Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de diploma para optar por el título

Ingeniero en Ciencias Informáticas

IMPLEMENTACIÓN DEL PROCESO REALIZAR AUTOPSIA DEL MÓDULO ANATOMÍA PATOLÓGICA DEL SISTEMA DE INFORMACIÓN HOSPITALARIA alas HIS

Autores: Amalia Isabel Rivero Chávez

Fidel Pérez Celorio

Tutores: Ing. Nadiezka Milán Cristo

Ing. Yasser M. Garbey Bermúdez

Ciudad de La Habana, Julio de 2010

"Año 52 de la Revolución"

DATOS DE CONTACTO

Tutores:

Ing. Nadiezka Milán Cristo: Graduado en el año 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Especialista del Departamento de Sistema de Gestión Hospitalaria y profesor Instructor vinculado a la Facultad 7. Ha impartido las asignaturas Inteligencia Artificial, Práctica Profesional y Gestión de Software.

Correo electrónico: nmilan@uci.cu

Ing. Yasser M. Garbey Bermúdez: Instructor recién graduado en el año 2009 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Profesor vinculado a la Facultad 7 y miembro del Departamento de Sistema de Gestión Hospitalaria. Ha impartido la asignatura de Introducción a la informática.

Correo electrónico: ymgarbey@uci.cu

RESUMEN

El presente trabajo tiene como objetivo diseñar e implementar las funcionalidades que dan respuestas a los procesos de gestión de la información, obtenidas durante la realización de una autopsia dentro del área Anatomía Patológica de las instituciones hospitalarias. De igual forma, su integración a las funcionalidades del proceso realizar análisis, ya existentes en el módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS.

Para su desarrollo fueron utilizadas las siguientes herramientas: en el diseño del sistema, la metodología RUP, como lenguaje de programación orientado a objetos del lado del servidor, Java; Eclipse como Entorno de Desarrollo Integrado y PostgreSQL 8.3 como Sistema Gestor de Bases de Datos. Además, Hibernate como herramienta ORM para la persistencia de los datos y el framework Seam para la lógica del negocio.

Se prevé que el sistema permita eliminar el procesamiento manual de la información posibilitando con ello estandarizar los protocolos de autopsia y agilizar la obtención del informe final o diagnóstico definitivo. Asimismo posibilitará reducir los costos de inversión que debería hacer el país si adquiere un sistema de este tipo.

INDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

La necesidad humana de mejorar los procesos de la vida cotidiana ha provocado la búsqueda de soluciones a través de las Tecnologías de la Información y las Comunicaciones (TICs), logrando paulatinamente la informatización de la sociedad. Como resultado de esta búsqueda se desarrollaron sistemas de gestión de la información, aplicados a distintos sectores sociales.

En la década de los 70 surgieron los primeros sistemas de información para la salud, estos fueron evolucionando y dieron paso a los Sistemas de Información Hospitalaria (HIS siglas en inglés). Un HIS es un sistema de información orientado a satisfacer las carencias existentes de generación de información, para almacenar, procesar y reinterpretar datos médico-administrativos de cualquier institución hospitalaria. Minimizan los inconvenientes burocráticos que enfrentan los pacientes lo que permite hacer más eficiente la administración de los recursos humanos y materiales.

Al analizar la composición y estructura de los sistemas actuales, se han determinado características comunes entre ellos; gestionan información de diferentes áreas de un hospital por lo que disponen de enormes volúmenes de datos, los flujos de procesos son muy complejos y generan reportes e informes dependiendo del área o servicio para el cual se requiera.

Una de las áreas que compone un hospital es Anatomía Patológica. Esta se encarga del procesamiento histológico de las muestras extraídas en biopsias y citologías y del examen anatómico de un cadáver, comúnmente conocido como autopsia.

El proceso para realizar una autopsia describe el flujo de información desde que el cadáver ingresa a la institución y se registran sus datos hasta que se diagnostica la causa de muerte a través de estudios macroscópicos y microscópicos, realizados a las muestras que se extraen del cuerpo. El cadáver puede ser retirado por los familiares o por alguna institución, registrándose los datos de las personas involucradas en esta actividad.

Durante la práctica de una autopsia, en ocasiones, la documentación se emite de forma manual, esto trae consigo falta de legibilidad y deterioro de los datos, lo que provoca realizar nuevamente el procedimiento y afecta la generación de informes estadísticos con fines administrativos. Además, representa el empleo de gran parte del tiempo de la jornada laboral, que sin dudas pudiera ser destinado a otras tareas.

La falta de homogeneidad y estandarización de las solicitudes y los protocolos macroscópicos y microscópicos de autopsia, que se generan durante este proceso, golpea la codificación de los datos dentro del área. Conjuntamente puede existir retraso a la hora de realizar los diagnósticos pertinentes por la carencia de información para efectuar los estudios. Otra dificultad es la lentitud al documentar el proceso debido al volumen de datos repetidos y muestras que se almacenan.

Analizando la situación anterior se define como problema a resolver: ¿Cómo facilitar la gestión de información durante la realización de una autopsia en el área de Anatomía Patológica de las instituciones hospitalarias?

El objeto de estudio es el proceso de gestión de la información en el área de Anatomía Patológica de las instituciones hospitalarias. El campo de acción está enmarcado en el proceso de gestión de la información durante el desarrollo de una autopsia en el área de Anatomía Patológica de las instituciones hospitalarias.

El objetivo de la investigación es implementar el proceso realizar autopsia del módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS, que facilite la gestión de información en esta área de las instituciones hospitalarias.

Para dar cumplimiento al objetivo planteado se definen las siguientes tareas a desarrollar:

- Evaluar las tendencias actuales de los Sistemas de Información Hospitalaria en el ámbito internacional.
- Asimilar la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
- Asimilar las pautas para el desarrollo de los entregables y el diseño, definidas en el departamento Sistemas de Gestión Hospitalaria.
- Analizar los patrones de diseño a utilizar.
- Obtener los artefactos correspondientes a los flujos de trabajo “Diseño”, “Implementación” y “Pruebas”.
- Implementar las funcionalidades del proceso realizar autopsia, definido en el análisis del módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS.

- Iniciar autopsia
- Estudiar muestra macroscópica de autopsia
- Estudiar muestra para diagnóstico definitivo de autopsia
- Darle entrada y salida a un cadáver

– Obtener el acta de liberación otorgada por calidad interna, de la documentación y de las funcionalidades implementadas del proceso realizar autopsia.

Se prevé que las nuevas funcionalidades, vinculadas al proceso realizar autopsia del módulo de Anatomía Patológica del Sistema de Información Hospitalaria alas HIS proporcionen beneficios como:

– Evitar el creciente deterioro de la documentación en formato duro debido al factor tiempo.

– Eliminar el procesamiento manual de la información contribuyendo a la informatización de todo el proceso.

– Facilitar a los patólogos una herramienta rápida y estandarizada, para la realización de solicitudes, descripciones macroscópicas y microscópicas de autopsia.

– Obtener los datos necesarios para la generación de informes estadísticos.

– Agilizar el procesamiento de las muestras obtenidas en la autopsia.

– Incrementar la seguridad de acceso a los protocolos generados.

El presente documento está estructurado en cuatro capítulos que se relacionan a continuación:

Capítulo 1 Fundamentación teórica: se detallarán cada uno de los aspectos relacionados con el sistema a implementar, así como el análisis de soluciones existentes relacionadas con el mismo a nivel nacional e internacional. Explica además un estudio detallado de las tecnologías actuales y las herramientas utilizar para el diseño e implementación del sistema.

Capítulo 2 Descripción de la arquitectura: describe la arquitectura, definiéndose además los requerimientos no funcionales, la estrategia de integración a otros servicios, la seguridad a implementar en el sistema así como la estrategia de codificación.

Capítulo 3 Descripción y análisis de la solución propuesta: se centra en el modelado detallado y la construcción de la aplicación.

Capítulo 4 Modelo de prueba: refleja el método de prueba a utilizar y se describen los casos de uso correspondientes.

Capítulo 1. Fundamentación teórica

El presente capítulo muestra los conceptos básicos relacionados con la gestión de la información en el área de Anatomía Patológica. Además son analizados los sistemas o subsistemas similares, existentes a nivel nacional e internacional así como las tecnologías, herramientas y metodologías que se utilizan en el desarrollo del software justificando el uso de cada una de ellas.

1.1 Conceptos básicos relacionados con el dominio del problema

La Anatomía Patológica constituye un pilar importante e indispensable, sin el cual, el resto de las especialidades médicas difícilmente podrían funcionar de manera óptima. Se interrelaciona con todas y cada una de ellas, ocupándose del estudio por medio de técnicas morfológicas, de las causas, desarrollo y consecuencias de las enfermedades. Su fin lo constituye el diagnóstico correcto de biopsias, piezas quirúrgicas, citologías y autopsias. [1]

La práctica de las autopsias forma parte de la labor diagnóstica de los patólogos. La misma consiste en el examen de un cuerpo después de la muerte con el objetivo de obtener o verificar un diagnóstico.[2]

Las autopsias se realizan por varias razones:

- Cuando ocurre una muerte sospechosa.
- Existencia de preocupación de salud pública, como por ejemplo una enfermedad misteriosa.
- Si alguien muere sin atención médica, o si el médico que atendió a la persona tiene alguna inconformidad al firmar el acta de defunción.
- La familia del fallecido puede pedirle al hospital que se realice una autopsia.

Los procedimientos de la autopsia comienzan con pasos generales y concluyen con pasos específicos:

1. Primero, se hace un examen visual de todo el cuerpo, así como de los órganos y estructuras internas.
2. Luego, pueden realizarse exámenes microscópicos, químicos y microbiológicos en los órganos y tejidos.

[1] IH-SW-DE-020 ALAS-HIS_Anatomía Patológica_Glosario de términos
[2] La autopsia (Autopsy): http://www.healthsystem.virginia.edu/uvahealth/adult_path_sp/autopsy.cfm

3. Todos los órganos extirpados que van a examinarse se pesan y se guarda una sección para procesarla en platinas de microscopio.
4. El informe final se hace después de que todos los resultados de laboratorio estén completos, estos son denominados protocolos de autopsia.

Los informes son el producto final derivado como resultado del estudio de las muestras por cualquiera de los procedimientos antes mencionados que contiene la opinión médica experta basada en observaciones personales integradas con los datos clínicos y científicos del momento cuyo peso principal descansa en el conocimiento y experiencia del patólogo.

