos para evitar redundancias y errores que afecten la integridad y seguridad de la información registrada. Además, garantiza el control centralizado de la información de manera sistemática y única y el acceso a la base de datos, así como la integración y sincronización de ella para el uso de múltiples usuarios. (41)

1.3.4.18 PostgreSQL

Es el sistema de gestión de base de datos relacional orientada a objetos de código abierto más avanzado del mundo. PostgreSQL es un potente sistema de base de datos objeto/relacional. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de los datos, y corrección. Funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX y Windows. Es altamente escalable, tanto en la enorme cantidad de datos que puede administrar como en el número de usuarios concurrentes que puede soportar. (42)

1.3.4.19 pgAdmin

Es la plataforma de desarrollo de PostgreSQL y la más avanzada base de datos de código abierto en el mundo. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir simples consultas SQL hasta la elaboración de complejas bases de datos. La interfaz gráfica soporta todas las características de PostgreSQL y hace fácil la administración. La conexión con el servidor puede hacerse utilizando el protocolo TCP / IP y puede ser encriptado SSL para la seguridad. No se requieren drivers adicionales para comunicarse con el servidor de base de datos. Es desarrollado por una comunidad de expertos de PostgreSQL de todo el mundo y está disponible en más de una docena de idiomas. (43)

1.3.4.20 Visual Paradigm

Es una herramienta multiplataforma de modelado visual UML y una herramienta CASE muy potente y fácil de utilizar. VP-UML aporta a los desarrolladores de software una plataforma de desarrollo puntera para construir aplicaciones de calidad, mejores y menos costosas. Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE líderes del mercado. (44) Se integra con las herramientas Java y está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. (45)

1.3.4.21 Eclipse

La plataforma Eclipse consiste en un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) abierto y extensible. Un IDE es un programa compuesto por un conjunto de herramientas útiles para un desarrollador de software. Como elementos básicos, un IDE cuenta con en un editor de código, un compilador/intérprete y un depurador. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plugins) para extender la funcionalidad.

Tiene características potentes como el completamiento de código, permite mostrar desde varias perspectivas el entorno de cada framework permitiendo la integración de sus funcionalidades y el trabajo en equipo mediante el Subeclipse para el manejo de versiones. Eclipse necesita tener instalado en el sistema una máquina virtual Java, preferiblemente JRE (Java Runtime Environment).

1.3.4.22 JBoss Tools

Es de manera general un paquete de programas que añade funcionalidades a Eclipse, fue diseñado con el objetivo de ayudar a los desarrolladores que utilizan JBoss y Java EE a implementar sus aplicaciones rápidamente. Ofrece varios módulos como RichFacesVE, Hibernate Tools, Seam Tools y jBPM Tools.

En este capítulo, han sido investigados los HIS para la gestión de información en el área de Bloque Quirúrgico, estos no satisfacen la totalidad de las necesidades. Es por eso que después del estudio realizado de las tendencias y tecnologías actuales que se podrían emplear para dar una solución informática, se propone el desarrollo de una aplicación web basada en el modelo cliente servidor, donde se tendrá: se utilizará la metodología de desarrollo de software el Proceso Unificado de Desarrollo, el Lenguaje Unificado de Modelado y la Notación para el Modelado de Procesos de Negocio con Visual Paradigm como herramienta CASE.

Además, para crear la aplicación se utilizará la herramienta de desarrollo Eclipse para la tecnología Java, el lenguaje Java, JSF y Richfaces 3.2 como librerías de componentes web y para su funcionamiento se hará uso del servidor web Jboss Server 4.2. Para integrar todas estas tecnologías se utilizará el framework Jboss Seam 2.1.1 y Jboss Tools para el desarrollo de aplicaciones web. Se empleará el Sistema gestor de base de datos PostgreSQL 8.3, pgAdmin III como cliente para gestionar la BD y el framework Hibernate para la gestión de los datos.

Capítulo 2: Descripción de la arquitectura

El software tradicional, como único proceso ejecutándose en un único ordenador, no supone arquitecturas complejas ni grandes riesgos. Sin embargo, los sistemas actuales aprovechan componentes comerciales, código abierto, sistemas distribuidos, nuevos y diversos lenguajes de programación, entornos de alojamiento (hosting) y de ejecución remota, y otras dependencias externas, que convierten a la arquitectura de un sistema en su producto técnico más crítico. Alcanzar una arquitectura estable que de garantías sobre la viabilidad del proyecto se considera el punto de transición entre lo que se suele denominar la fase de ingeniería para la arquitectura desde la perspectiva de los programadores.

De acuerdo con la utilización del patrón de arquitectura MVC, el modelo de diseño tiene una estructuración lógica donde las páginas clientes y formularios se crean a partir del código XHTML que construyen las páginas servidoras cuyas clases conforman las vistas. Estas invocan a las clases controladoras las cuales se encargan de ejecutar la lógica del negocio asociada a una petición del usuario. Estas acciones que ejecutan las clases controladoras modifican las clases entidades que indirectamente son actualizadas y consultadas por las vistas del sistema.

2.1 Requisitos no funcionales

Los requisitos no funcionales son cualidades o propiedades que el sistema debe cumplir para facilitar su uso y darle valor agregado a las funcionalidades que le brinda al usuario. Estos requisitos son de gran significación en la aceptación del software, debido a que representan las ventajas más visibles al usuario y repercuten en el óptimo funcionamiento y mantenimiento del sistema.

RNF Usabilidad

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

- Para alcanzar un nivel Elemental asociado al dominio del sistema y el uso eficiente del mismo, serán necesarios 20 días de preparación, obteniendo la categoría de Usuarios normales.
- Para alcanzar un nivel Avanzado asociado al dominio del sistema y el uso eficiente del mismo, serán necesarios 30 días de preparación, obteniendo la categoría de Usuarios avanzados.
- Brindará comodidad a la hora de acceder a las diferentes funcionalidades que proporciona la aplicación mediante teclas de acceso rápido.

RNF Soporte

Seguridad de acceso y administración de usuarios:

- Se mantendrá seguridad y control entre usuario, garantizando su acceso solo a los niveles establecidos de acuerdo con la función que realizan. Las contraseñas podrán cambiarse por el propio usuario o por el administrador del sistema.
- Se mantendrá un segundo nivel de seguridad entre estaciones de trabajo, garantizando únicamente la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión. Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.
- Se establecerán mecanismos de control y verificación para los procesos susceptibles de fraude.
- El sistema proporcionará un registro de actividades (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 que se realizó, 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.

RNF 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.

RNF 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.

RNF 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 menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario.
- Todos los textos y mensajes en pantalla aparecerán en idioma español.

Interfaces software:

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

Interfaces de comunicación:

Para el intercambio electrónico de datos entre aplicaciones se usará el protocolo HL7 (Health Level Seven). El sistema usará el formato estándar WSDL (Web Services Desciption Language) para la descripción de los servicios web. El sistema implementará mecanismos de encriptación de datos para el intercambio de información con sistemas externos. El sistema utilizará mecanismos de compactación de los datos que se intercambiarán con sistemas externos con el objetivo de minimizar el tráfico en la red y economizar el ancho de banda.

Interfaces hardware:

Los equipos autoanalizadores se podrán programar desde el sistema para realizar análisis y obtener resultados automáticamente.

RNF Requerimientos de rendimiento

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

RNF 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 que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, se recomienda IE 7, Firefox 2 o versiones superiores. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz con sistema operativo Linux.

Servidores:

La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables. Servidores de Base de datos: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux. Servidores de Aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux. Servidores de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 2 GB de memoria y 2x72GB de disco y sistema operativo Linux.

RNF Requerimientos de software

El sistema debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss AS y PostgreSQL). El sistema deberá disponer de un navegador web, estos pueden ser IE 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

2.2 Descripción de la arquitectura. Fundamentación.

Un sistema software es una única entidad, pero al arquitecto y a los desarrolladores les resulta útil presentar el sistema desde diferentes perspectivas para comprender mejor el diseño. Estas perspectivas son vistas del modelo del sistema. Todas las vistas juntas representan la arquitectura. La arquitectura abarca decisiones importantes sobre: la organización del sistema software; los elementos que compondrán el sistema y sus interfaces, junto con sus comportamientos, tal y como se especifican en las

colaboraciones entre estos elementos; la composición de elementos estructurales y del comportamiento en subsistemas progresivamente más grandes; el estilo de la arquitectura que guía esta organización, los elementos y sus interfaces, sus colaboraciones y su composición.

Sin embargo, la arquitectura de software está afectada no solo por la estructura y el comportamiento, sino también por el uso, la funcionalidad, el rendimiento, la flexibilidad, la reutilización, la facilidad de comprensión, las restricciones y compromisos económicos, tecnológicos y la estética.

Patrones de arquitectura.

Un patrón es una solución a un problema que aparece con frecuencia. Se presentan como plantillas donde se les asigna un nombre y un resumen de los problemas, las fuerzas que lo hacen surgir, y las ventajas y desventajas que provee su utilización. El patrón de arquitectura que más se pone de manifiesto en el diseño de la aplicación es el Modelo-Vista-Controlador (MVC). Este patrón es útil principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de mejor manera, facilitando la programación en diferentes capas de forma paralela e independiente.

La vista(o presentación) es la representación de los datos en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación web la vista es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones. La capa de presentación está desarrollada básicamente con JSF, usando la librería de componentes RichFaces 3.2.0 G.A., esta integra fácilmente con el framework de integración escogido (Seam) y permite generar vistas no necesariamente basadas en HTML (PDF, etc.), adiciona además controles listos para usar y el framework de extensión AJAX para los controles JSF básicos Ajax4Jsf. Por su parte los controles para interfaz de usuario de Seam adicionan varias mejoras a JSF, desde validación, integración de la navegación en la interfaz de usuario basada en flujos de navegación o procesos del negocio, etc.

El controlador es el encargado de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando los datos en caso de que sea requerido. En esta capa y como framework de integración se usa Seam, un poderoso y moderno framework creado para unificar todas las tecnologías estándares JSF, EJB3, JPA, además de BPM (Business Process Management). Fue creado desde el inicio para eliminar la complejidad a nivel desde arquitectura hasta API, permitiendo la creación de complejas aplicaciones web basadas en POJOs, componentes de interfaz de usuario y el mínimo y

solamente necesario XML. Se integra con librerías de controles de código abierto basadas en JSF como RichFaces, ICEFaces, etc.

Una de sus principales innovaciones es en el campo de la administración de estado, mientras que en los frameworks tradicionales todo el estado es administrado básicamente en la sesión HTTP, Seam provee una mayor granularidad de contextos de estado. La principal, quizás es el contexto conversacional, así como el asociado a procesos del negocio, con estos se logra un uso más eficiente de la memoria. Integra además el concepto de espacios de trabajo, permitiendo que el usuario tenga en varias ventanas del navegador actividades del negocio con contextos completamente aislados. Seam integra de forma transparente la administración de procesos del negocio vía JBoss JBPM, haciendo muy fácil implementar y optimizar complejas colaboraciones y complejas interacciones con el usuario.

El modelo es la representación de la información que maneja la aplicación. El modelo son los datos puros que puestos en el contexto del sistema provee de información al usuario o a la aplicación misma. 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. Además se pueden establecer validaciones gracias a los Hibernate Validators.