1.2 Sistemas automatizados existentes vinculados con el campo de acción

Un sistema de gestión de información es una de las herramientas más eficaces para proporcionar más y mejor asistencia en el área de Anatomía Patológica. En muchos países existen empresas dedicadas y patentadas para la producción de estos tipos de sistemas, a continuación se muestran características fundamentales de algunos productos.

SARCAP

En 1985 fue creado en Cuba por el Grupo Nacional de Anatomía Patológica, un Sistema Automatizado de Registro y Control de Anatomía Patológica, conocido como SARCAP. El mismo está compuesto por dos subsistemas que actúan de forma independiente sobre las bases de datos de autopsias y biopsias respectivamente. Ambos subsistemas presentan una estructura modular y utilizan un fichero común denominado diccionario, en el cual están almacenados los códigos de las enfermedades con su correspondiente descriptor.

La interacción del usuario con el sistema se realiza a través de un menú que posee las mismas opciones para los subsistemas. Este sistema facilita el conocimiento de los diagnósticos realizados tanto clínicos como anatomopatológicos y en especial, las causas de muerte que pueden ser directas, intermedias, básicas y contribuyente, así como las coincidencias o discrepancias diagnósticas. A la vez, permite la evaluación de la calidad de la labor médica realizada, lo que beneficia el trabajo de los comités de auditoría médica.

Toda la información puede obtenerse sobre un caso individual o sobre todos los estudiados, en determinado plazo de tiempo, que puede ser solicitado a voluntad e incluir todas las autopsias o biopsias

acumuladas en la base de datos. Las bases de datos que brinda el SARCAP posibilitan la realización de trabajos científicos con el aporte de sólida argumentación a diversas líneas de investigación siendo algunas de ellas, infecciones, daño multiorgánico, edema pulmonar de permeabilidad, muerte por hechos violentos y cáncer.

Principales ventajas del SARCAP

Administrativas

- Permite conocer el trabajo de las biopsias y autopsias realizadas en el hospital en cualquier plazo de tiempo, incluida toda la información que esos procederes suministran; el control del personal que ha trabajado en dichas investigaciones; los diferentes plazos de tiempo empleados en su realización; los principales recursos materiales utilizados, así como realizar la consolidación y comparación de estos resultados entre diferentes hospitales.
- Ofrece ventajas económicas que se derivan del ahorro de tiempo y personal que se necesitarían para el procesamiento manual de la información elaborada por el SARCAP.

Asistenciales

- Facilita la confección de los informes.
- Favorece la elevación de la calidad en el diagnóstico.

Docente

- Permite obtener datos para la enseñanza y sirve de base para el aprendizaje con el auxilio de computadoras.

Investigativas

- Brinda los resultados en forma de tablas, por lo que facilita la elaboración de trabajos científicos.
- Las experiencias que se derivan del procesamiento de los datos acumulados pueden generalizarse en la práctica médica.

El SARCAP facilita la utilización de los conocimientos que aportan los métodos de la Anatomía Patológica en el control, evaluación y aseguramiento de la calidad del trabajo médico. Con este sistema se ha creado

el Banco de Datos de Autopsias Nacional pero la clasificación de varias especialidades es insuficiente para el trabajo de algunos laboratorios. [3]

AvantPat 1.1

En el año 2004 se dio a conocer el sistema AvanPat 1.1, desarrollado en el Hospital Clínico Quirúrgico Joaquín Albarrán, de La Habana. Es un sistema que almacena, organiza y gestiona la información relacionada con los estudios que se realizan en los departamentos de Anatomía Patológica.

Este genera de forma automática los informes administrativos que deben entregar los departamentos de Anatomía Patológica acerca del trabajo realizado en un período de tiempo seleccionado. Posee un módulo para estadísticas con fines investigativos. Incorpora un registro de pacientes que permite almacenar la información de forma personalizada para su análisis evolutivo.

Se ejecuta sobre Windows 9x, NT, 2000, XP, 2003 y posee una interfaz fácil de utilizar tanto para la entrada de información como para la recuperación y utilización de la misma, está diseñado para trabajar en red. Consta de dos aplicaciones: un cliente y un servidor, ambos están programados en el Borland Delphi 7.0 y para el gestor de la base de datos se utiliza el MySQL 4.0.

Engloba cinco estudios básicos prediseñados como son: citología vaginal, citología de líquidos, punción aspirativa con aguja fina, biopsia y necropsia, los cuales cuentan con dinamismo entre sus campos de forma tal que el usuario tiene la posibilidad de adicionar nuevos campos a los mismos. Cada estudio posee cuatro secciones: datos generales, laboratorio, galería de imágenes y una sección que se recogen los resultados anatomopatológicos: descripción macroscópica, microscópica, resultado anatomopatológico final, entre otros.

No requiere grandes recursos técnicos para su utilización, los requerimientos mínimos necesarios son:

- PC 486.
- 16 Mb de memoria RAM.
- 25 Mb de espacio libre en el disco duro.
- Pantalla VGA (Resolución mínima de pantalla de 800x600 píxel).

[3] Registro nacional de autopsias en Cuba Utilización del SARCAP http://www.patologia.es/volumen37/vol37-num1/37-1n04.htm

- Impresora compatible con Windows.

x-HIS

El sistema de información extensible x-HIS, constituye una solución global, resultado del esfuerzo de iSOFT para ofrecer la última tecnología en gestión clínica y administrativa, en un entorno abierto que permite la integración con otros sistemas de información. Constituye además una herramienta puramente clínica, adaptada a las necesidades de los profesionales sanitarios, imitando sus formularios y documentos habituales, con el fin de que su aprendizaje y uso cotidiano sea fácil, cómodo, y rápido. x-HIS sitúa al paciente en el centro del sistema. Tras la identificación del paciente, el usuario del sistema podrá navegar por todo su historial clínico, a través de un navegador asistencial, sin necesidad de salir y entrar en diferentes aplicaciones.

El software integra la información de un hospital o conjunto de hospitales con las áreas de gestión, hospitales de referencia, centros de especialidades, laboratorios, sistemas RIS y PACs, contabilidades y otros organismos o corporaciones que se desee, de forma que proporcionen una historia clínica única y completa del paciente. También puede gestionar la información de varios departamentos entre los que se encuentran, laboratorios clínicos, unidad de cuidados intensivos y uno de gran importancia para la labor de diagnóstico que se realiza dentro de los hospitales: Anatomía Patológica. El sistema permite registrar las muestras que llegan al departamento, especificar las diferentes técnicas, descripciones macroscópicas y microscópicas, observaciones realizadas, así como asignarles resultados diagnósticos, dando salidas, impresiones de informes y control de costos, entre otras acciones.

Características

Entre otras características, se destacan:

- Uso de Java como herramienta de desarrollo.
- Simplicidad y usabilidad.
- Uso de estándares, por ejemplo XML.
- Integración asistencial para conocer en todo momento todos los encuentros del paciente donde quiera que se hayan producido, atención primaria, hospitalaria y/o sociosanitaria.
- Integración del flujo de trabajo del personal médico y de enfermería.

- Modularidad, escalabilidad y personalización al 100%, gracias a su estructura modular basada en objetos.
- Seguridad en el sentido más amplio de la palabra.
- Uso de perfiles de trabajo como son tratamientos, cuidados de enfermería, etcétera.

Los resultados funcionales obtenidos hasta el momento demuestran que, teniendo como referencia al paciente y a la actividad clínica que puede ser realizada por los distintos profesionales, es posible la construcción de un entorno de trabajo simple, pero donde es posible aglutinar los aspectos más importantes para el desarrollo del trabajo diario de una forma completamente configurable a demanda del usuario.[4]

Kewan HIS

Este sistema es una solución integral y completa en un único producto para la gestión de centros sanitarios: hospitales públicos y privados, clínicas y policlínicas. Incluye la gestión completa de pacientes y sistemas departamentales. Forma parte de un Sistema de Gestión de Recursos Empresariales (ERP siglas en inglés) que provee, en un mismo producto, una solución para la gestión de recursos humanos, logística y financiera.

Además de servir de apoyo a la gestión hospitalaria, posibilita:

- Procesos administrativos de admisión de pacientes, generación de documentos e impresión de etiquetas.
- Gestión de áreas básicas: admisión, archivo, facturación así como otras áreas.
- Gestión de sistemas departamentales como farmacia y rehabilitación.
- Además cubre la funcionalidad de atención primaria y permite la gestión de la información clínica.

Entre sus funcionalidades se encuentran, la gestión de pacientes y procesos clínicos, la admisión de pacientes, gestión de camas, unidad de observación, cambio de servicio así como libro de registros mediante el módulo de urgencias. Incluye la gestión de solicitudes, define agendas por recurso, por cantidad o por tiempo. La gestión de citas puede realizarse de forma manual, automática o periódica. Mediante la funcionalidad de quirófanos pueden gestionarse las listas de espera, definición de sesiones

[4] xHIS: Un nuevo concepto en Sistemas de Información Hospitalarios
http://www.conganat.org/SEIS/inforsalud03/INFORSALUD2003_garciaa1.pdf

quirúrgicas: planificador gráfico, programación de intervenciones, captura de actividad, como de personal así como consumo de material.

En general Kewan HIS puede gestionar la información de varios departamentos como son, la Central de esterilización, Radiodiagnóstico y Anatomía Patológica, este último con el objetivo de llevar el control de las solicitudes y resultados de diagnósticos de las muestras llevadas al laboratorio, todo ello con fines estadísticos e investigativos.[5]

PAT-win

PAT-win creado por la empresa Novasoft, es para el departamento de Anatomía Patológica, una de las soluciones que permite realizar una gestión integrada de los departamentos anatomopatológicos. Puede ser utilizada dentro del sistema de información de un hospital o de forma individual en servicios o laboratorio de esta área de la salud. La gestión permite principalmente registrar las muestras que llegan al departamento, especificar las diferentes técnicas, realizar descripciones macroscópicas y microscópicas, incluir observaciones realizadas, así como asignar los resultados diagnósticos, gestionando las salidas, las impresiones de informes y el control de costos. PAT-win, con más de diez años en el mercado, es el sistema de información líder y con más cobertura en el mercado español.

Características:

- Integración con el sistema de información hospitalaria del centro que permite mantener un historial único de pacientes.
- Multicentro: permite integrar información de varios servicios de Anatomía Patológica (base de datos única).
- Multinforme: permite trabajar con estudios independientes de un mismo paciente y poder recuperar toda la información en un único informe.
- Compatibilidad con los sistemas operativos Windows 95, 98, 2000, XP.
- Uso de las bases de datos más extendidas: Informix, Oracle, MS SQL Server, Sybase.
- Interfaz gráfica de fácil manejo y muy intuitivo, con posibilidad de utilizar atajos de teclado.

[5] La experiencia Kewan – Cosmosalud. http://www.informatica2007.sld.cu/Members/jherreroypf/la-experiencia-kewan-cosmosalud

- Posibilidad de gestión y confección de todo tipo de estudios (biopsias, citologías generales, ginecológicas, autopsias, inmunohistoquímica, casos consulta), fácil acceso a otros estudios de pacientes.
- Módulo de gestión de imágenes (opcional).
- Corrector ortográfico integrado.
- Gran número de listados y estadísticas.

Funcionalidades

- Incluyen módulo de laboratorio. Gestiona el trabajo pendiente y realizado por cada técnico de laboratorio.
- Gestor/Editor de informes integrado, disminuyendo los tiempos de acceso a los informes al evitar el uso aplicaciones externas.
- Permite el uso de sistemas de reconocimiento de voz para confección de descripciones macroscópicas, microscópicas y diagnósticos.
- Almacén integrado: posibilidad de integración con el almacén hospitalario.
- Gestión de modificaciones en informes emitidos. Notas adicionales con posibilidad de configuración: nota imprimible, no imprimible y tipos de notas, quedando constancia del patólogo que ha realizado la modificación y cuándo la ha llevado a cabo.
- Posibilidad de solicitar peticiones y consultar informes vía web.[6]

1.3 Tecnologías actuales a considerar

Teniendo en cuenta las características del entorno donde se desplegará el producto y el estado del arte de los HIS en el mundo, se define que las tecnologías a utilizar en el proceso de desarrollo del sistema deben ser libres, multiplataforma y que optimicen el mismo.

Lenguaje de programación: Java

Java es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

[6] Pat-Win http://www.ittrade.com.mx/pdf/PATwin_080108.pdf

La plataforma Java consta de las siguientes partes:

- El lenguaje de programación, Java.
- La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.
- El API Java, una biblioteca estándar para el lenguaje.
- El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk que a éste.

Java es un lenguaje orientado a objeto que toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros. Tiene una arquitectura neutral, portátil y robusta. Posee las estructuras mínimas de un lenguaje de programación tradicional, sin añadir ninguna estructura más.

Es un lenguaje distribuido proporcionando una colección de clases para su uso en aplicaciones de red, que le permitan establecer conexiones con servidores o clientes remotos. Es compilado e interpretado a la vez: compilado al generar un código intermedio (bytecodes) que es interpretado por la maquina virtual. Proporciona numerosas comprobaciones en compilación y tiempo de ejecución y elimina la necesidad de liberación explicita de memoria, convirtiéndolo en un lenguaje robusto.

Posee implementadas barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real. Está diseñado para soportar aplicaciones que serán ejecutadas en varios entornos de red: Unix, Windows, Mac. El bytecodes generado es indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas. Soporta sincronización de múltiples hilos de ejecución (multithreading).

Plataforma: Java EE

Java Enterprise Edition o Java EE, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura distribuida de N niveles, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process.

Seam

JBoss Seam es una nueva y potente infraestructura para desarrollar aplicaciones Web 2.0 de próxima generación, al unificar e integrar tecnologías como AJAX, JavaServer Faces (JSF), Enterprise Java Beans (EJB3), Java Portlets, Business Process Management (BPM), Drools, Hibernate y JPA en una única solución.

Elimina la complejidad a nivel desde arquitectura hasta API, permitiendo la creación de complejas aplicaciones web basadas en Plain Old Java Objects (POJO), 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 y ICEFaces.

En la administración de estado, Seam provee una mayor granularidad de contextos. El 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 evitando memory-leaks. Integra además el concepto de workspaces permitiendo que el usuario tenga en varios tabs o ventanas del navegador actividades del negocio con contextos completamente aislados. Además integra transparentemente la administración de procesos del negocio vía jBoss jBPM, haciendo muy fácil implementar y optimizar complejas colaboraciones (workflows) y complejas interacciones con el usuario (pageflows). También a través de ficheros de reglas (Drools) define las posibles bifurcaciones del negocio permitiendo el fácil manejo de las condiciones sin tener que modificar el código fuente.[7]

JBoss AS como servidor de aplicaciones

JBoss Application Server es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado. Está licenciado bajo la LGPL, por lo que puede libremente usarse sin costo alguno en cualquier aplicación comercial o ser redistribuido. Por ser una plataforma certificada JEE 5, soporta todas las especificaciones correspondientes, 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) 3.0.8

[7] Allen, Dan. Seam in Action. s.l.: Manning Publication, 2008.
[8] Javid Jamae, Peter Johnson. JBoss in Action. s.l.: Manning Publications, enero 2009. 1933988029.

Los componentes claves son: JBoss AS 4.2, Hibernate 3.2.4, Seam 2.0.

JBoss Tools

JBoss Tools es un conjunto de plug-ins de Eclipse que tiene como objetivo ayudar a los desarrolladores a crear aplicaciones webs de forma rápida y sencilla.

Los módulos de JBoss Tools son:

- **RichFaces VE:** El editor visual aportado por Exadel proporciona el apoyo para la edición visual de páginas HTML, JSF, JSP y Facelets. También incluye soporte visual para las librerías de componentes JSF incluyendo JBoss RichFaces.
- **Seam Tools:** Incluye soporte para seam-gen, RichFaces VE.
- **Hibernate Tools:** Soporta el mapeo de archivos, anotaciones y JPA con la ingeniería inversa, completamiento de código, asistentes de proyecto, refactorización, ejecución interactiva de HQL/JPA-QL/Criteria.
- **JBoss AS Tools:** Fácil de iniciar, detener y debuguear al estar integrado con Eclipse. También incluye funciones para el despliegue eficaz de cualquier tipo de proyecto en el IDE.
- **JBossWS Tools:** Desarrollo, invocación, inspección y pruebas de webservices sobre http con la adición y soporte de características JBossWS.

JSF

JavaServer Faces (JSF) es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa Facelets como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL. [9]

Rich Faces 3.2 como librería de componentes JSF

RichFaces es una rica biblioteca de componentes para JSF que posee un avanzado marco para integrar fácilmente capacidades AJAX en el desarrollo de aplicaciones de negocios. Permite a los desarrolladores ahorrar tiempo y aprovechar las características de los componentes para crear aplicaciones Web, ricas en

[9] UI Develoment with JaveServer Faces.

interfaz. Proporciona componentes fáciles de utilizar con etiquetas predefinidas, y brinda capacidades AJAX (Ajax4jsf).[10]

Ajax4jsf

Ajax4jsf es una librería open source que se integra totalmente en la implementación de JSF usada, 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 recargar por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. Estas características de la librería mencionada dotan a la aplicación JSF de contenido mucho más profesional con muy poco esfuerzo.[11]

PosgtreSQL 8.3 como servidor de base de datos

PostgreSQL es un sistema de base de datos relacional que destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL. Pertenece al ámbito del software libre, está bajo la licencia BSD (Berkeley Software Distribution). Esta licencia tiene menos restricciones en comparación con otras como la GPL(de la Fundación del Software Libre). Bajo esta licencia, el autor mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación. A su vez, el usuario tiene libertad ilimitada con respecto al software, puede decidir incluso redistribuirlo como no libre. Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac OS X, Solaris, BSD, Tru64 y otros más.

Soporta ACID, o lo que es lo mismo, la realización de transacciones seguras; también, vistas, uniones, claves extranjeras, procedimientos almacenados, triggers, implementa internamente lenguajes de consulta de muy alto nivel como son el plpgsql (muy similar al plsql de Oracle), el plperl, el plpiton y el c; además adicionar, que es un gestor preparado para incluirle lenguajes de consultas adicionales, habiendo la industria desarrollado alrededor de 20 lenguajes que este gestor soporta. Incluye la mayor parte de los tipos de datos especificados en los estándares SQL92 y SQL99, como: entero, numérico, booleano, char, varchar, fecha, interval o timestamp.

[10] Red Hat. RichFaces developer Guide. 2007.
[11] Ajax4jsf Developer Guide. 2007.

Para el control de réplicas en la actualidad existen variadas soluciones como es el Slony I (que permite réplicas asíncronas entre servidores de tipo Master-Slave) y el PgCluster (que permite la creación de sistema de réplicas síncronos entre servidores de tipo Master-Master).

El tamaño máximo de la base de datos es ilimitado; el de una tabla asciende a 32 TB, el de una fila a 1.6 TB y el de un campo de datos a 1 GB; el número de filas en una tabla es ilimitado, pero no el de columnas, que oscila entre 250 y 1600 columnas por tabla. El número de índices por tabla es también ilimitado.

Entre las ventajas que proporciona la versión 8.3 se encuentran:

- Soporte SQL/XML de acuerdo al estándar ANSI, incluyendo exportación en formato XML.
- Búsqueda en texto: ha sido incorporada la herramienta, TSearch2, en la distribución central, con mejor manejo y nuevos diccionarios e idiomas.
- Soporte de autentificación GSSAPI y SSPI.
- Nuevos tipos de datos: UUIDs, ENUMs y arreglos de tipos compuestos.

La versión 8.3 del gestor de base de datos ofrece mayor rendimiento que las versiones anteriores (entre un 5% y un 30% más dependiendo de la carga de trabajo).[12]

Framework Hibernate para acceso a los datos

Hibernate es una herramienta de Mapeo objeto-relacional(ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.[13]

JPA (Java Persistence API)

Java Persistence API, más conocida por su sigla JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es

[12] The postgreSQL Global Develoment Group. PostgreSQL 8.3.6 Documentation. 2008.
[13] Cristian Bauer, Gavin King. Java Persistence with Hibernate. 2005. 1-932394-88-5.

no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como POJOs).

RUP como metodología de desarrollo

Rational Unified Process (RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Utiliza UML para definir los modelos de software y se divide en 4 fases: Inicio, Elaboración, Construcción y Transición.