Algunas incumbencias son horizontales, se encuentran en todas las capas de la aplicación, la seguridad es una de ellas. En este caso, toda la autorización, desde el acceso a directorios, páginas, controles, opciones del menú, servicios del negocio, está basado en reglas. Lo que permite que ninguna de las “reglas del negocio” esté escrita como código en la aplicación y que el cambio de alguna de ellas, no requiera cambio alguno en el código, solo en la definición de alguna regla en un fichero de configuración. El Seam Security Framework permite todo esto gracias a su integración con el potente motor de reglas JBoss Rules.

2.3 Análisis de componentes rehusados

El módulo de Bloque Quirúrgico se encarga de la disponibilidad de quirófanos, equipos, y materiales para organizar el trabajo de los servicios quirúrgicos. Entre sus principales funciones se encuentra la programación quirúrgica y la documentación de los procedimientos quirúrgicos realizados para la atención clínica del paciente en este servicio. Uno de los principales procesos que se llevan a cabo en esta área es la elaboración del plan quirúrgico, donde se realiza una planificación de las intervenciones por quirófano

teniendo en cuenta el equipo que participará en la cirugía, en dicho proceso la secretaria de quirófanos es la encargada de recibir las solicitudes de intervención procedentes de otras áreas y aprobarlas o cancelarlas, de acuerdo a la distribución de quirófanos por servicio.

En el proceso de intervención quirúrgica el anestesiólogo es el encargado de suministrarle al paciente las drogas necesarias para la cirugía, monitorear constantemente el comportamiento del paciente y registrar cada cierto tiempo los signos vitales. Debe realizar un seguimiento riguroso al paciente tanto en el preoperatorio, transoperatorio como en el posoperatorio.

Estos datos son registrados en la hoja de anestesia donde se le asocia la información de la Historia Clínica de la solicitud de intervención quirúrgica seleccionada de acuerdo con el plan quirúrgico. Dada la importancia que tiene el registro de esta información en el acto quirúrgico es necesario contar con la funcionalidad crear hoja de anestesia estándar y crear hoja de anestesia cardiovascular correspondiente a una intervención quirúrgica determinada y poder registrar dentro de las mismas los datos obtenidos en las diferentes etapas por las cuales transita el paciente durante todo el proceso quirúrgico.

Para el desarrollo de dicha tarea fue necesario utilizar cuatro funcionalidades comunes que ya se encuentran implementadas pues las mismas son necesarias para la realización de otros procesos que se llevan a cabo dentro del módulo, las cuales se mencionan a continuación: Seleccionar Médico, Seleccionar Procedimiento Quirúrgico, Seleccionar Medicamento y Seleccionar Enfermera.

Seleccionar Médico brinda la posibilidad de seleccionar los criterios de búsqueda para localizar al médico, el sistema busca y muestra los médicos que cumplen con los criterios de búsqueda, se selecciona el médico deseado, el sistema devuelve el médico seleccionado para que sus datos sean manejados por el proceso que los solicite.

Seleccionar Procedimiento Quirúrgico brinda la posibilidad de seleccionar el procedimiento quirúrgico deseado, el sistema devuelve los procedimientos quirúrgicos seleccionados para que sus datos sean manejados por el proceso que los solicite.

Seleccionar Medicamento brinda la posibilidad de seleccionar los criterios de búsqueda para localizar el medicamento deseado, el sistema busca y muestra los medicamentos que cumplen con los criterios de búsqueda, se selecciona el medicamento deseado, el sistema devuelve los medicamentos seleccionados para que sus datos sean manejados por el proceso que los solicite.

Seleccionar Enfermera brinda la posibilidad de seleccionar los criterios de búsqueda para localizar la enfermera deseada, el sistema busca y muestra las enfermeras que cumplen con los criterios de

búsqueda, se selecciona la enfermera deseada, el sistema devuelve las enfermeras seleccionadas para que sus datos sean manejados por el proceso que los solicite.

Estas funcionalidades fueron reutilizadas en la implementación de la Hoja de anestesia estándar y la Hoja de anestesia cardiovascular como se describe a continuación:

- Hoja de anestesia estándar.

1. Consulta preanestésica.
    - 1.1. Seleccionar Procedimiento Quirúrgico.
      Se selecciona la intervención propuesta.
    - 1.2. Seleccionar Médico.
      Se selecciona el anestesiólogo y su auxiliar.
2. Preoperatorio.
    - 2.1. Seleccionar Medicamento.
      Se seleccionan los medicamentos (drogas, dosis, vías, horas) que se deben suministrar al paciente antes de la anestesia.
    - 2.2. Seleccionar Médico.
      Se seleccionan los anestesiólogos.
3. Transoperatorio.
    - 3.1. Seleccionar Medicamento.
      Se seleccionan los medicamentos de anestesia, drogas y líquidos que se le suministran al paciente, se especifica la cantidad de la dosis.
    - 3.2. Seleccionar Médico.
      Se selecciona el equipo de anestesiólogos que participan en la intervención (Anestesiólogo principal al frente de la intervención y sus asistentes).
      Se selecciona el cirujano de la intervención.
      Se seleccionan los asistentes del cirujano.
    - 3.3. Seleccionar Procedimiento Quirúrgico.
      Seleccionar la intervención propuesta.
    - 3.4. Seleccionar Enfermera.
      Se seleccionan las enfermeras que participan en la intervención (circulante de anestesia, circulante de cirugía e instrumentista).

4. Posoperatorio.
    4.1. Seleccionar Médico.
    Se selecciona el anestesiólogo que determina aprobar el alta del paciente.
- Hoja de anestesia cardiovascular.

1. Transoperatorio.
    1.1. Seleccionar Médico.
    Se selecciona el equipo de anestesiólogos que participan en la intervención.
    Se selecciona el cirujano de la intervención.
    1.2. Seleccionar Procedimiento Quirúrgico.
    Seleccionar el procedimiento quirúrgico.

Se demuestra así la importancia de la reutilización para dar solución a problemas existentes que sean comunes en la implementación de los procesos dentro del módulo Bloque Quirúrgico.

2.4 Especificación de los términos de seguridad

Los problemas de seguridad informática no pueden ser tratados aisladamente ya que la seguridad de todo el sistema es igual a la de su punto más débil. Por lo tanto, se apoyarán en técnicas desarrolladas para proteger los equipos informáticos individuales y conectados en una red frente a daños accidentales o intencionados. Estos daños incluyen el mal funcionamiento del hardware, la pérdida física de datos y el acceso a bases de datos por personas no autorizadas. La seguridad es un tema de gran impacto en el sistema, pues es de vital importancia el control de la información que se almacena y se visualiza para garantizar la calidad de los servicios quirúrgicos. De ahí que todo usuario que necesite utilizar alguna funcionalidad del módulo deberá autenticarse para realizar alguna acción sobre el mismo.

Un usuario podrá tener más de un rol en el sistema de acuerdo con las acciones que realiza, mediante el cual se le otorgan determinados permisos para el acceso a la información y cada vez que el usuario realice una acción sobre el sistema, se registrará una traza que contiene la información gestionada mediante su estancia en el módulo. A continuación se describen detalladamente las funcionalidades que ofrece el módulo de Configuración del Sistema de Información Hospitalaria alas HIS, el cual se encargará de garantizar la seguridad en el sistema a desarrollar.

Iniciar/Cerrar sesión de trabajo

Cuando el usuario necesita acceder al sistema, este solicita: nombre de usuario y contraseña. El usuario introduce los datos solicitados, el sistema verifica que los datos introducidos sean válidos, si es así, el

usuario accede al módulo Bloque Quirúrgico. El sistema muestra como opciones del menú las funcionalidades a las que tiene permiso de acceder el usuario en el módulo, lo que garantiza el acceso de los mismos solo a los niveles establecidos de acuerdo con la función que realizan. El sistema permite: cerrar sesión y salir del módulo.

Registrar trazas

Cuando el usuario realiza una acción sobre el sistema, que puede ser: inicio o cierre de sesión, acceso al módulo, modificación a un atributo de una entidad o cualquier otra operación sobre el sistema, este registra una traza en la base de datos.

Administrar seguridad

El sistema brinda la posibilidad de asignar o denegar permiso a roles y usuarios en las funcionalidades del módulo.

Configurar funcionalidades

El sistema brinda la posibilidad de configurar las funcionalidades del módulo.

2.5 Diagrama de despliegue

El modelo de Despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Para la implantación y la utilización de la aplicación en una institución hospitalaria el usuario debe conectarse a esta mediante una PC cliente utilizando un navegador web. Las peticiones por el protocolo HTTP serán procesadas por el servidor de aplicaciones que enviará la respuesta al cliente y en caso que sea necesario también hará peticiones mediante el protocolo TCP-IP al servidor de base de datos. El usuario puede imprimir reportes o gráficos desde la PC cliente utilizando una impresora.

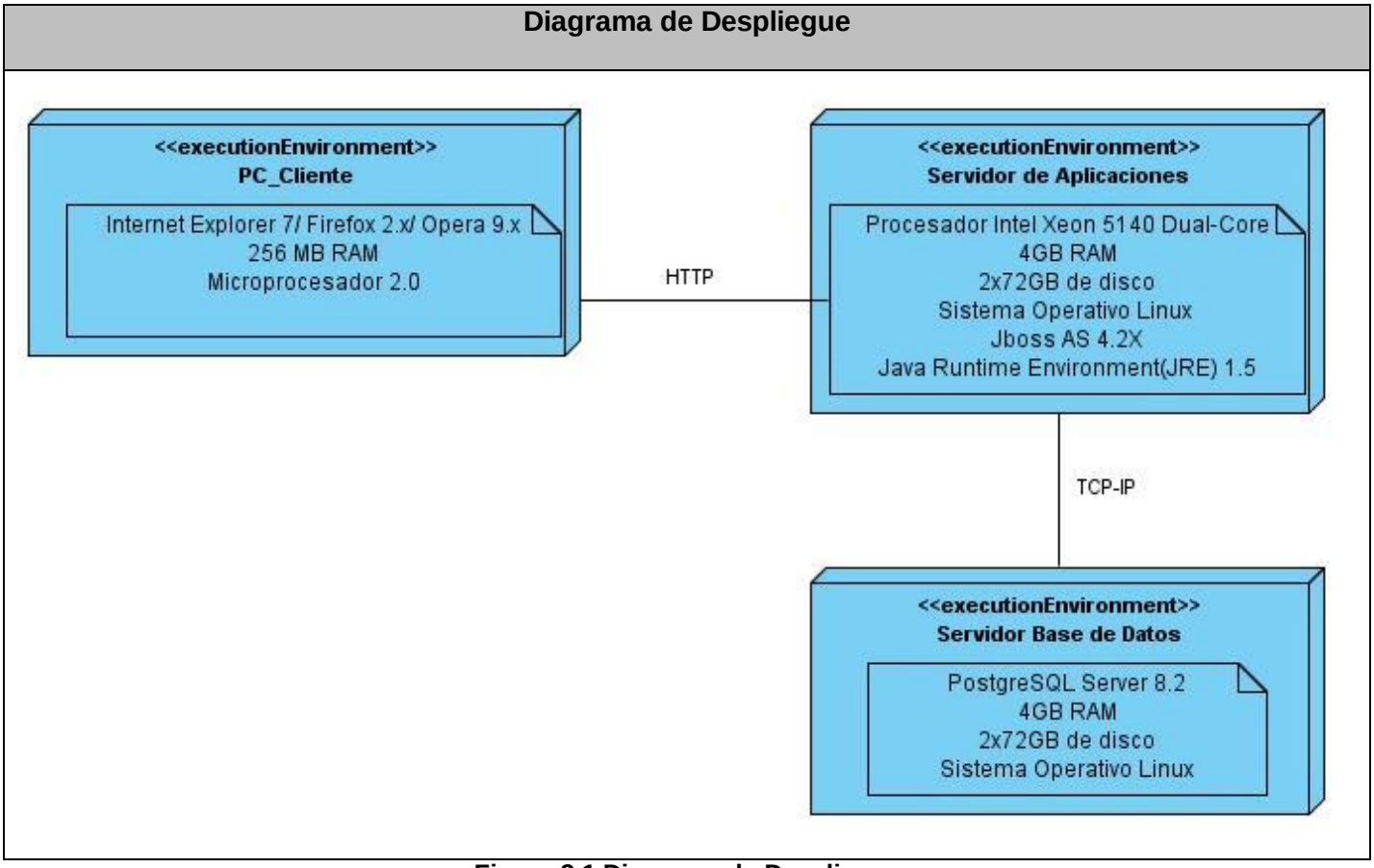

Figura 2.1 Diagrama de Despliegue

2.6 Estrategias de codificación. Estándares y estilos a utilizar

El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación, estos son reglas específicas de cada lenguaje de programación cuyo cumplimiento reduce de forma significativa el riesgo de que los desarrolladores introduzcan errores. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que la aplicación pueda modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Para la implementación del sistema propuesto se utilizaron varios estándares de codificación como son:

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

Identación

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

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 PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay solo una instrucción. Nunca colocar ({) en la línea de un código cualquiera, esto requiere una línea propia.