En esta metodología se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería, de los cuales se desarrollaran las actividades correspondientes a los flujos de Diseño y Pruebas. Los tres últimos constituyen flujos de apoyo. RUP tiene tres características esenciales: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental.[14]

UML

Lenguaje Unificado de Modelado (UML siglas en inglés) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

UML ofrece un estándar para describir un "plano" del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Este puede ser utilizado en sistemas desarrollados en varios lenguajes de implementación y plataformas, incluyendo lenguajes de programación, bases de datos. Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos y está especialmente pensado para apoyar un estilo de desarrollo iterativo e incremental.

[14] Pressman, Roger S. Ingeniería de software, un enfoque práctico . 2005.

Herramientas de desarrollo

Eclipse como herramienta de desarrollo

Eclipse es un Entorno de Desarrollo Integrado (IDE siglas en inglés), multiplataforma, abierto y extensible. Eclipse es un IDE Java, aunque da soporte a otros lenguajes de programación, como son: C/C++, Cobol, Fortran, PHP o Pyton. Este entorno de desarrollo fue creado inicialmente por la IBM y actualmente es desarrollado por la Fundación Eclipse. Se le puede añadir a este IDE soporte de lenguajes adicionales mediante pluggins.

Ha sido diseñado de forma que pueda ejecutarse en cualquier plataforma. La última versión estable se encuentra disponible para los sistemas operativos Windows, Linux, Solaris, AIX, HP-UX y Mac OSX. Emplea un diseño basado en módulos (plug-in) los cuales se le pueden añadir para extender sus funcionalidades.

Todas las versiones de Eclipse necesitan tener instalado en el sistema una máquina virtual Java (JVM), preferiblemente JRE (Java Runtime Environment) o JDK (Java Developer Kit) de Sun. Se distribuye bajo licencia EPL (Eclipse Public License). Esta licencia es considerada como libre por la FSF y por la OSI. La licencia EPL permite usar, modificar, copiar y distribuir nuevas versiones del producto licenciado. El antecesor de EPL es CPL (Common Public Licencse) escrita por IBM.

PgAdmin como aplicación cliente para manejar la Base de datos

PgAdmin es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede

hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad.

Visual Paradigm como herramienta de modelado

Visual Paradigm es una poderosa herramienta para visualizar y diseñar elementos de software, para ello utiliza UML (UML 2.1) y ofrece una gama de facilidades para el modelado de aplicaciones. Está orientada a la creación de diseños usando el paradigma de programación orientada a objetos.

Provee soporte para la generación de código, tiene integración con diversos IDE's como NetBeans (de Sun Microsystems), JDeveloper (de Oracle), Eclipse (de IBM), JBuilder (de Borland), así como la posibilidad de realizarse la ingeniería inversa para aplicaciones realizadas en JAVA, .NET, XML e Hibernate.

Tiene dentro de sus características que es portable y posee gran facilidad de uso. Su diseño se centra en casos de uso y se enfoca al negocio que genera un software de mayor calidad. También tiene disponibilidad en múltiples plataformas. Soporta BPMN (Business Process Modeling Notation), modelado colaborativo con CVS y Subversion.

JRE

JRE es el acrónimo de Java Runtime Environment (entorno en tiempo de ejecución Java) y se corresponde con un conjunto de utilidades que permite la ejecución de programas java sobre todas las plataformas soportadas.

JVM (máquina virtual Java) es una instancia de JRE en tiempo de ejecución, este es el programa que interpreta el código Java y además por las librerías de clases estándar que implementan el API de Java. Ambas JVM y API deben ser consistentes entre sí, de ahí que sean distribuidas de modo conjunto.

Un usuario sólo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JDK, que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java.

Los sistemas analizados en este capítulo, provocan que se restrinja el uso de los mismos en las instituciones hospitalarias. Debido a que son software propietario no es posible realizarle modificaciones, el código fuente no está publicado, además son más costosos. Por otra parte, se limita el uso para el usuario impidiendo su redistribución. Generalmente son aplicaciones de escritorio, las cuales aumentan los gastos por la necesidad de equipos de mayor potencia en relación al uso de memoria, espacio en disco y CPU para su previa instalación.

Se hace necesario desarrollar un software flexible, que no sea muy costoso ya que deberá ser desplegado en una amplia red de instituciones. Con el objetivo de obtener una aplicación de costo mínimo, se definieron las siguientes herramientas que conforman el ambiente de desarrollo del módulo de Anatomía Patológica:

- Entorno de desarrollo: Eclipse.
- Gestor de base datos: PostgreSQL 8.3.
- Herramienta de Modelado: Visual Paradigm.
- JRE.

Capítulo 2. Descripción de la arquitectura

El presente capítulo muestra los requisitos no funcionales, la arquitectura definida por el departamento de Sistema de Gestión Hospitalaria para el desarrollo de sus aplicaciones, así como la seguridad, la vista de despliegue y las estrategias de codificación.

2.1 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. A través de estos se especifican propiedades del sistema como restricciones de ambiente y desarrollo, rendimiento, dependencias de plataformas y mantenibilidad. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. [15]

2.1.1. 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.

La estructura concebida para la organización de la información agiliza el entendimiento del sistema por parte del usuario.

Los usuarios serán capaces de alcanzar sus objetivos con un mínimo esfuerzo y obteniendo los resultados máximos.

El sistema será capaz de solucionar un error cometido por el usuario o sugerir las posibles soluciones, indicándole al usuario las acciones pertinentes a seguir.

Brindará comodidad a la hora de acceder a las diferentes funcionalidades que proporciona la aplicación mediante teclas de acceso rápido. Serán reutilizados diseños de aplicaciones comúnmente usada por los usuarios finales con vistas a aprovechar la experiencia de usuario.

[15] Pressman, Roger S. Ingeniería de software, un enfoque práctico . 2005.

2.1.2. Fiabilidad

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

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

Las informaciones médicas relacionadas con los pacientes y que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas.

Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.

Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando sólo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión.

Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.

Se establecerán mecanismos de control y verificación para los procesos susceptibles de fraude. Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.

El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (log) de cada usuario en el sistema.

El sistema soportará el uso de firmas digitales para la transferencia de información cuya certificación sea imprescindible para validar el uso de la misma.

El sistema implementará un control de cambios a determinados campos de información (seleccionados por su importancia), de forma tal que sea posible determinar cuáles han sido las actualizaciones que se le han realizado.

Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista.

El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

2.1.3. Eficiencia

El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos.

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

2.1.4. Soporte

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP. Además la administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación. Realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo, además de recuperar la base de datos a partir de los respaldos realizados. Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema. Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

2.1.5. Seguridad de acceso y administración de usuarios

Se mantendrá seguridad y control a nivel de usuario, garantizando su acceso sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.

Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando únicamente la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión. 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.

2.1.6. Monitoreo de funcionamiento

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

2.1.7. 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.

2.1.8. 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.

2.1.9. Configuración de parámetros

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

2.1.10. Réplica

Se permitirá realizar réplica de la base de datos de los hospitales con el Centro de Datos. Esta réplica se podrá hacer de forma manual y automatizada a través de la red.

2.1.11. 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.

2.1.12. 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.

2.1.13. 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 Description 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.

2.1.14. 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.

2.1.15. 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.

2.1.16. 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.[16]

[16] IH-SW-DR-090 ALAS-HIS_Requerimientos suplementarios

2.2 Descripción de la arquitectura

Una de las tareas más importantes en la construcción de cualquier sistema de gestión es la definición de su ambiente de desarrollo. La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos para la creación de un producto de software. Selecciona y diseña con base a objetivos prefijados para el sistema de información, que pueden ser de tipo funcional, de mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información, teniendo en cuenta las limitaciones derivadas de las tecnologías disponibles para implementarlos.[17]

La arquitectura definida se basa en el patrón arquitectónico Modelo-Vista-Controlador. Este patrón separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información.

Controlador: Controla el flujo de datos entre la vista y el modelo.

Entre las ventajas del estilo Modelo-Vista-Controlador están las siguientes:

- **Soporte de múltiples vistas:** Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes.
- **Adaptación al cambio:** Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no genera afectaciones.

[17] IH-SW-DR-091 ALAS-HIS_Documento de Arquitectura del Sistema

Una desventaja es el costo de actualizaciones frecuentes, si se experimentan cambios frecuentes, por ejemplo, podría desbordar las vistas con una gran cantidad de requerimientos de actualización.[18]

La elección de este patrón está basada en el hecho de que convierte a una aplicación en un paquete mantenible, modular y de desarrollo rápido permitiendo que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros.

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

El área de Anatomía Patológica es un servicio de apoyo que aporta resultados confiables y oportunos a las instituciones hospitalarias. Para su funcionamiento se nutre de la información extraída de las solicitudes de biopsias y citologías que se generan internamente en el hospital por los servicios Consulta externa, Hospitalización y Bloque quirúrgico, y las solicitudes de autopsia generadas por Emergencia y Hospitalización. De igual forma se reciben solicitudes de instituciones externas, siendo tarea de Anatomía Patológica, el registro de los datos correspondientes y el envío de los diagnósticos finales.

El módulo fue diseñado para satisfacer las necesidades del área, al contrario de otros, puede trabajar de forma independiente. Cuenta con dos procesos generales: realizar análisis y realizar autopsia. Actualmente se encuentran implementadas las funcionalidades que responden al primero, siendo una de ellas el subproceso que permite gestionar el flujo de información durante el procesamiento de muestras en los laboratorios. Este será reutilizado al extraer las muestras del cadáver para emitir el diagnóstico final durante el proceso realizar autopsia.

Los procesos antes mencionados contienen funcionalidades que permiten la generación de informes finales para los cuales fue reutilizado el componente ReportManager, ubicado en las funcionalidades comunes a otros módulos del sistema. Se reutilizan además aquellas que brinda el EJB Bitácora del módulo Configuración para llevar a cabo el registro de las acciones que realiza el usuario como son: actualizar, modificar, ver, liberar, aceptar y crear alguna entidad.

2.4 Seguridad

La seguridad es de vital importancia para cualquier sistema de información ya que de ella depende la integridad de los datos. Para garantizar la seguridad toda la autorización a directorios, páginas, controles,

[18] Larman, Craig. UML y Patrones. Introdución al análisis y diseño. 2004

opciones del menú, servicios del negocio, está basada en reglas del negocio. Ninguna de estas reglas están contenidas en el código de la aplicación, sino en ficheros no compilados, dotando a la aplicación de mayor dinamismo al no requerir una recompilación en caso que cambie alguna. Esto puede llevarse a cabo por la posibilidad de integración con el motor de reglas JBoss Rules, que brinda el Framework de Seguridad de JBoss Seam.

Se realiza además el control a nivel de usuarios y contraseñas, garantizando el acceso sólo a los niveles establecidos de acuerdo a la función que realiza cada usuario. Las contraseñas sólo pueden ser cambiadas por el propio usuario o por el administrador del sistema. También se validan todos los identificadores o llaves enviadas por el URL a fin de asegurar que sean correctos.

Todas las actividades realizadas por los usuarios quedan registradas a cada momento en una especie de bitácora, almacenándose la fecha, la hora, el usuario y la actividad que realizó el mismo.

2.5 Vista de despliegue

La vista de despliegue muestra la distribución física del sistema y sus conexiones. Cuando se modela la vista de despliegue, normalmente se utilizarán los diagramas de despliegue para modelar dicho sistema.

Un diagrama de despliegue representa la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

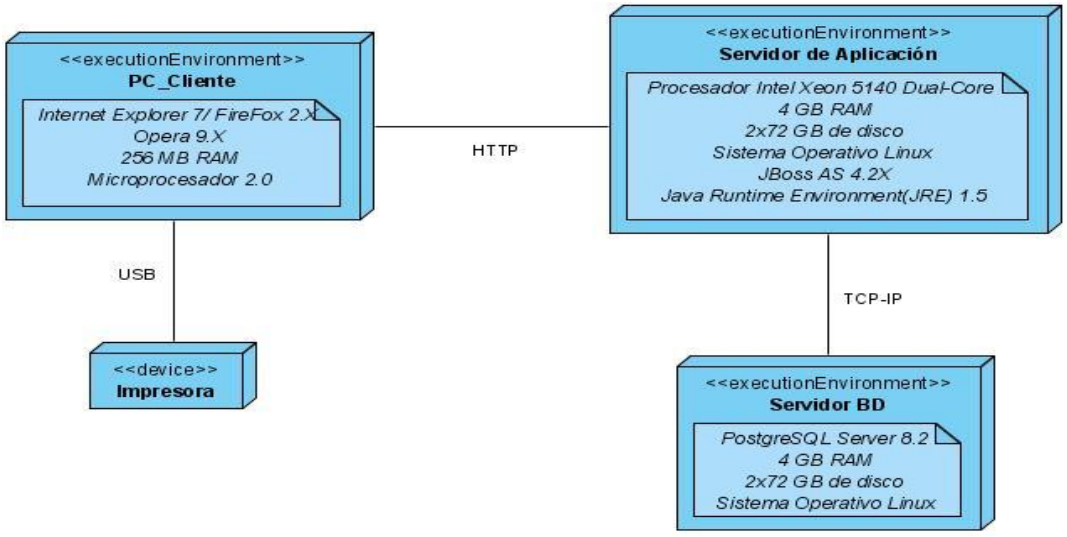

Figura 2.1 Diagrama de despliegue

2.6 Estrategias de codificación. Estándares y estilos a utilizar

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

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

Comentarios, separadores, líneas, espacios en blanco y márgenes

Objetivo: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo 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 identificara el tipo de datos al que se refiere (*Ver tabla 1.1*), en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Ejemplo: sNombrePaciente.

Apariencia de constantes

Todas sus letras en mayúscula.

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

Aspectos generales

Nombres de las variables y constantes.

El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Clases y objetos

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 comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.

Apariencia de atributos

Primera letra en minúscula.

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

Apariencia de las funciones

Primera letra en mayúscula.

Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación 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 sólo 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 utilizara 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 de 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 de 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 utilizara 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 base de datos, vistas, tablas atributos y procedimientos.

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

Controles

Apariencia de los controles

Los controles tendrán un prefijo para el tipo de datos en minúscula.

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

Ejemplo: btnAceptar.

Notación 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	dPesoPaciente
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

Tabla 2.1 Notación de los tipos de datos

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

Casilla de Selección	Cbx	cbxSexo

Tabla 2.2 Notación de los controles

A lo largo del capítulo 2 fueron analizados los aspectos más significativos del sistema a desarrollar, dentro de ellos los requerimientos no funcionales. La descripción de la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones, permite analizar las características del producto para un mejor entendimiento de la solución propuesta. Se investigaron los aspectos relacionados con la seguridad del sistema los cuales, sin lugar a dudas, contribuyen a que el mismo sea robusto, confiable y eficiente. En el diagrama de despliegue se evidencia la propuesta de la distribución física del sistema. Para una mejor comprensión y mantenimiento del software se establecieron los estándares de codificación así como los estilos a utilizar.

Capítulo 3. Descripción y análisis de la solución propuesta

El presente capítulo muestra la descripción y análisis de la solución propuesta. Para ello se realiza una valoración crítica del diseño presentado por el analista, descripción de las nuevas clases u operaciones necesarias, el modelo de datos, una breve valoración de las técnicas de validación, así como la vista de implementación.

3.1 Valoración crítica del diseño propuesto por el analista

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.

Este último describe las clases, su organización en paquetes y subsistemas, sirviendo como abstracción para la implementación y utilizándose como entrada fundamental de las actividades de este flujo de trabajo. Las realizaciones de casos de uso están compuestas por las clases del diseño y las colaboraciones entre ellas.

Para la realización 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, porque pueden ser reusables aplicándose a diferentes problemas en distintas circunstancias, ahorrando tiempo y mejorando el software haciéndolo más eficiente, dinámico y seguro.

La etapa de implementación requiere de un buen diseño de clases basado en patrones que permitan aumentar la reutilización de código y contribuir al óptimo manejo de los sus instancias. Seam asegura la persistencia de datos a través de Hibernate, mediante la abstracción del gestor de bases de datos a utilizar y al mapeo de las entidades del sistema.

EntityManagerFactory, es uno de los componentes principales de la arquitectura, encargado de crear objetos de EntityManager cuando el mismo se inyecte en algún contexto de la aplicación. Dicho componente implementa el patrón de diseño Abstract Factory permitiendo el acceso a la base de datos. EntityManager es la interfaz principal de JPA utilizada para la persistencia de las aplicaciones, cada

instancia puede realizar operaciones como inserción, lectura, modificación y eliminación (CRUD siglas en inglés) sobre un conjunto de objetos persistentes.

Hibernate implementa también el patrón Active Record donde una fila en la tabla de la base de datos se transforma en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. También implementa los patrones de mapeo: Identity Field, Foreign Key Mapping y Association Table Mapping. El primero centra su atención en la forma en que los objetos se relacionan en memoria, consiste básicamente en la generación de un único identificador (ID) para definir la identidad de una entidad. El segundo es utilizado para mapear relaciones de uno a muchos y el tercero, para mapear relaciones de muchos a muchos.

Hibernate implementa además los patrones de comportamiento Identity Map y Lazy Load. El primero es utilizado para evitar tener en memoria dos representaciones distintas del mismo objeto en una transacción de negocio; funciona como una caché de objetos de negocio. El segundo resuelve problemas de carga desmesurada y dependencias circulares o sea optimiza la carga de datos de la base de datos manteniendo en memoria sólo los que se invoquen en cada momento.

Uno de los patrones más importantes que implementa Hibernate es el patrón Query Object que se encarga de interpretar la estructura de objetos y traducirlos a consultas (querys) SQL. A través de este, se pueden crear estas consultas, haciendo referencia a las clases y sus campos, en lugar de tablas y columnas independientemente del esquema en que puedan estar.

En el diseño elaborado se utilizaron los patrones GRASP, patrones para asignar responsabilidades, que tuvieron una importante utilidad en el diseño realizado. A cada clase le fueron asignadas las tareas que podían realizar según la información que poseía, además de crear las instancias de otras clases en correspondencia con la responsabilidad dada, poniéndose de manifiesto los patrones Experto y Creador siendo el primero uno de los más usados. Con esto de logró conservar el encapsulamiento pues los objetos logran valerse de su propia información para realizar lo que se les pide.

El diseño obtenido cumple con los patrones de Bajo Acoplamiento y Alta Cohesión permitiendo la colaboración entre los elementos del diseño (clases), sin verse afectados la reutilización de los mismos y el entendimiento de estos cuando se encuentran aislados. La creación de clases controladoras facilitó realizar las operaciones del sistema, debido a que estas reflejan los procesos de la empresa o dominio y no es factible manejarse en la capa de interfaz o presentación.

Respondiendo a la arquitectura definida se realizó el modelo de diseño. Los criterios de empaquetamiento quedaron definidos por procesos y clases. Cada proceso definido en el sistema se grafica en un paquete correspondiente, haciendo uso de las clases que se encuentran dentro del paquete del repositorio de clases estructurado en tres subpaquetes:

- Secciones: Contiene las clases controladoras autogeneradas, las controladoras personalizadas y las controladoras propias del proceso. Las primeras son autogeneradas por el entorno de desarrollo, las segundas son aquellas que se modifican y las terceras son propias del proceso.
- Entidades: Contiene las entidades autogeneradas de la base de datos y personalizadas. Las autogeneradas son aquellas obtenidas mediante el proceso de ingeniería inversa de la base de datos, utilizando para ello el mapeo objeto-relacional (ORM siglas en inglés) de Hibernate. Las personalizadas son aquellas modificadas y, en algunos casos, pueden presentar relaciones de herencia o composición con las autogeneradas.
- Vistas: Contiene el conjunto de clases que conforman o representan la interfaz de usuario.