Comentarios, separadores, líneas, espacios en blanco y márgenes

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

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 identificará el tipo de datos al que se refiere, 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 solo leer se conozca el propósito de las mismas.

Clases y objetos

Objetivo**:** Nombrar las clases e instancias de forma estándar para todas las aplicaciones.

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 CamelICasing**.

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 PascalCasing*. Ejemplo: function BuscarUnidad (). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set.

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, especificando el tipo de datos.

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 solo leerlo se conozca el propósito de los mismos.

Bases de Datos, Tablas, esquemas y Campos

Apariencia de la Base de Datos

Las 2 primeras letras representan el tipo. Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación underscoard y luego el nombre completamente en minúscula, en caso de que sea un nombre compuesto se empleará notación. Los nombres serán cortos y descriptivos.

Ejemplo: bd_balancematerial.

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 las tablas

Las 2 primeras letras representan el tipo. Todas las letras en minúscula. El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizará underscoard para separarlo.

Ejemplo: “tb_producto”.

Tablas que representen Relaciones

Las 2 primeras letras representan el tipo. Todas las letras en minúscula. El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula. Ejemplo: “tr_paciente_enfermedad”.

Tablas que representen nomencladores

Las 2 primeras letras representan el tipo. Todas las letras en minúscula. El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre será corto y descriptivo todo en minúscula. Ejemplo: “tn_color_piel”.

Apariencia de los procedimientos almacenados

Las 2 primeras letras representan el tipo. Todas las letras en minúscula. El nombre a emplear para los procedimientos debe comenzar con el prefijo pa seguido de underscoard y luego debe escribirse todas las letras en minúscula en caso de que sea un nombre compuesto se utilizará underscoard para separarlo.

Ejemplo: pa _ paciente_especialidad.

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 solo 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 identificarán el tipo de datos al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Ejemplo: btnAceptar

***Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula.

Ejemplo: NotacionPascalCasing.

***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.

Tipo de datos	Prefijo	Ejemplo
Int	I	iCantPacientes
flota	F	fPesoPaciente
double	D	dPesoCarro
bool	B	bPacienteActivo
string	S	sNombrePaciente
char	C	cLetra
De tipo enum	E	eSexo
byte	B	bCantDiasPaciente
sbyte	Sb	sbEdadPaciente
short	Sh	shVariableShort
ushort	Us	usVariableUshort
uint	Ui	uiVariableUint
long	L	lVariableLong
ulong	Ul	ulVariableUlong
decimal	Dc	dcVariableDecimal

Objetos	O	oPacienteHistorico
Objetos de tipo Struct	St	stUnaStruct
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 tipos de datos

En este capítulo se ha realizado una completa descripción de la arquitectura, donde se ha tenido en cuenta los requerimientos no funcionales, el análisis de las posibles implementaciones y componentes a reutilizar. Además de la seguridad, la vista de despliegue, la estrategia de codificación, estándares y estilos a utilizar.

Capitulo 3: Descripción y análisis de la solución propuesta.

Haciendo uso de la arquitectura descrita en el capítulo anterior, resulta mucho más fácil la implementación de un sistema que cumpla con los requisitos funcionales y no funcionales. Este capítulo te permite crear una entrada apropiada y un punto de partida para actividades de implementación. Se realiza una valoración de los patrones de diseño, se describen las nuevas clases u operaciones necesarias para desarrollar un subproceso específico. Se expone el modelo de datos y una evaluación de las técnicas de validación, así como la vista de implementación lo cual incluye el diagrama de componentes.

3.1 Modelo de diseño

El diseño es el núcleo técnico de la ingeniería del software durante el cual se desarrollan, revisan y documentan los refinamientos progresivos de la estructura de datos, arquitectura, interfaces y datos procedimentales de los componentes del software. En el diseño se modela el sistema y encuentra su forma para que soporte todos los requisitos y a las restricciones que se le suponen, por lo que se utiliza el modelo de diseño.

El modelo de diseño contiene los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos y las realizaciones de los casos de uso. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objetos, incluyendo sus operaciones y atributos. Con el objetivo de propiciar una mejor comprensión y calidad del diseño fueron aplicados patrones de diseño durante la realización de los componentes del mismo, facilitándose la reusabilidad, extensibilidad y mantenimiento.

Para la realización de dicho modelo se utilizaron patrones de diseño que no son más que soluciones a problemas comunes que se presentan en el diseño de aplicaciones. Constituye una buena práctica de programación implementarlos pues pueden ser reusables aplicándose a diferentes problemas en distintas circunstancias, ahorrando tiempo y mejorando el software haciéndolo más eficiente, dinámico y seguro.

Grasp:

Estos patrones se basan en los principios generales de asignar responsabilidades para el diseño eficiente del software orientado a objetos. Se utilizarán con el objetivo de determinar las responsabilidades que tendrán las diferentes clases que se definan en el diseño. Dentro de este grupo se identifican cinco patrones muy utilizados: experto, creador, alta cohesión, bajo acoplamiento y controlador. Estos patrones son utilizados frecuentemente cuando se requiere implementar una clase que conozca o realice las funcionalidades de acuerdo con la información que contiene en el sistema y su relación de dependencia

con otras clases u objetos. Con este patrón se evitan los riesgos con tareas no planificadas, aumenta la confianza en los sistemas y disminuyen los errores, además de que favorece la reutilización de código.

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

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

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

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

Singlenton:

Este patrón propone que la clase sea la responsable de controlar la existencia de una única instancia. Con esto se consigue que sea imposible de crear nuevas instancias y que el acceso a la instancia única se hace a través de un único punto bien definido, que es gestionado por la propia clase y que puede ser accedido desde cualquier parte del código. Este patrón se manifiesta a través de la clase Entity Manager que se utiliza para manipular las entidades que se generan.

Patrones en la capa de datos

En la capa de datos, y para lograr una mayor flexibilidad y por tanto asimilar mejor los cambios futuros que se produzcan en la base de datos se utilizan los siguientes patrones:

- Active record: consiste en que un objeto envuelve una tupla de una estructura de datos de un recurso externo, como una fila en una base de datos, y adiciona alguna lógica del dominio a dicho objeto.
- Identity field: la idea fundamental de este patrón es que guarda el campo id de la base de datos en un objeto para mantener la relación entre el objeto cargado en memoria y la fila correspondiente en la base de datos.
- Foreign Key Mapping: consiste en mapear una asociación entre objetos por cada relación entre tablas por claves foráneas.

- Query Object: mantener un objeto que permita hacer consultas a la base de datos.
- Lazy load: el objeto que mapea una fila en la base de datos no necesariamente debe cargar toda la información desde un principio. Debe conocer la forma de obtenerla y cargarla solo cuando sea necesario.

Estructuración

En UML, un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. Los Paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre ellos. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

Para el sistema propuesto se ha decidido organizar la realización de los casos de uso de forma que cumpla con los patrones de diseño y de arquitectura explicados previamente. Para ello se cuenta con tres paquetes: Sesiones, Entidades y Vista, que contendrán las clases controladoras, las entidades o modelo, y la vista o presentación respectivamente. El paquete Sesiones contendrá a su vez tres paquetes que agruparán: las clases controladoras autogeneradas por las herramientas de generación de código, las personalizaciones de las clases autogeneradas y las clases controladoras que no son autogeneradas agrupadas por procesos y por casos de uso. El paquete Entidades contiene las entidades autogeneradas y el paquete Vista todas las interfaces de la aplicación.

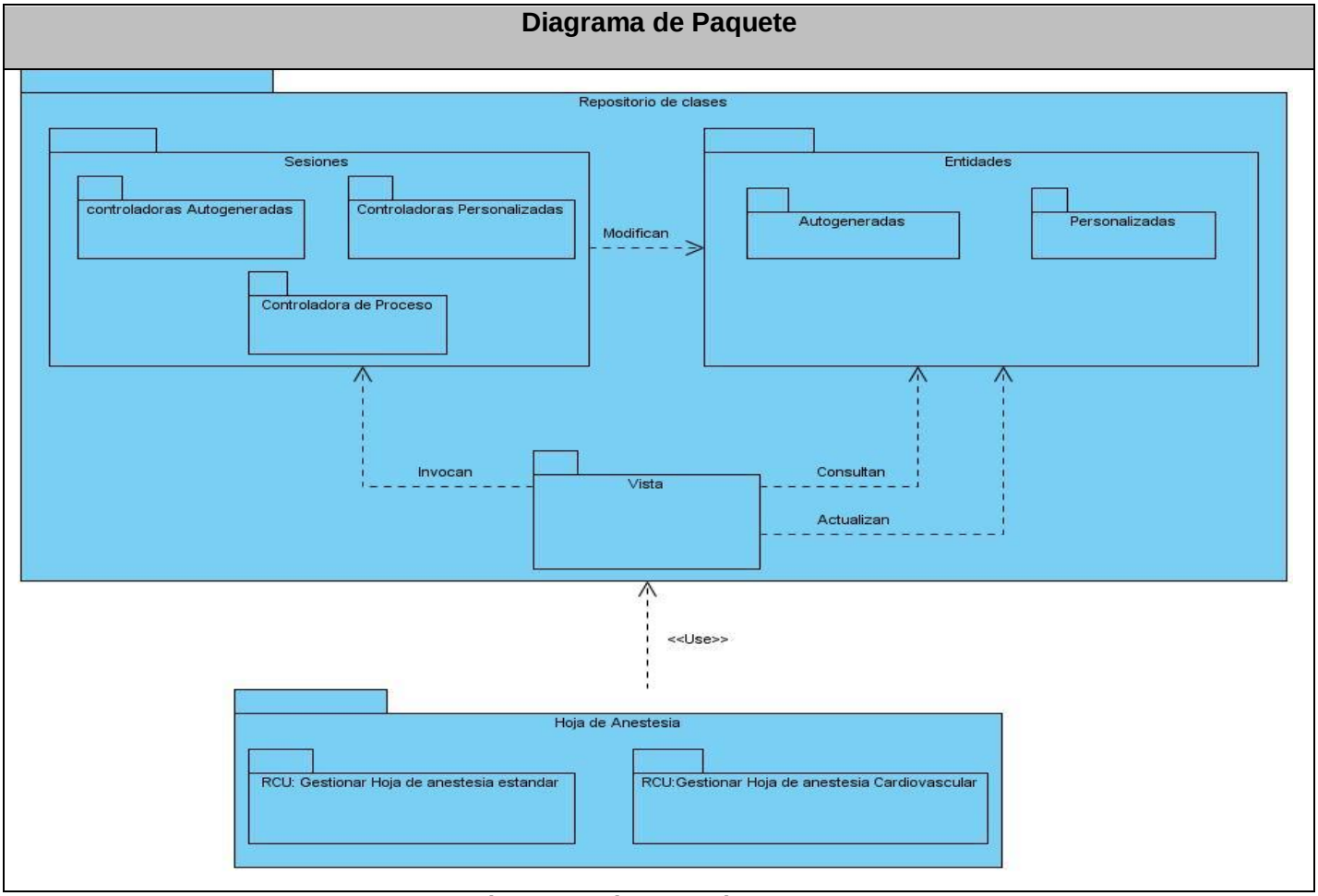

Figura 3.1 Diagrama de Paquete

3.2 Diagramas de clases del diseño

Una clase de diseño es una abstracción de una clase o construcción en la implementación del sistema. Los diagramas de clases del diseño exponen un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema y son importantes no solo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

Los diagramas de clases del diseño fueron modelados incluyendo los estereotipos web. Un estereotipo es una sutileza que permite definir un nuevo significado de la semántica para el elemento a modelar. A continuación se describen los estereotipos web utilizados para modelar los diagramas de clases:

Página servidora: Representa la página web que tiene código que se ejecuta en el servidor y que interactúa con recursos en el mismo. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.

Página cliente: Una instancia de página cliente es una página web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para páginas con cualquier función dentro de esta.

Formulario: Colección de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (input boxes, text áreas, radio buttons, check boxes y hidden fields). No tienen operaciones.

Build: Representa una asociación especial que relaciona las páginas clientes con las páginas servidoras, de forma general se expresa como que las páginas que se encuentran en el servidor construyen las páginas en el cliente. Debe ser una relación direccional, donde una página servidor puede construir una o más páginas cliente.

Submit: Es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.

Redirect: La página de servidor además de construir una página cliente puede redireccionar el procesamiento a otra página. Esto se representa con una asociación con el estereotipo <<redirect>>.

Diagrama de clases

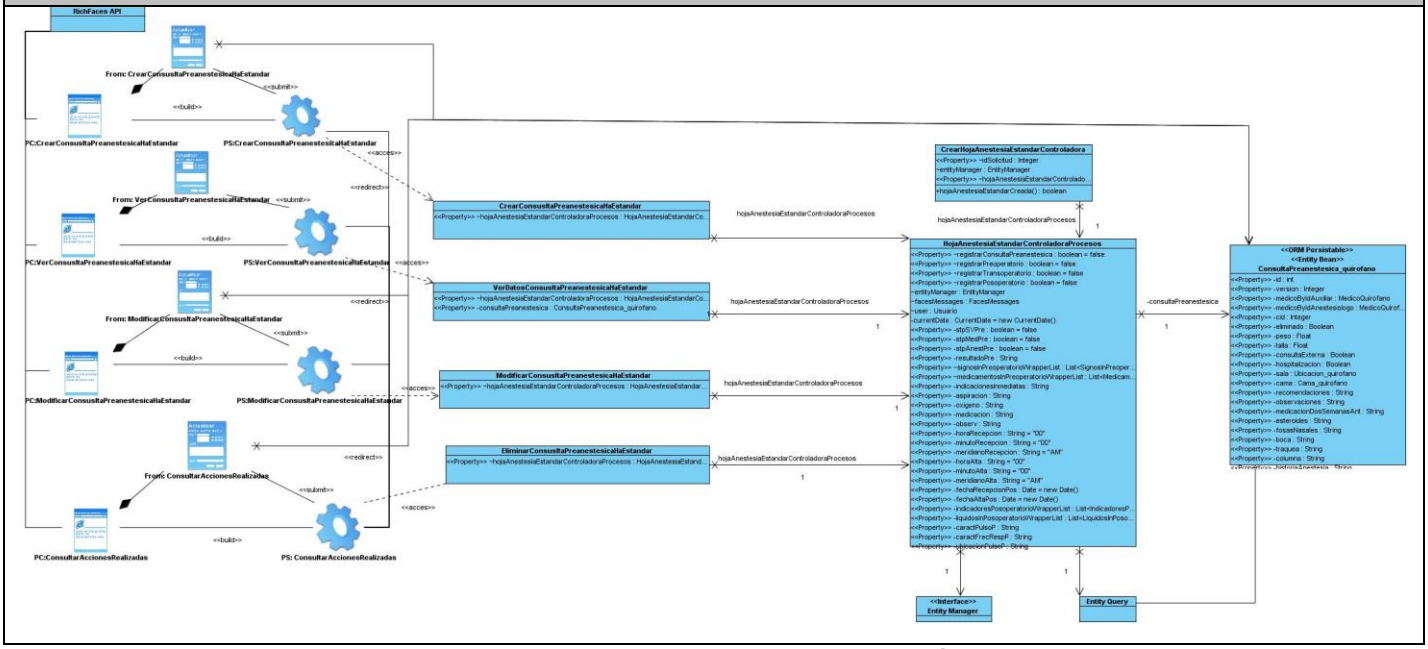

Figura 3.2 DCD_Gestionar consulta preanestésica HAE

Diagrama de secuencia

Es un diagrama de interacción que destaca la ordenación temporal de los mensajes. Se forma colocando en primer lugar los objetos que participan en la interacción en la parte superior del diagrama, a lo largo del eje X. Normalmente, se coloca a la izquierda el objeto que inicia la interacción, y los objetos subordinados a la derecha. A continuación, se colocan los mensajes que estos objetos envían y reciben a lo largo del eje Y, en orden de sucesión en el tiempo, desde arriba hasta abajo.

Esto ofrece al lector una señal visual clara del flujo de control a lo largo del tiempo. Los objetos poseen una línea de vida que es la línea discontinua vertical que representa la existencia de un objeto a lo largo de un período de tiempo y un foco de control que es un rectángulo delgado y estrecho que representa el período de tiempo durante el cual un objeto ejecuta una acción, bien sea directamente o a través de un procedimiento subordinado. La parte superior del rectángulo se alinea con el comienzo de la acción; la inferior se alinea con su terminación (y puede marcarse con un mensaje de retorno).