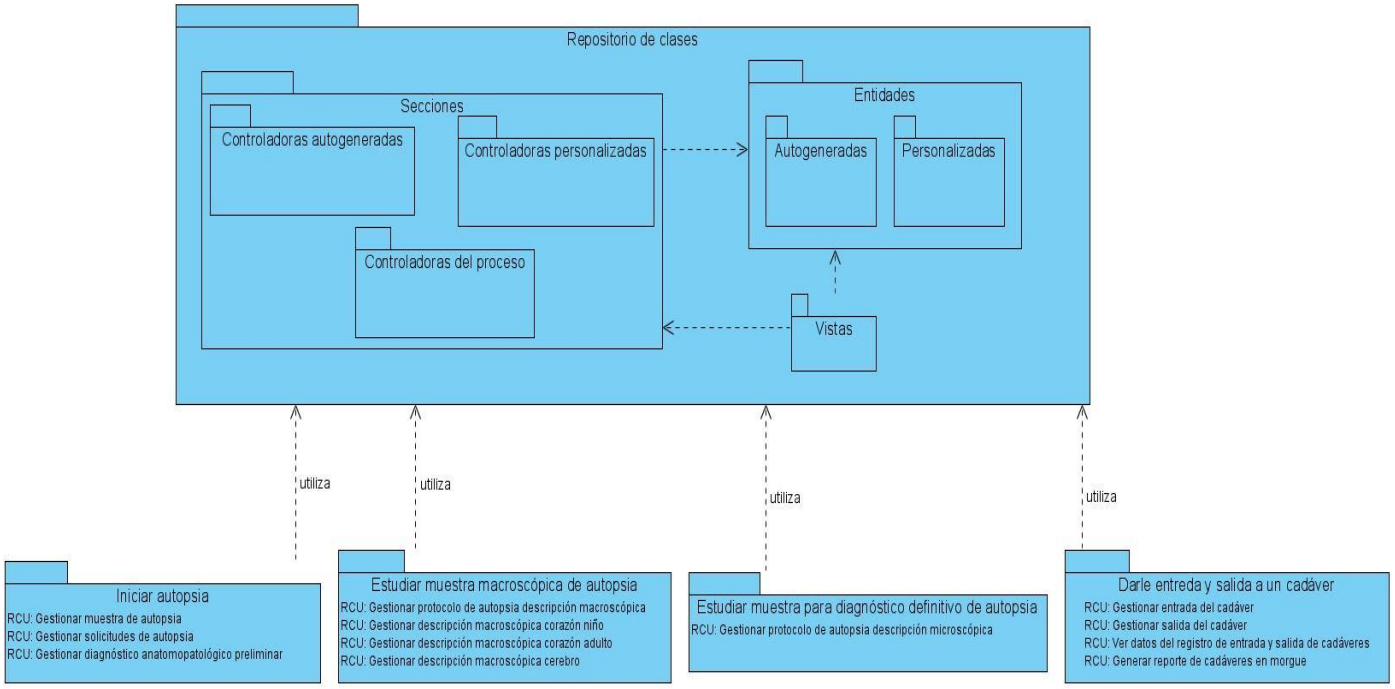

Figura 3.1 Diagrama de paquetes

Existen varios elementos que componen al modelo de diseño entre los que se destacan los diagramas de clases de diseño y los diagramas de interacción. Los primeros, capturan la estructura lógica del sistema, es decir, las clases y los elementos que constituyen al modelo. Los diagramas de clases son los más útiles para ilustrar las relaciones entre las clases e interfaces. Por su parte, los diagramas de interacción describen cómo grupos de objetos colaboran para conseguir algún fin, mostrando a dichos objetos así como los mensajes entre ellos dentro de la realización del caso de uso.

Para la realización de los diagramas de clases del diseño se utiliza la extensión de UML para la utilización de estereotipos web. Esta extensión presenta como elementos más significativos tres clases UML "página servidora", "página cliente" y "formulario" empleadas para el código servidor, código cliente y formularios respectivamente.

El código servidor se encarga de construir o generar el resultado XHTML que conforma el código cliente (<<build>>), los formularios envían sus datos al código servidor para ser procesados los pedidos (<<submit>>), además forman parte del código cliente o resultado XHTML. Es por esto que la relación entre la clase empleada para el código cliente y la clase empleada para el formulario es de agregación. Entre las páginas clientes pueden existir vínculos (<<link>>) o redireccionamientos (<<redirect>>). Es importante destacar que una página cliente es construida por una sola página servidora; esta a su vez, puede completar su funcionamiento incluyendo código existente en otra página de este mismo tipo, utilizando la relación de inclusión (<<include>>), que aunque no es propia de la extensión de UML, las herramientas de modelado la consideran para representar todas las relaciones existentes en el modelo.[19]

Las vistas corresponden a las realizaciones de los casos de uso, conteniendo cada una, el diagrama de clases de diseño y los distintos diagramas de secuencia correspondientes.

[19] Navarro Franco, Ángel José. UML en acción. Modelando Aplicaciones Web. Instituto Superior Politécnico José Antonio Echeverría. La Habana, Cuba, Mayo 2005.