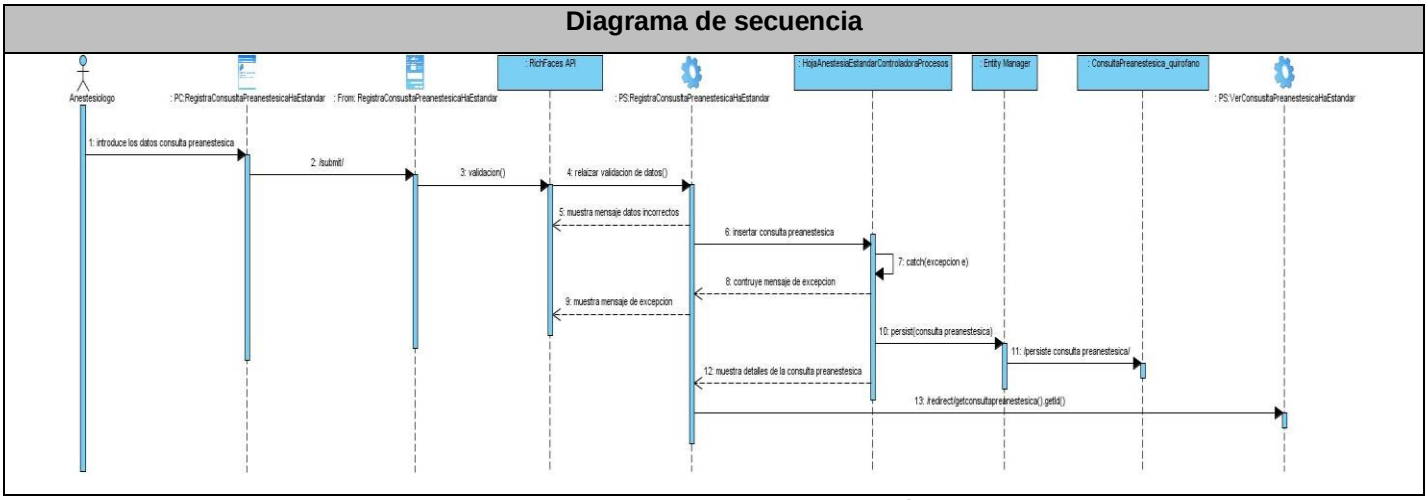

Figura 3. 3 DS_Crear consulta preanestésica HAE

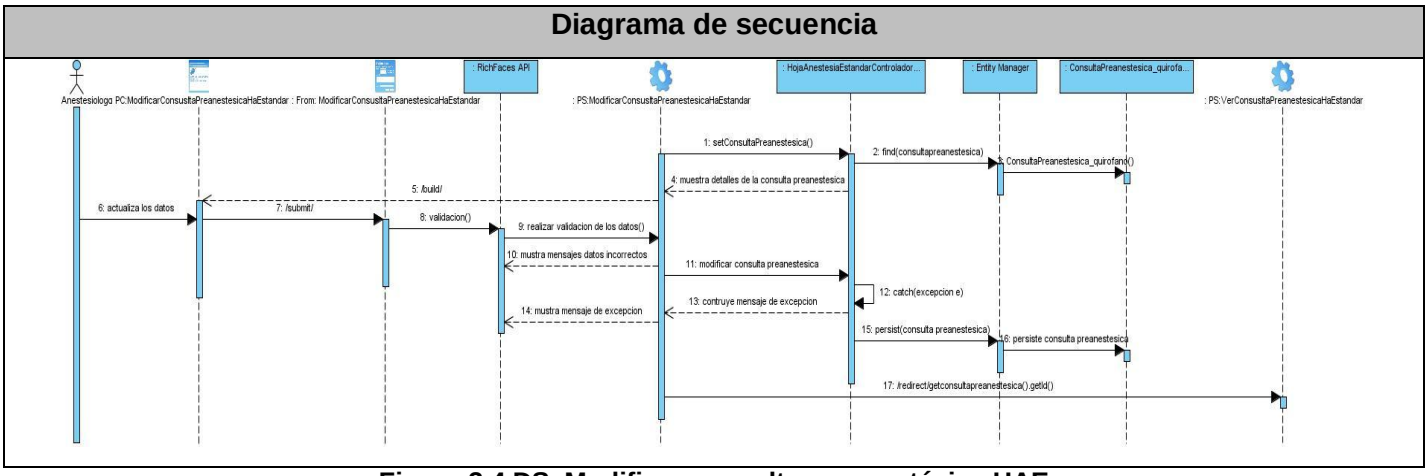

Figura 3.4 DS_Modificar consulta preanestésica HAE

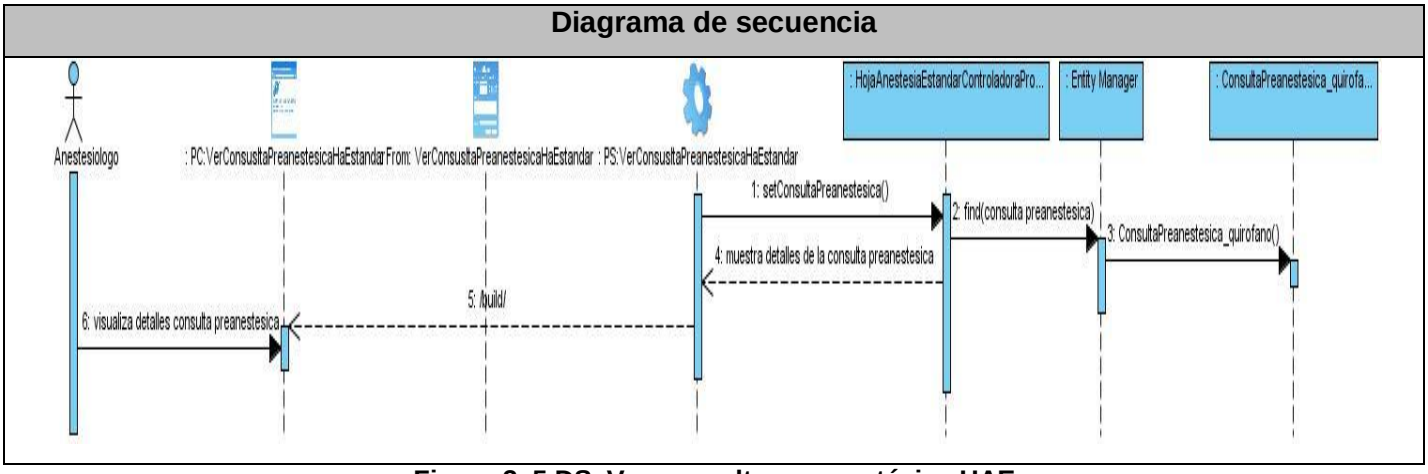

Figura 3. 5 DS_Ver consulta preanestésica HAE

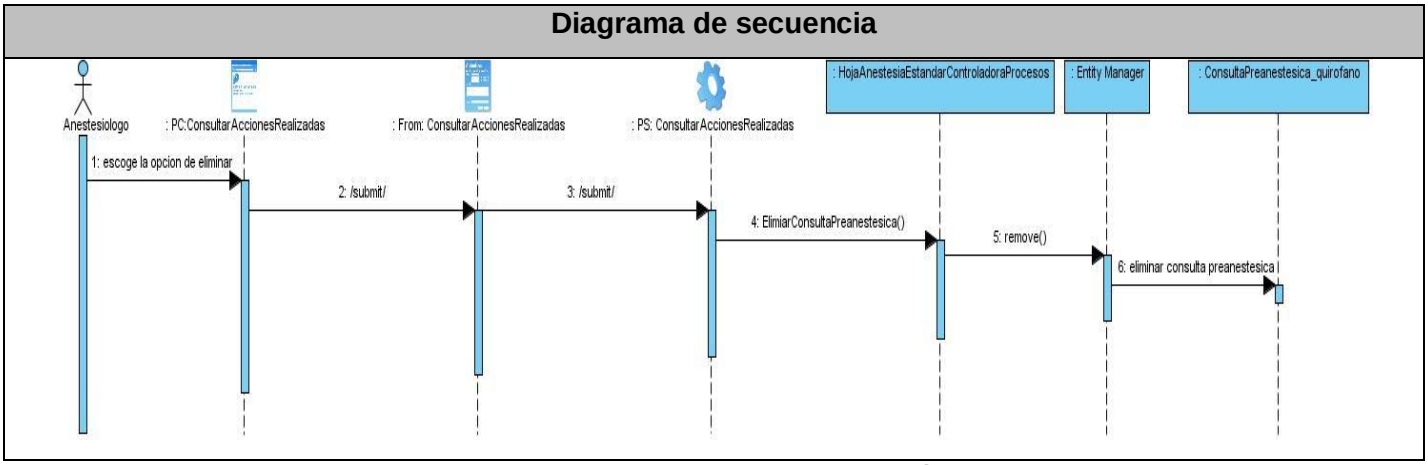

Figura 3. 6 DS_Eliminar consulta preanestésica HAE

3.3 Descripción de las clases u operaciones necesarias.

Nombre: CrearHojaAnestesiaEstandarControladora	
Tipo de clase: Controladora	
Atributo	**Tipo**
idSolicitud	Integer
EntityManager	entityManager
HojaAnestesiaEstandarControladoraProcesos	HojaAnestesiaEstandarControladoraProcesos
Para cada responsabilidad:	
Nombre:	hojaAnestesiaEstandarCreada
Descripción:	Comprobar si se creó la hojaAnestesiaEstandarCreada

Tabla 3.1 CrearHojaAnestesiaEstandarControladora

Nombre: ConsultaPreanestesica_quirofano	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	Integer
version	Integer
medicoByIdAuxiliar	MedicoByIdAuxiliar
medicoByIdAnestesiologo	MedicoByIdAnestesiologo
cid	integer
eliminado	boolean
peso	float
talla	float
consultaExterna	boolean

hospitalizacion	boolean
sala	Ubicacion_quirofano
cama	Cama_quirofano
recomendaciones	string
observaciones	string
medicacionDosSemanasAnt	string
esteroides	string
fosasNasales	string
boca	string
traquea	string
columna	string
historiaAnestesia	string
diagnosticoMedico	DiagnosticoMedico
cipProcedimiento	CipProcedimiento
riesgoQuirurgico	RiesgoQuirurgico
estadoFisico	EstadoFisico
hojaAnestesiaEstandar	HojaAnestesiaEstandar
consultaPaExamenCv	ConsultaPaExamenCv
consultaPaExamenResp	ConsultaPaExamenResp
consultaPaInterrogatorio	ConsultaPaInterrogatorio
consultaPaLab	ConsultaPaLab
consultaPreanestesicaHabs	ConsultaPreanestesicaHabs

Para cada responsabilidad:	
Nombre:	ConsultaPreanestesica_quirofano
Descripción:	Constructor

Tabla 3. 2 ConsultaPreanestesica_quirofano

Nombre: HojaAnestesiaEstandarControladoraProcesos	
Tipo de clase: Controladora	
Atributo	**Tipo**
crearConsultaPreanestesica	CrearConsultaPreanestesica
entityManager	EntityManager
facesMessages	FacesMessages
user	User
aspiracion	string
oxigeno	string
medicacion	string
observ	string
caractPulsoP	string
caractFrecRespP	string
ubicacionPulsoP	string
horaComienzoAnestesia	string
horaFinAnestesia	string
horaComienzoCirugia	string
horaFinCirugia	string
codificacion	string

sueroTotal	integer
sangreTotal	integer
plasmaTotal	integer
anestesiaRegional	boolean
estaFisco	string
metodo	string
agente	string
hojaAnestesiaEstandar	HojaAnestesiaEstandar
medicoAuxiliarAnestesiaTrans	MedicoQuirofano
medicoCirujanoTrans	MedicoQuirofano
medicoAnestesiologoTrans	MedicoQuirofano
enfermeraQuirofanoTrans	EnfermeraQuirofano
enfermeraTransfusionista	EnfermeraQuirofano
enfermeraAuxiliarTransfusionista	EnfermeraQuirofano
enfermeraQuirofanoGenerico	EnfermeraQuirofano

Para cada responsabilidad:

Nombre:	SalvarProceso
Descripción:	Salva los datos de la consulta preanestesica, preoperatorio, transoperatorio y posoperatorio.
Nombre:	VerPosoperatorioCreado
Descripción:	Muestra los datos registrado del posoperatorio
Nombre:	CrearPreoperatorio
Descripción:	Crea el preoperatorio

Nombre:	CrearTransoperatorio
Descripción:	Registra los datos introducidos en el transoperatorio
Nombre:	VerConsultaPreanestesicaPCreada
Descripción:	Muestra los datos registrado de la consulta preanestesica

HojaAnestesiaEstandarControladoraProcesos

Nombre: CrearConsusltaPreanestesicaHaEstandar	
Tipo de clase: Controladora	
Atributo	**Tipo**
hojaAnestesiaEstandarControladoraProcesos	HojaAnestesiaEstandarControladoraProcesos
Para cada responsabilidad:	
Nombre:	getHojaAnestesiaEstandarControladoraProcesos()
Descripción:	Devuelve el objeto creado de la HojaAnestesiaEstandarControladoraProcesos
Nombre:	set HojaAnestesiaEstandarControladoraProcesos()
Descripción:	Inserta el objeto creado de la HojaAnestesiaEstandarControladoraProcesos

Tabla 3. 3 CrearConsusltaPreanestesicaHaEstandar

Nombre: VerDatosConsusltaPreanestesicaHaEstandar	
Tipo de clase: Controladora	
Atributo	**Tipo**
hojaAnestesiaEstandarControladoraProcesos	HojaAnestesiaEstandarControladoraProcesos
consultaPreanestesica	ConsultaPreanestesica
Para cada responsabilidad:	
Nombre:	getHojaAnestesiaEstandarControladoraProcesos()

Descripción:	Devuelve el objeto creado de la HojaAnestesiaEstandarControladoraProcesos
Nombre:	set HojaAnestesiaEstandarControladoraProcesos()
Descripción:	Inserta el objeto creado de la HojaAnestesiaEstandarControladoraProcesos

Tabla 3. 4 VerDatosConsusltaPreanestesicaHaEstandar

Nombre: ModificarConsusltaPreanestesicaHaEstandar	
Tipo de clase: Controladora	
Atributo	**Tipo**
hojaAnestesiaEstandarControladoraProcesos	HojaAnestesiaEstandarControladoraProcesos
Para cada responsabilidad:	
Nombre:	getHojaAnestesiaEstandarControladoraProcesos()
Descripción:	Devuelve el objeto creado de la HojaAnestesiaEstandarControladoraProcesos
Nombre:	set HojaAnestesiaEstandarControladoraProcesos()
Descripción:	Inserta el objeto creado de la HojaAnestesiaEstandarControladoraProcesos

Tabla 3. 5 ModificarConsusltaPreanestesicaHaEstandar

Nombre: EliminarConsusltaPreanestesicaHaEstandar	
Tipo de clase: Controladora	
Atributo	**Tipo**
hojaAnestesiaEstandarControladoraProcesos	HojaAnestesiaEstandarControladoraProcesos
Para cada responsabilidad:	
Nombre:	getHojaAnestesiaEstandarControladoraProcesos()
Descripción:	Devuelve el objeto creado de la HojaAnestesiaEstandarControladoraProcesos
Nombre:	set HojaAnestesiaEstandarControladoraProcesos()
Descripción:	Inserta el objeto creado de la HojaAnestesiaEstandarControladoraProcesos

Tabla 3. 6 EliminarConsusltaPreanestesicaHaEstandar

3.4 Modelo de datos

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. En un enfoque más amplio, un modelo de datos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí.

Es usado para describir la estructura lógica y física de la información persistente manejada por el sistema. Controla que la información básica almacenada para cada atributo o propiedad de un concepto sea válida. Está formado por objetos (entidades que existen y que se manipulan), atributos (características básicas de estos objetos) y relaciones (forma en que enlazan los distintos objetos entre sí).

Modelo de Datos

3.5 Descripción de las tablas

Nombre: ConsultaPreanestesica_quirofano		
Descripción: Tabla para registrar los datos de la consulta preanestesica de la hoja de anestesia estándar		
Atributo	**Tipo**	**Descripción**
id	Integer	Identificador de la tabla
Versión	Integer	Atributo para persistir o actualizar la entidad
medicoByIdAuxiliar	MedicoQuirofano	Objeto de la entidad MedicoQuirofano. Médico auxiliar
medicoByIdAnestesiologo	MedicoQuirofano	Objeto de la entidad MedicoQuirofano. Médico anestesiólogo
cid	Integer	Identificador de modificaciones registradas en la bitácora
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado
peso	Float	Peso del paciente
talla	Float	Talla del paciente
consultaExterna	Boolean	Permite verificar si el paciente es remitido desde la consulta externa
hospitalizacion	Boolean	Permite verificar si el paciente es remitido desde la hospitalización
sala	Ubicacion_quirofano	Objeto de la entidad Ubicacion_quirofano. Sala en la que está ubicado el paciente
cama	Ubicacion_quirofano	Objeto de la entidad Ubicacion_quirofano. Cama en la que está ubicado el paciente
recomendaciones	String	Recomendación al paciente
observaciones	String	Conclusiones una vez revisado el paciente
medicacionDosSemanasAnt	String	Medicación que tenía el paciente dos semanas antes
esteroides	String	Esteroides que puede usar
fosasNasales	String	Descripción del estado de las fosas nasales
boca	String	Descripción del estado de la boca
traquea	String	Descripción del estado de la tráquea
columna	String	Descripción del estado de la columna
historiaAnestesia	historiaAnestesia	Descripción de la historia del proceso anestésico

diagnosticoMedico	DiagnosticoMedico	Objeto de la entidad DiagnosticoMedico. Datos del diagnóstico médico
cipProcedimiento	CipProcedimiento	Objeto de la entidad CipProcedimiento
riesgoQuirurgico	RiesgoQuirurgico	Objeto de la entidad CipProcedimiento. Datos del riesgo quirúrgico
estadoFisico	EstadoFisico_quirofano	Objeto de la entidad EstadoFisico_quirofano
hojaAnestesiaEstandar	HojaAnestesiaEstandar_quirofano	Objeto de la entidad HojaAnestesiaEstandar_quirofano
consultaPaExamenCv	ConsultaPaExamenCv_quirofano	Objeto de la entidad ConsultaPaExamenCv_quirofano
ConsultaPaExamenResp	ConsultaPaExamenResp_quirofano	Objeto de la entidad ConsultaPaExamenResp_quirofano
ConsultaPaInterrogatorio	ConsultaPaInterrogatorio_quirofano	Objeto de la entidad ConsultaPaInterrogatorio_quirofano
ConsultaPaLab	ConsultaPaLab_quirofano	Objeto de la entidad ConsultaPaLab_quirofano
ConsultaPreanestesicaHab	ConsultaPreanestesicaHab_quirofano	Objeto de la entidad ConsultaPreanestesicaHab_quirofano

Tabla 3. 7 ConsultaPreanestesica_quirofano

3.6 Valoración de las Técnicas de validación.

Hibernate Validation

Validar los datos es una tarea común que se produce a través de cualquier aplicación, desde la capa de presentación a la capa de persistencia. A menudo, la misma lógica de validación es implementada en cada capa, resultando lento y propenso a errores. Para evitar la duplicación de estas validaciones en cada capa, los desarrolladores a menudo utilizan los paquetes de validaciones lógicas en el modelo de dominio, llenando las clases de dominio con código de validación que es en realidad metadatos sobre la propia clase.

JSR 303(Bean Validation) define un modelo de metadatos y un API para la validación de la entidad. La fuente predeterminada de metadatos son las anotaciones, con la posibilidad de redefinir y ampliar los metadatos a través del uso de XML. El API no está vinculado a un nivel específico de la aplicación o modelo de programación. Hibernate Validator es la implementación de referencia de esta JSR. Hibernate Validator se integra con Hibernate mediante la aplicación de las restricciones en el esquema de base de

datos, se asegura de su validez de la entidad antes de insertar o de actualizaciones de los datos. Se puede utilizar con cualquier proveedor de persistencia Java, no solo de Hibernate.

En Hibernate Validator la comprobación de restricciones puede ser activada mediante programación, por ejemplo, la capa de negocio. JBoss Seam hace uso de esta capacidad y la integra con JSF, proporcionando de un extremo a otro la validación del nivel de presentación hasta las restricciones de la base de datos con una definición única y la declaración de las normas de integridad. Hibernate Validator está plenamente internacionalizado. Las principales características son:

- Se definen las validaciones muy fácilmente con anotaciones (es posible anotar tanto los atributos como los getters).
- Tener un conjunto predefinido de validaciones típicas, conjunto que pueden extender fácilmente con las propias validaciones.
- El sistema soporta internacionalización: Ya trae mensajes de error traducidos a diez idiomas. Estos mensajes los pueden cambiar fácilmente, simplemente escribiendo el propio fichero de propiedades y sobrescribiendo los mensajes que interesen.
- Se integra directamente con Hibernate, y en general con cualquier OR Mapping, de forma que antes de hacer una inserción o actualización se validarán los objetos.
- Con el uso de Hibernate, las validaciones que indiquen se tendrán en cuenta a la hora de generar el DDL (los scripts de creación de la base de datos).

Con la utilización de Hibernate Validation la aplicación se libraría de encontrar repetidas las mismas validaciones en diferentes formularios que tratan con los mismos objetos de negocio. Y se aseguraría de cumplir con el principio DRY (Don't repeat yourself – no te repitas).

3.7 Vista de Implementación

Diagrama de Componente

Los componentes son una parte física y reemplazable de un sistema que está conformado por un conjunto de interfaces y proporciona la realización de dicho conjunto. Se usan para modelar los elementos físicos que pueden hallarse en un nodo por lo que empaquetan elementos como clases, colaboraciones e interfaces; poseen relaciones de traza con los elementos del modelo de diseño que implementan.

Un diagrama de componentes muestra la organización y las dependencias lógicas entre un conjunto de componentes software, sean estos de código fuente, librerías, binarios o ejecutables. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación,

especificando los subsistemas de implementación y sus dependencias a la hora de importar el código y organizando los subsistemas de implementación en capas. Para mantener una trazabilidad directa desde el diagrama de paquetes de diseño al diagrama de componentes, se ha establecido un paquete de componentes por cada paquete del diseño.

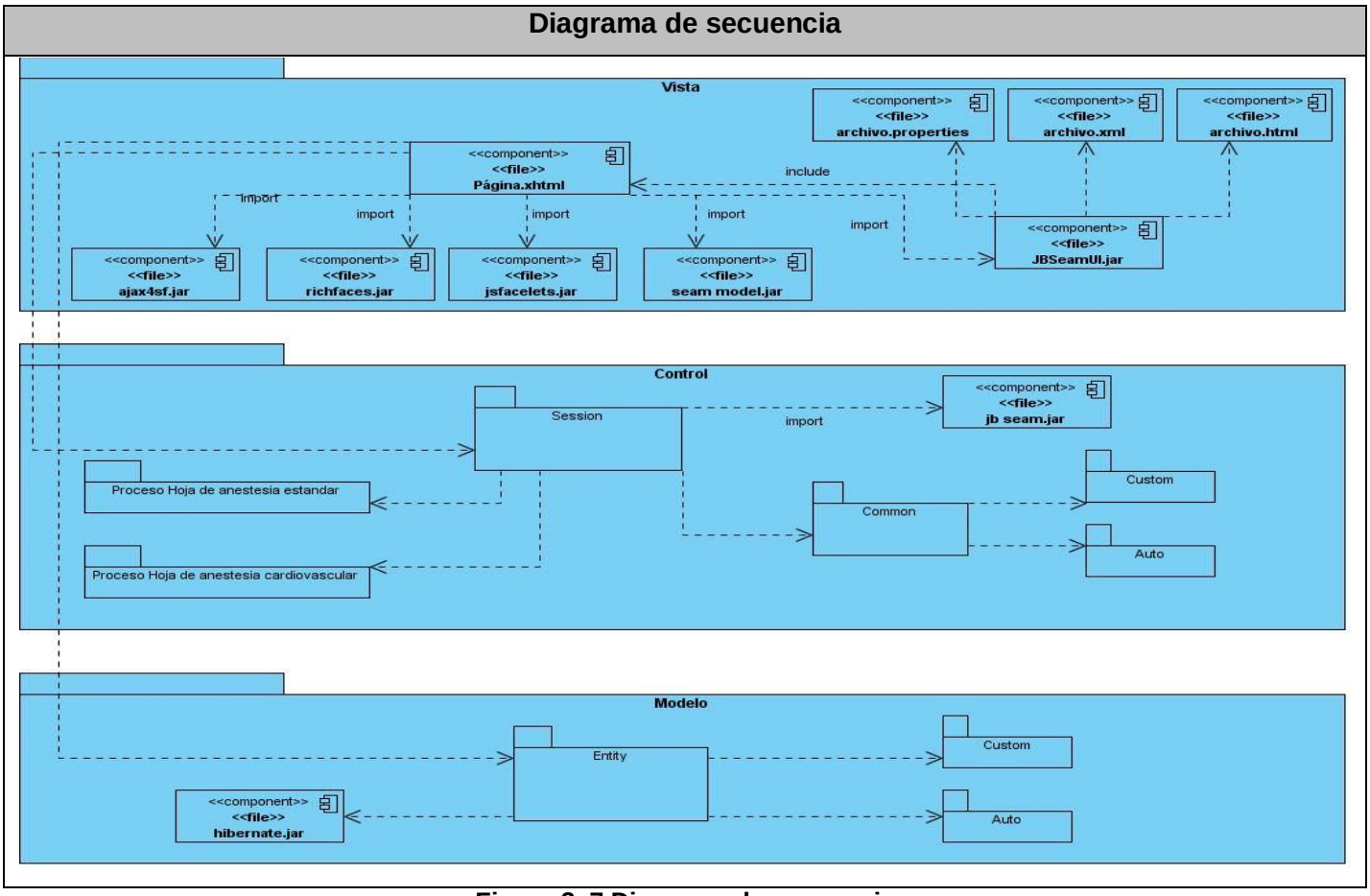

Figura 3. 7 Diagrama de secuencia

3.8 Descripción de los algoritmos no triviales a implementar. Análisis de complejidad de los mismos

Un algoritmo es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien lo ejecute. Dado un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.

Cuando se solucionan problemas mediante la construcción de un algoritmo, normalmente, se ataca el problema desde distintos puntos de vista, aplicando distintas estrategias, y por tanto, llegando a soluciones algorítmicas distintas. Desde el punto de vista computacional, es necesario disponer de alguna forma de comparar una solución algorítmica con otra, para conocer cómo se comportarán cuando se implementen, especialmente al atacar problemas grandes. La complejidad algorítmica es una métrica teórica que se aplica a los algoritmos en este sentido. Es un concepto fundamental para todos los programadores.

Para lograr el desarrollo de los procesos de la hoja de anestesia estándar y cardiovascular, del módulo del quirófano del sistema de información hospitalaria alas HIS se implementaron algoritmos de alta complejidad como son:

- Ver datos de la hoja de anestesia estándar.
- Ver datos de la hoja de anestesia cardiovascular.
- Crear hoja de anestesia estándar.
- Crear hoja de anestesia cardiovascular.
- Crear consulta preanestésica.
- Ver detalles de la consulta preanestésica.
- Modificar consulta preanestésica.
- Crear consulta preanestésica cardiovascular.
- Ver detalles de la consulta preanestésica cardiovascular.
- Modificar consulta preanestésica cardiovascular.