Diagramas de clases del diseño

Proceso: Dar entrada y salida a un cadáver

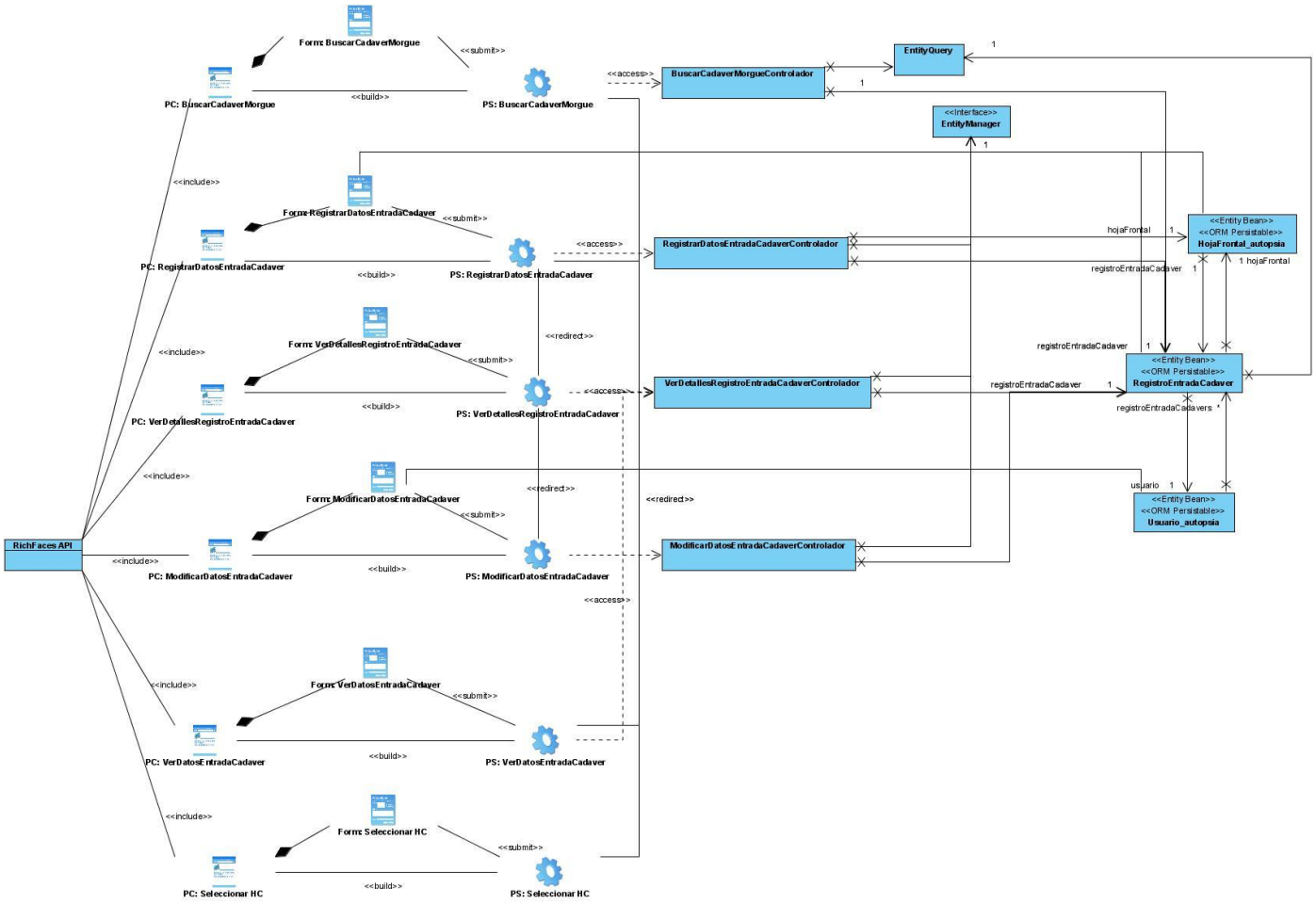

Figura 3.2 DCD_Gestionar entrada del cadáver

Diagramas interacción

Proceso: Dar entrada y salida a un cadáver

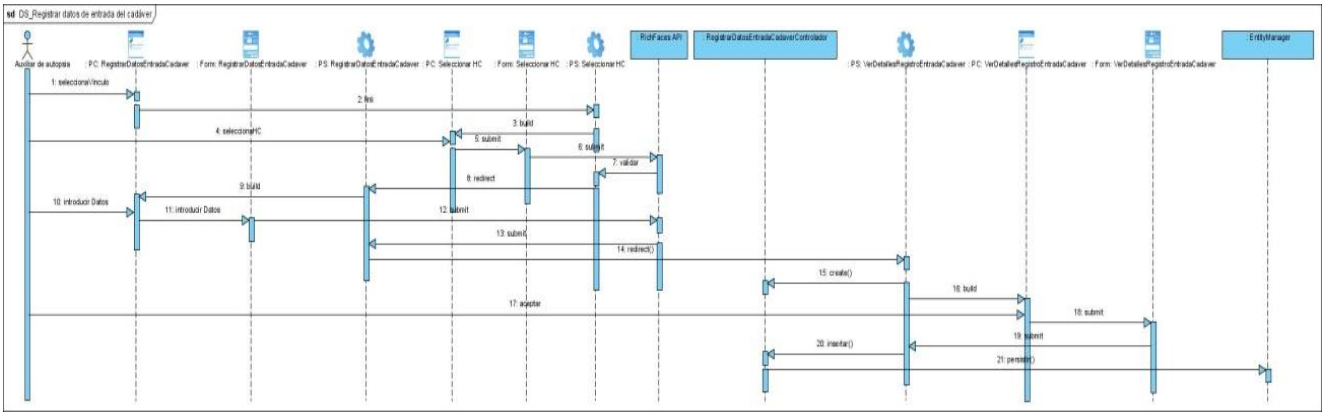

Figura 3.3 DS_ Registrar datos de entrada del cadáver

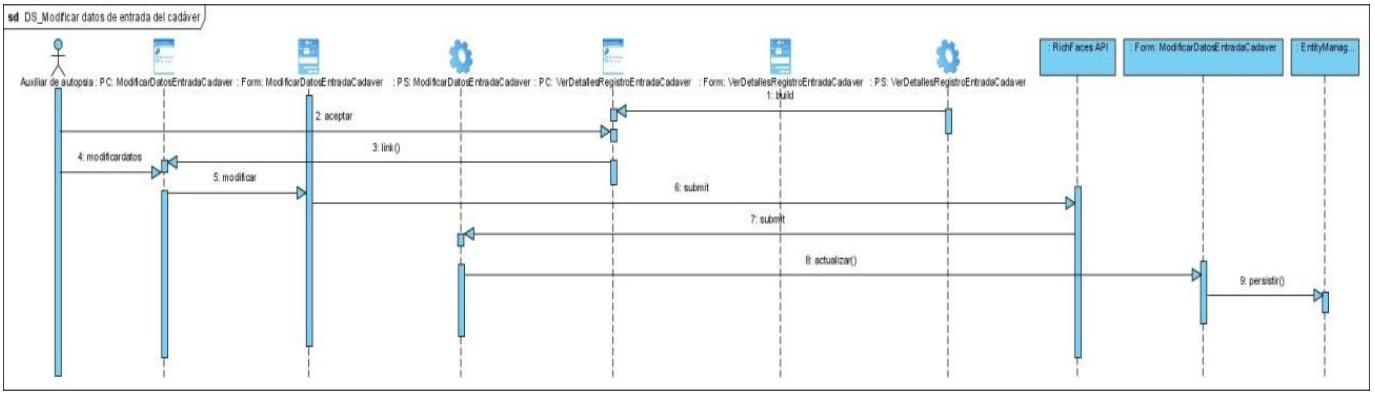

Figura 3.4 DS_ Modificar datos de entrada del cadáver

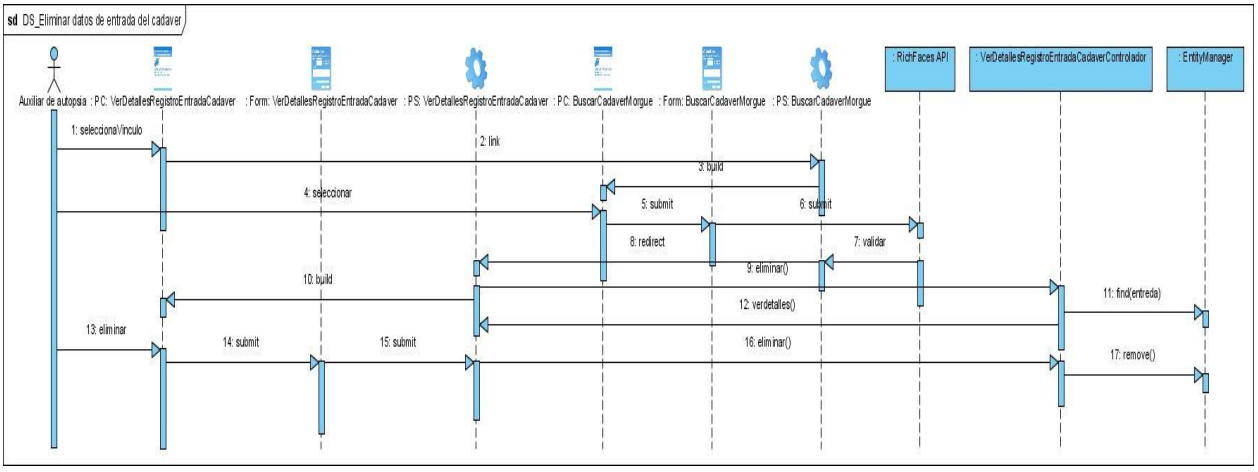

Figura 3.5 DS_ Eliminar datos de entrada del cadáver

3.2 Descripción de las nuevas clases u operaciones necesarias.

Nombre: RegistarDatosEntradaCadaverControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
hojaFrontalId	Integer
hora	String
minutos	String
ampm	String
usuario	Usuario
entityManager	EntityManager
hojaFrontal	hojaFrontal_autopsia
registroEntradaCadaver	RegistroEntradaCadaver
Para cada responsabilidad:	
Nombre:	create()
Descripción:	Permite crear un objeto de la entidad RegistroEntradaCadaver
Nombre:	init()
Descripción:	Permite inicializar los valores del objeto
Nombre:	cargar()
Descripción:	Permite una vez realizada la solicitud cargar los datos hacia la página de ver los detalles.
Nombre:	insertar()
Descripción:	Una vez que se hallan introducidos los datos correspondiente de forma satisfactoria inserta el objeto en la base de datos.

Tabla 3.1 RegistarDatosEntradaCadaverControlador

Nombre: ModificarDatosEntradaCadaverControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
registroEntradaCadaverId	Integer
hora	String
Minutos	String
ampm	String
entityManager	EntityManager
registroEntradaCadaver	RegistroEntradaCadaver
Para cada responsabilidad:	
Nombre:	eliminar()
Descripción:	Elimina el objeto que este viendo el usuario.

3.3 Modelo de datos

Los modelos de datos constituyen una base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. Un modelo de datos es la estructura o representación física de las tablas de la base de datos.

Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas y actualizaciones de datos. Además, los más modernos incluyen conceptos para especificar comportamiento, permitiendo especificar un conjunto de operaciones definidas por el usuario. A la descripción de una base de datos mediante un modelo de datos se le denomina, esquema de la base de datos. Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo.

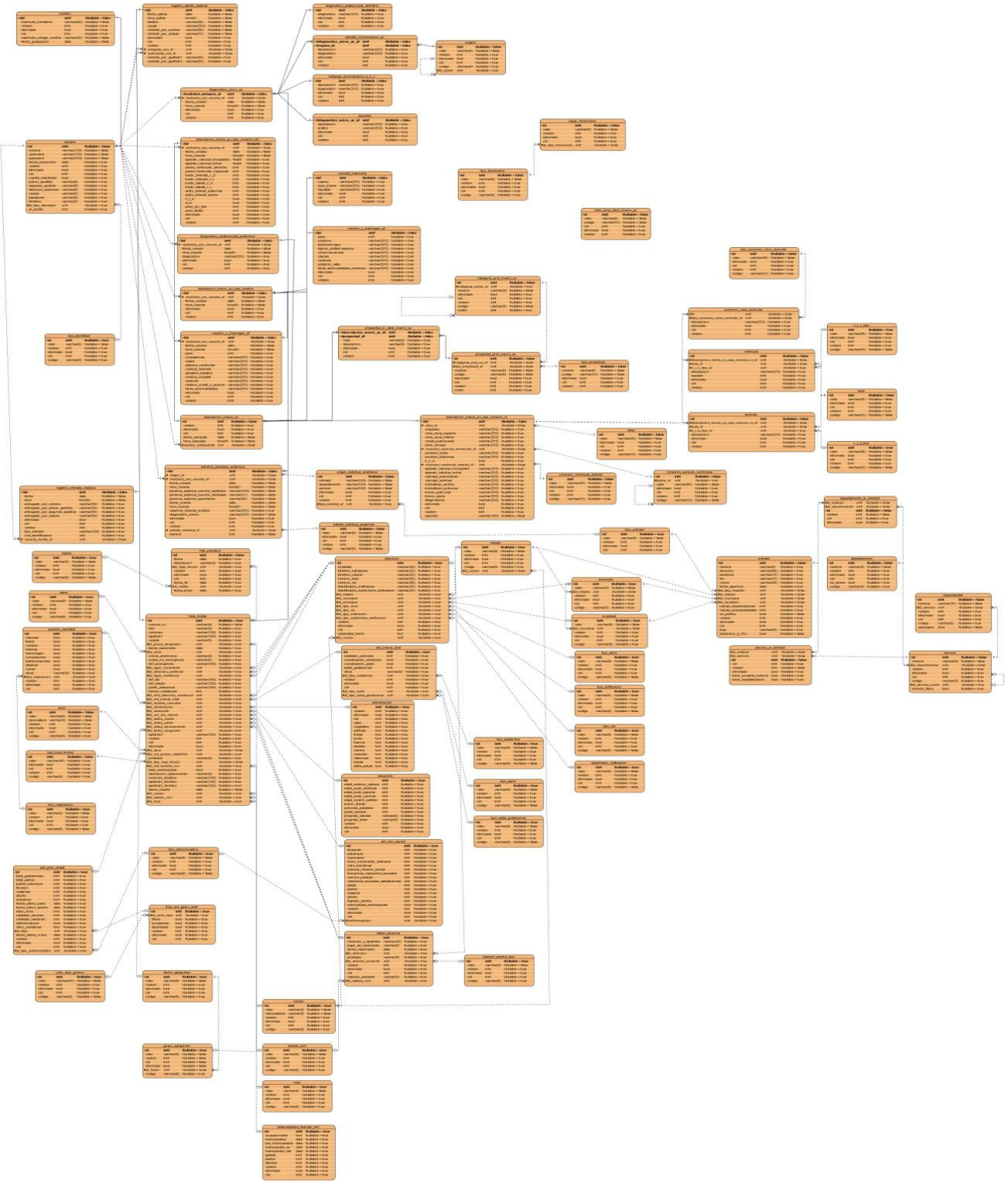

Figura 3.6 Modelo de datos

3.4 Valoración de las técnicas de validación

Cuando se desarrollan aplicaciones, comúnmente son realizadas validaciones de las entradas del usuario. Frameworks como JSF proporcionan ayudas para definir estas validaciones en la vista. Esto provoca repetir las validaciones una y otra vez. Hibernate Validator se utiliza para resolver este problema. Esto sucede de la siguiente forma: se define una única vez las validaciones en los objetos de negocio (POJOS), y se invocan dichas validaciones desde el punto que interese.

Las principales características de Hibernate Validator son:

- Definición de las validaciones muy fácilmente con anotaciones.
- Trae un conjunto predefinido de validaciones típicas, conjunto que se puede extender fácilmente con las validaciones propias.
- El sistema soporta internacionalización: Trae mensajes de errores traducidos a diez idiomas. Estos mensajes se pueden cambiar fácilmente, simplemente escribiendo un fichero de propiedades y sobrescribiendo los que interese.
- Se integra directamente con Hibernate, y en general con cualquier ORM, de forma que antes de hacer una inserción o actualización se validarán los objetos.
- Si se usa Hibernate, las validaciones que se indiquen se tendrán en cuenta a la hora de generar el DDL.[20]

Descripción de las tablas.