3.9 Selección de las estructuras de datos apropiadas para la implementación de estos algoritmos.

Una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Define la organización e interrelación de éstos y un conjunto de operaciones que se pueden realizar sobre ellos. Un dato elemental es la mínima información que se tiene en un sistema.

En una estructura de datos se identifican dos niveles, en el primero se identifican la colección de elementos a agrupar y en el segundo se piensa en el lenguaje de programación en específico que se va a utilizar.

El lenguaje de programación Java contiene un paquete java.util que consta de 34 clases y 13 interfaces que implementan algunas de las estructuras de datos más comunes. Una de estas estructuras es el Arraylist que es una lista con la que se ordena correctamente un conjunto de elementos, la misma tiene varios métodos y es una clase que implementa le interfaz.

3.10 Descripción de las clases que se utilicen para representar computacionalmente dicha estructura.

Las aplicaciones frecuentemente necesitan almacenar un grupo de datos en un solo objeto. Los arrays sirven bien para este propósito, pero algunas veces es necesario incrementar o reducir dinámicamente el número de elementos del array, o hacer que contenga distintos tipos de datos. ArrayList es una de las muchas clases del Collection Framework, que proporciona un conjunto de interfaces y clases bien diseñadas para almacenar y manipular grupos de datos como una sola unidad, una colección. Un ArrayList contiene tantos objetos como se necesiten, tiene varios constructores, dependiendo de cómo construya el ArrayList. Un objeto solo contiene referencias a objetos. Para almacenar tipos primitivos como double, long, o float, se utiliza una clase envoltura.

Esta clase provee una serie de métodos para manipular la lista, seguidamente se explican algunos de estos métodos:

- Size: devuelve el número de elementos en esta lista.
- toArray: Devuelve una matriz que contiene todos los elementos en esta lista en el orden correcto.
- Remove: Elimina el elemento en la posición especificada en esta lista. Cambios de todos los elementos posteriores a la izquierda.
- isEmpty: Comprueba si esta lista no tiene elementos.
- indexOf: Las búsquedas de la primera aparición de los argumentos dados, las pruebas para la igualdad mediante el método equals.
- get: Devuelve el elemento en la posición especificada en esta lista.
- add: Inserta el elemento especificado en la posición especificada en esta lista. Cambia el elemento actualmente en esa posición (si hay) y todos los elementos posteriores a la derecha.

En este capítulo se ha descrito y analizado toda propuesta de solución del trabajo. Se realiza una valoración del diseño: dentro de ello los patrones GRASP, Singlenton y los de la capa de datos. Han sido presentadas las clases del diseño que participan en la realización de los casos de uso de la aplicación propuesta, los diagramas de secuencia del diseño y la descripción textual de cada una de las clases controladoras donde se implementan las principales funcionalidades.

Capítulo 4: Modelo de prueba

Es vital para el éxito de un software que tenga un buen modelo de prueba. Para ello es necesario comprobar que el software realice correctamente las tareas indicadas en la especificación del problema. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral, para así llegar al objetivo.

Se considera una buena práctica el que las pruebas sean efectuadas por alguien distinto al desarrollador que la programó; sin perjuicio de lo anterior el programador debe hacer sus propias pruebas. Una forma de organizar un área de pruebas, es que esté compuesta por personal inexperto y que desconozca el tema de pruebas, de esta forma, se evalúa que la documentación entregada sea de calidad, que los procesos descritos son tan claros que cualquiera puede entenderlos y el software hace las cosas tal y como están descritas. Para ello se realiza una caracterización en este capítulo de las pruebas de caja negra como método a utilizar, además de definir y describir los casos de prueba.

4.1 Prueba de caja negra

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. Está especialmente indicada en aquellos módulos que van a tener interacción con el usuario y se apoyan en la especificación de requisitos del módulo. Los datos de prueba se escogerán atendiendo a las especificaciones del problema, a fin de verificar que el programa corra bien.

Con este tipo de prueba se intentará encontrar:

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

Con los casos de prueba de caja negra se pretende demostrar además que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.

- La integridad de la información externa se mantiene.

Métodos de prueba de caja negra

Métodos de prueba basados en grafos: en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

Partición equivalente: presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Análisis de valores límite: los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite. El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, lleva la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, obtiene casos de prueba también para el campo de salida.

Adivinando el error: dado un programa particular, se conjetura, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso intuitivo. La idea básica es enumerar una lista de errores posibles o de situaciones propensas a error y después escribir los casos de prueba basados en la lista.

Descripción de los casos de pruebas

Caso de prueba: Conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Es una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que se debe probar.

Caso de prueba: Crear la consulta preanestésica de la hoja de anestesia estándar.

Escenarios del Crear consulta preanestésica	Descripción de la funcionalidad	Flujo Central
EC 1: Crear consulta preanestésica	Crear una consulta preanestésica satisfactoriamente.	Se selecciona la opción Crear consulta preanestésica. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se registran los datos de la consulta preanestésica en la hoja de anestesia estándar creada. Muestra al usuario una vista previa de la consulta preanestésica. Ver DCP: Ver detalles consulta preanestésica estándar.
EC 2: Cancelar operación	Cancelar la opción de Crear consulta preanestésica.	Se selecciona la opción Crear consulta preanestésica. 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 haber incompletos muestra un indicador sobre los campos incompletos.	Se selecciona la opción Crear consulta preanestésica. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.
EC 4: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incorrectos muestra un indicador sobre los campos incorrectos.	Se selecciona la opción Crear consulta preanestésica. Se introducen los datos incorrectos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incorrectos.

4. 1 CP_Crear la consulta preanestésica

Id del escenario	EC 1	EC 2	EC 3	EC 3
Escenario	Crear consulta preanestésica	Cancelar operación	Existen datos incompletos	Existen datos incorrectos
Variable 1 (Estado físico)	Estado físico I	Estado físico I	Sin seleccionar	
Variable 2 (Riesgo quirúrgico)	Riesgo quirúrgico I	Riesgo quirúrgico I	Sin seleccionar	
Variable 3 (Diastólica)	80	80	Sin seleccionar	
Variable 4 (Sistólica)	120	120	Sin seleccionar	
Variable 5 Pulso (Valor)	75	75	Sin seleccionar	
Variable 6 Pulso (Características)	Pulso fuerte y saltón	Pulso fuerte y saltón	Sin seleccionar	
Variable 7 (Ubicación)	Pulso radical	Pulso radical	Sin seleccionar	
Variable 8 Frecuencia respiratoria (Valor)	100	100	Sin seleccionar	
Variable 9 Frecuencia respiratoria (Características)	Disnea	Disnea	Sin seleccionar	
Botón 1 (Aceptar)	NA		NA	NA
Botón 2 (Cancelar)		NA		
Respuesta del Sistema	Crear consulta preanestésica y se visualiza la interfaz ver detalles.	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.	Muestra un indicador sobre los campos incorrectos.
Resultado de la Prueba				

4. 2 SC_Crear consulta preanestésica

Con este capítulo se ha comprendido la importancia de las pruebas para detectar que no existan errores en el software. La mayoría de los usuarios de programas extensos no se interesan en los detalles del funcionamiento del programa, lo que buscan son las respuestas. Es por ello la importancia que reviste la utilización de pruebas para desarrollar un software con la máxima calidad, para brindar un servicio de excelencia.

Conclusiones

Una vez finalizada la investigación referida al área del Bloque Quirúrgico específicamente de las hojas de anestesia estándar y cardiovascular en las instituciones hospitalarias se llegaron a las siguientes conclusiones:

- La mayoría de las aplicaciones existentes a nivel mundial vinculadas al campo de acción tienen como limitante ser productos propietarios. El sistema desarrollado con tecnologías libres, no cubre todas las funcionalidades y prestaciones tecnológicas que se necesitan desarrollar.
- La arquitectura, tecnologías y herramientas definidas, favorecen el desarrollo de la aplicación web, con una interfaz sencilla, robusta y flexible que gestiona correctamente la información generada por los procesos de la hoja de anestesia estándar y cardiovascular.
- El desarrollo del proceso de la hoja de anestesia estándar y la hoja de anestesia cardiovascular del módulo del Bloque Quirúrgico del Sistema de Información hospitalaria alas HIS, facilitará la gestión de la información en los quirófanos.
- La utilización de las pautas definidas garantizan uniformidad visual de todas las interfaces del sistema.

Recomendaciones

Para lograr la continuidad de este trabajo debido a la importancia que tiene el área de Bloque Quirúrgico para las instituciones hospitalarias, se recomienda a partir de la investigación realizada:

- Realizar el piloto del sistema para detectar las posibles fallas en la aplicación y mejorarla posteriormente.
- Comunicar el sistema con equipos médicos que mantienen un monitoreo constante del estado del paciente para lograr así datos precisos del la situación actual del mismo.

Referencias Bibliográficas

1. Victoria**.** Definición ABC guia en la red. [En línea] [Citado el: 2010 de Enero de 26.] http://www.definicionabc.com/tecnologia/navegador.php.
2. Suárez González, Hécto**.** Java Hispano. [En línea] 2003. [Citado el: 2010 de Enero de 28.] http://www.javahispano.org/contenidos/es/manual_hibernate/.
3. Sánchez Suárez, José Manuel**.** Adictos al trabajo. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=migrateJSF2Facelets.
4. Rojo, J. Oscar. Asociación de usuarios de GNU/Linux de castilla y León. [En línea] 2003. [Citado el: 2010 de Enero de 26.] http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos.
5. Ramos, Juan Alonso**.** Adictos al trabajo. [En línea] 2007. [Citado el: 2010 de Enero de 28.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.
6. Mato García, Rosa María. Diseño de base de datos. 1999.
7. Lopez, Angel J**.** ajlopez. [En línea] 2007. [Citado el: 2010 de Enero de 26.] http://www.ajlopez.net/ArticuloMuestra.php?Id=2.
8. UML y patrones Tomo I Prentice Hall Iberoamericana. Larma, Craig. 1999.
9. Konig, Dr Sergio**.** Sistema información hospitalario Hospital P. [En línea] 2008. [Citado el: 26 de Octubre de 2009.] http://neopuertomontt.blogspot.com/2008/07/sistemas-de-informacin.html.
10. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Lenguaje Unificado de Modelado. Manual de referencia. Addison Wesley. Madrid, Pearson Educación. 2000.
11. Hernandis, José Alberto. VersionCero. [En línea] 2004. [Citado el: 2010 de Enero de 28.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
12. Gonzalez Cornejo, Jorge Enrique. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.docirs.cl/uml.htm.
13. CHCA Opera. [En línea] 2004. [Citado el: 2010 de Enero de 26.] http://www.chca.ca/opera.php?lang=es.
14. Florez, Jose**.** Informatica y Salud. [En línea] 2006. [Citado el: 26 de Octubre de 2009.] http://www.informaticaysalud.com/CMS/.
15. Eliurkis**.** Deep in PHP. [En línea] 2007. [Citado el: 2010 de Enero de 26.] http://www.deepinphp.com/2007/06/17/modelo-vista-controlador-mvc/.
16. Cerritos, Dr. Antonio. Manual de introducción de la Informática Médica. [En línea] 2003. [Citado el: 20 de Enero de 2010.] http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf.
17. Care2x. [En línea] 2002. [Citado el: 2010 de Enero de 26.] http://www.care2x.org/.
18. Bolotchi, Ing. Hernan. Siemens. [En línea] 2003. [Citado el: 2010 de Enero de 26.] http://www.siemens.com.ar/sites/internet/legacy/sie-med/soluciones_informaticas_para_la_gestion_hospitalaria.htm.
19. Bermúdez, Javier. Utilizando BPM. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://hpmslab.ing.puc.cl/LinkClick.aspx?fileticket=i9q%2B3gJIn5g%3D&tabid=305&mid=769.
20. Barral, David**.** Rincón del programador. [En línea] 2004. [Citado el: 2010 de Enero de 26.] http://www.elrincondelprogramador.com/default.asp?id=45&pag=articulos%2Fleer.asp.
21. Web taller. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://www.webtaller.com/manual-java/caracteristicas-java.php.
22. W3C. [En línea] 1999. [Citado el: 2010 de Enero de 28.] http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm.
23. Vanguard. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.vanguard.de/dynasite.cfm?dssid=4669.

24. Valen Computer S.A. [En línea] 2003. [Citado el: 2010 de Enero de 26.] http://www.valen.es/cas/hospitales.html 2008.
25. Universidad de Jaén. [En línea] 2007. [Citado el: 2010 de Enero de 28.] http://wwwdi.ujaen.es/asignaturas/isg/Software.html.
26. Sun Microsystems. [En línea] 1994. [Citado el: 2010 de Enero de 26.] http://java.sun.com/javaee/technologies/javaee5.jsp.
27. Sun Developer Network. [En línea] 1994. [Citado el: 2010 de Enero de 28.] http://java.sun.com/javaee/javaserverfaces/.
28. Sivsa. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.sivsa.com/.
29. Seam - Contextual Components. [En línea] 2009. [Citado el: 2010 de Enero de 28.] http://docs.jboss.org/seam/2.0.0.B1/reference/en/html/tutorial.html#d0e1805.
30. PostgreSQL. [En línea] 1996. [Citado el: 2010 de Enero de 28.] http://www.postgresql.org/about/.
31. Portal de salud en Cuba. [En línea] 2000. [Citado el: 2009 de Octubre de 26.] http://www.sld.cu/sistema_de_salud/aspectos.html.
32. Plataforma J2EE. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/.
33. pgAdmin PostgreSQL Tools. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://www.pgadmin.org/.
34. Master magazine. [En línea] 2006. [Citado el: 2010 de Enero de 26.] http://www.mastermagazine.info/termino/7006.php.
35. Masadelante. [En línea] 2006. [Citado el: 2010 de Enero de 26.] http://www.masadelante.com/faqs/servidor.
36. Kioskea. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://es.kioskea.net/contents/cs/csintro.php3.
37. Junta de Andalucía. [En línea] 2009. [Citado el: 2010 de Enero de 28.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces.
38. JBoss. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://www.jboss.com/products/seam.
39. IH-SW-DR-091 ALAS-HIS_Documento de Arquitectura del Sistema.
40. Hista Internacional. [En línea] 2007. [Citado el: 2010 de Enero de 26.] http://www.histaintl.com/servicios/consulting/rup.php.
41. Herramientas Case. [En línea] [Citado el: 2010 de Enero de 28.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.
42. Futurism. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://www.futurismtechnologies.com/Hospital-Information-Management-System-HMIS.htm.
43. Especificación de Requisitos de Bloque Quirúrgico Oftalmológico.
44. Editorial Club Universitario. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.editorial-club-universitario.es/libro.asp?ref=367.
45. Documento de Arquitectura de Software de Bloque Quirúrgico Oftalmológico.

Bibliografía

1. Barral, David**.** Rincón del programador. [En línea] 2004. [Citado el: 2010 de Enero de 26.] http://www.elrincondelprogramador.com/default.asp?id=45&pag=articulos%2Fleer.asp.
2. Bermúdez, Javier. Utilizando BPM. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://hpmslab.ing.puc.cl/LinkClick.aspx?fileticket=i9q%2B3gJIn5g%3D&tabid=305&mid=769.
3. Bolotchi, Ing. Hernan. Siemens. [En línea] 2003. [Citado el: 2010 de Enero de 26.] http://www.siemens.com.ar/sites/internet/legacy/sie-med/soluciones_informaticas_para_la_gestion_hospitalaria.htm.
4. Care2x. [En línea] 2002. [Citado el: 2010 de Enero de 26.] http://www.care2x.org/.
5. Cerritos, Dr. Antonio. Manual de introducción de la Informática Médica. [En línea] 2003. [Citado el: 20 de Enero de 2010.] http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf.
6. CHCA Opera. [En línea] 2004. [Citado el: 2010 de Enero de 26.] http://www.chca.ca/opera.php?lang=es.
7. Documento de Arquitectura de Software de Bloque Quirúrgico Oftalmológico.
8. Editorial Club Universitario. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.editorial-club-universitario.es/libro.asp?ref=367.
9. Eliurkis**.** Deep in PHP. [En línea] 2007. [Citado el: 2010 de Enero de 26.] http://www.deepinphp.com/2007/06/17/modelo-vista-controlador-mvc/.
10. Especificación de Requisitos de Bloque Quirúrgico Oftalmológico.
11. Florez, Jose**.** Informatica y Salud. [En línea] 2006. [Citado el: 26 de Octubre de 2009.] http://www.informaticaysalud.com/CMS/.
12. Futurism. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://www.futurismtechnologies.com/Hospital-Information-Management-System-HMIS.htm.
13. Gonzalez Cornejo, Jorge Enrique. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.docirs.cl/uml.htm.
14. Hernandis, José Alberto. VersionCero. [En línea] 2004. [Citado el: 2010 de Enero de 28.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
15. Herramientas Case. [En línea] [Citado el: 2010 de Enero de 28.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.
16. Hista Internacional. [En línea] 2007. [Citado el: 2010 de Enero de 26.] http://www.histaintl.com/servicios/consulting/rup.php.
17. IH-SW-DR-091 ALAS-HIS_Documento de Arquitectura del Sistema.
18. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Lenguaje Unificado de Modelado. Manual de referencia. Addison Wesley. Madrid, Pearson Educación. 2000.
19. JBoss. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://www.jboss.com/products/seam.
20. Junta de Andalucía. [En línea] 2009. [Citado el: 2010 de Enero de 28.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces.
21. Kioskea. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://es.kioskea.net/contents/cs/csintro.php3.
22. Konig, Dr Sergio**.** Sistema información hospitalario Hospital P. [En línea] 2008. [Citado el: 26 de Octubre de 2009.] http://neopuertomontt.blogspot.com/2008/07/sistemas-de-informacin.html.
23. Lopez, Angel J**.** ajlopez. [En línea] 2007. [Citado el: 2010 de Enero de 26.] http://www.ajlopez.net/ArticuloMuestra.php?Id=2.
24. Masadelante. [En línea] 2006. [Citado el: 2010 de Enero de 26.] http://www.masadelante.com/faqs/servidor.

25. Master magazine. [En línea] 2006. [Citado el: 2010 de Enero de 26.] http://www.mastermagazine.info/termino/7006.php.
26. Mato García, Rosa María. Diseño de base de datos. 1999.
27. Milestone. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.
28. Osalt. [En línea] 2009. [Citado el: 2010 de Enero de 28.] http://www.osalt.com/es/jboss.
29. pgAdmin PostgreSQL Tools. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://www.pgadmin.org/.
30. Plataforma J2EE. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/.
31. Portal de salud en Cuba. [En línea] 2000. [Citado el: 2009 de Octubre de 26.] http://www.sld.cu/sistema_de_salud/aspectos.html.
32. PostgreSQL. [En línea] 1996. [Citado el: 2010 de Enero de 28.] http://www.postgresql.org/about/.
33. Ramos, Juan Alonso**.** Adictos al trabajo. [En línea] 2007. [Citado el: 2010 de Enero de 28.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.
34. Rojo, J. Oscar. Asociación de usuarios de GNU/Linux de castilla y León. [En línea] 2003. [Citado el: 2010 de Enero de 26.] http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos.
35. Sánchez Suárez, José Manuel**.** Adictos al trabajo. [En línea] 2008. [Citado el: 2010 de Enero de 28.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=migrateJSF2Facelets.
36. Seam - Contextual Components. [En línea] 2009. [Citado el: 2010 de Enero de 28.] http://docs.jboss.org/seam/2.0.0.B1/reference/en/html/tutorial.html#d0e1805.
37. Sivsa. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.sivsa.com/.
38. Suárez González, Hécto**.** Java Hispano. [En línea] 2003. [Citado el: 2010 de Enero de 28.] http://www.javahispano.org/contenidos/es/manual_hibernate/.
39. Sun Developer Network. [En línea] 1994. [Citado el: 2010 de Enero de 28.] http://java.sun.com/javaee/javaserverfaces/.
40. Sun Microsystems. [En línea] 1994. [Citado el: 2010 de Enero de 26.] http://java.sun.com/javaee/technologies/javaee5.jsp.
41. UML y patrones Tomo I Prentice Hall Iberoamericana. Larma, Craig. 1999.
42. Universidad de Jaén. [En línea] 2007. [Citado el: 2010 de Enero de 28.] http://wwwdi.ujaen.es/asignaturas/isg/Software.html.
43. Valen Computer S.A. [En línea] 2003. [Citado el: 2010 de Enero de 26.] http://www.valen.es/cas/hospitales.html 2008.
44. Vanguard. [En línea] 2009. [Citado el: 2010 de Enero de 26.] http://www.vanguard.de/dynasite.cfm?dssid=4669.
45. Victoria**.** Definición ABC guia en la red. [En línea] [Citado el: 2010 de Enero de 26.] http://www.definicionabc.com/tecnologia/navegador.php.
46. W3C. [En línea] 1999. [Citado el: 2010 de Enero de 28.] http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm.
47. Web taller. [En línea] 2008. [Citado el: 2010 de Enero de 26.] http://www.webtaller.com/manual-java/caracteristicas-java.php.

Glosario de términos

AJAX: Técnica de desarrollo web para crear aplicaciones interactivas.

API: (Application Programming Interface): Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

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

Bloque Quirúrgico: se define, desde el punto de vista estructural y organizativo, como el espacio en el que se agrupan todos los quirófanos, con los locales de apoyo, instalaciones y equipamiento necesarios para realizar los procedimientos quirúrgicos previstos, por parte del equipo multiprofesional que ofrece asistencia multidisciplinar, que garantiza las condiciones adecuadas de seguridad, calidad y eficiencia, para realizar la actividad quirúrgica.

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

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.

Posoperatorio: Período de cuidados que comienza cuando el paciente termina la cirugía, tiene el propósito de complementar las necesidades psicológicas y físicas directamente después de la cirugía.

Hoja de anestesia: Recogerá los datos e incidencias que se den durante la anestesia, mediante la cumplimentación de las gráficas y epígrafes contenidos en el documento. Corresponde su elaboración al anestesiólogo.

Preoperatorio: el periodo abarca el espacio de tiempo comprendido desde que el paciente es informado de que su problema de salud ha de ser tratado quirúrgicamente, acepta este tratamiento y se fija una fecha para la intervención quirúrgica hasta que el enfermo es trasladado al área quirúrgica.

Quirófano: es una estructura independiente en la cual se practican intervenciones quirúrgicas y actuaciones de anestesia-reanimación necesarias para el buen desarrollo de una intervención y de sus consecuencias que tienen lugar en general en el exterior del quirófano.

Transoperatorio: El tiempo que transcurre desde que el paciente está en la sala de operaciones hasta que son retirados los campos quirúrgicos.

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