Nombre: RegistroEntradaCadaver		
Descripción: Tabla para registrar los datos de entrada del cadáver		
Atributo	**Tipo**	**Descripción**
id	Integer	Identificador de la tabla
version	Integer	Atributo para persistir o actualizar la entidad
codIdentificacion	Integer	Código de identificación del cadáver

[20] Hibernate Validator, y como definir las validaciones sobre los objetos de negocio.2008

fecha	Date	Fecha de entrada del cadáver
hora	Date	Hora de entrada del cadáver
entregadoPorNombre	String	Persona que realiza la entrega
tipoEntidad	String	Tipo de entidad
entregadoPorPrimerApellido	String	Primer apellido de la persona que realiza la entrega
entregadoPorSegundoApellido	String	Segundo apellido de la persona que realiza la entrega
entregadoPorCedula	String	Cédula de la persona que realiza la entrega
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado
cid	Integer	Identificador de modificaciones registradas en la bitácora
hojaFrontal	HojaFrontal_autopsia	Objeto de la entidad HojaFrontal_autopsia
usuario	Usuario_autopsia	Objeto de la entidad Usuario_autopsia
registroSalidaCadaver	RegistroSalidaCadaver	Objeto de la entidad RegistroSalidaCadaver

Tabla 3.2 RegistroEntradaCadaver

3.5 Vista de implementación.

La vista de implementación describe la descomposición del software en capas y subsistemas de implementación. También provee una vista de la trazabilidad de los elementos de diseño de la vista lógica ahora para la implementación.

Esta contiene:

- Una enumeración de los subsistemas.
- Diagramas de componentes que ilustran la organización en capas y jerarquías de los subsistemas.
- Dependencia entre subsistemas.

Diagrama de componentes

Los diagramas de componentes modelan los aspectos físicos de un sistema, muestran un conjunto de componentes y sus relaciones. En UML se utilizan para visualizar los aspectos estáticos de los componentes físicos y sus relaciones, además para especificar sus detalles para la construcción. Esto implica modelar los elementos físicos que residen en un nodo, tales como ejecutables, bibliotecas, tablas, archivos y documentos. Los elementos que lo componen son:

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

A continuación se muestran como quedaron distribuidos los componentes en el sistema propuesto.

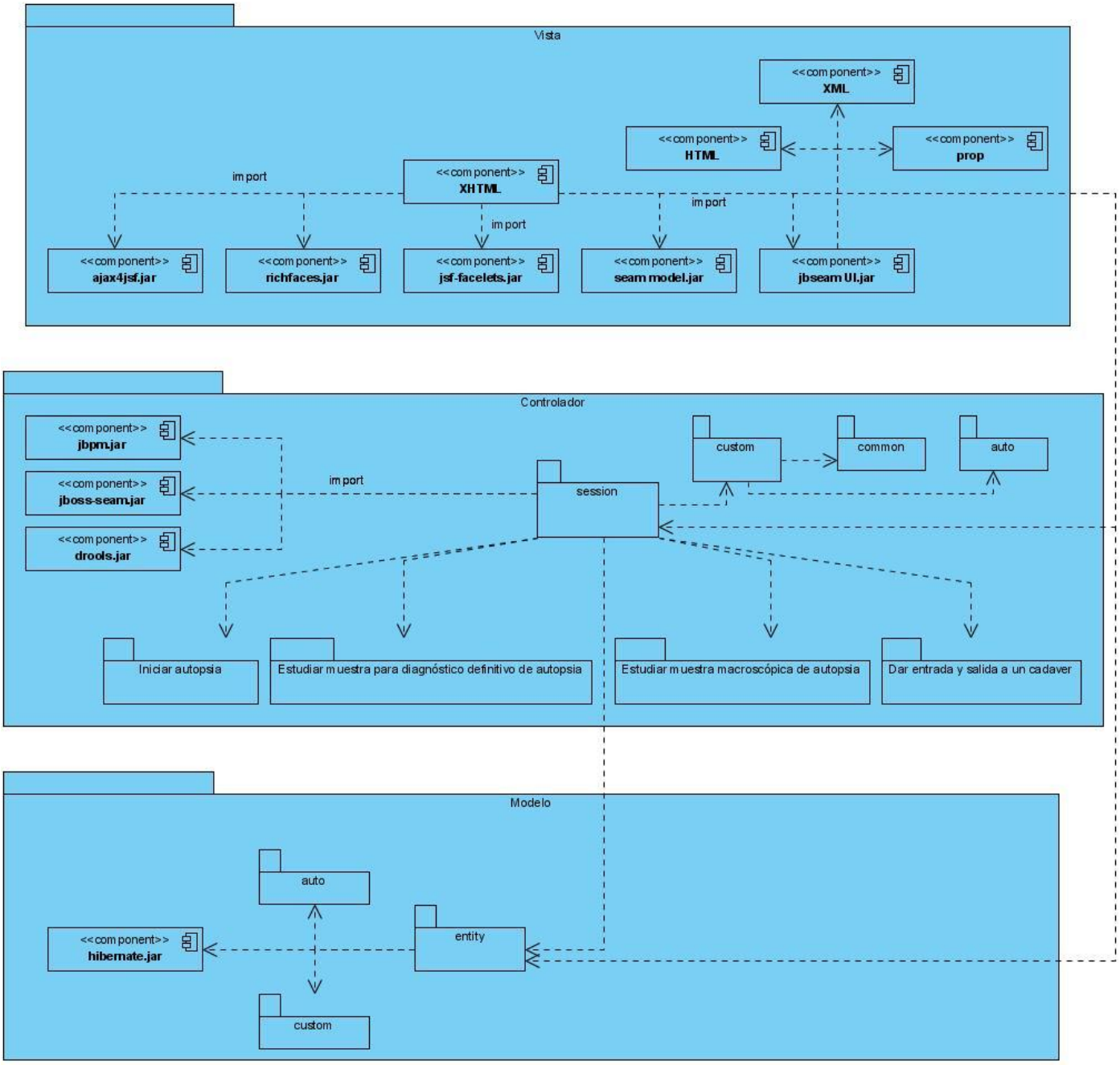

Figura 3.7 Diagrama de componentes

3.6 Descripción de los algoritmos no triviales a implementar. Análisis de su complejidad

Un algoritmo es una secuencia de instrucciones cuyo objetivo es dar solución a un problema. El término clave es el de problema**.** La complejidad de un algoritmo es una métrica teórica que da una idea de cuánto va a tardar un algoritmo en resolver dicho problema.

Para el desarrollo del proceso realizar autopsia del módulo Anatomía Patológica del sistema de información hospitalaria alas HIS se implementan algoritmos de alta complejidad. Dentro de los que se encuentran:

- Crear protocolo de autopsia descripción macroscópica.
- Modificar protocolo de autopsia descripción macroscópica.
- Ver detalles de protocolo de autopsia descripción macroscópica.
- Ver datos de protocolo de autopsia descripción macroscópica.
- Eliminar protocolo de autopsia descripción macroscópica.
- Crear descripción macroscópica corazón niño.
- Modificar descripción macroscópica corazón niño.
- Crear descripción macroscópica corazón adulto.
- Modificar descripción macroscópica corazón adulto.
- Crear descripción macroscópica cerebro.
- Modificar descripción macroscópica cerebro.
- Crear protocolo de autopsia diagnóstico microscópico.
- Modificar protocolo de autopsia diagnóstico microscópico.
- Ver detalles de protocolo de autopsia diagnóstico microscópico.
- Ver datos de protocolo de autopsia diagnóstico microscópico.
- Eliminar protocolo de autopsia diagnóstico microscópico.
- Generar reporte de cadáveres en morgue.

3.7 Selección de las estructuras de datos apropiadas para la implementación de estos algoritmos

La estructura de datos es un conjunto de variables de un determinado tipo agrupadas y organizadas de alguna manera para representar un comportamiento. Lo que se pretende es facilitar un esquema lógico para manipular los datos en función del problema que haya que tratar y el algoritmo para resolverlo. En algunos casos la dificultad para resolver un problema radica en escoger la estructura de datos adecuada. En general, la elección del algoritmo y de las estructuras de datos que manipulará, estarán muy relacionadas.[21]

El paquete java.util de Java contiene implementaciones de estructuras de datos. Una de estas y la que se utilizarán son las listas, que no son más que un conjunto de elementos con un orden concreto. Estas listas son interfaces que están compuestas por varios métodos. El ArrayList es una clase que implementa la interfaz.

3.8 Descripción de las clases que se utilizan para representar computacionalmente la estructura

En las listas se tiene un control preciso sobre el lugar donde se desea insertar cada elemento. Se puede acceder a los elementos por su índice de número entero (posición en la lista), y la búsqueda de elementos en la lista. A diferencia de otros conjuntos, las listas suelen permitir elementos duplicados y múltiples elementos nulos.

La interfaz de la lista implementa métodos fundamentales como son: iterador, agregar, quitar, es igual, posición, y los métodos hashCode. Estas operaciones se pueden ejecutar en un tiempo proporcional al valor del índice para algunas implementaciones (la clase LinkedList, por ejemplo).

Iterar sobre los elementos de una lista es normalmente preferible a la indexación a través de ella, si la persona que llama no sabe la aplicación. La lista proporciona un iterador especial, llamado ListIterator, que permite la inserción de elementos y de sustitución, y el acceso bidireccional, además de las operaciones normales que la interfaz de Iterator proporciona.

Para buscar un objeto especificado la interfaz lista proporciona dos métodos. Desde un punto de vista de rendimiento, estos métodos deben ser utilizados con precaución. En muchas implementaciones van a realizar costosas búsquedas lineales. Proporciona dos métodos para insertar y extraer de manera eficiente varios elementos en un punto arbitrario.

[21] http://www.algoritmia.net/articles.php?id=10 7de enero del 2003

Se aconseja gran precaución con los métodos hashCode pues ya no están bien definidos en una lista que se contiene a sí mismas como elemento. Algunas implementaciones de la lista tienen restricciones en los elementos que puedan contener. Por ejemplo, algunas implementaciones de prohibir elementos nulos, y algunos tienen restricciones sobre los tipos de sus elementos. El intento de agregar un elemento no subvencionable se produce una excepción, generalmente NullPointerException o ClassCastException.[22]

Al finalizar el capítulo han sido analizados los aspectos significativos del diseño e implementación del sistema y se generaron los principales artefactos de los flujos de trabajo. Entre estos se encuentran los diagramas de clases del diseño de los casos de uso arquitectónicamente significativos, así como los diagramas de secuencia correspondientes y el modelo de datos, los cuales serán de gran ayuda para la futura implementación del sistema. Además fue construido el diagrama de componentes mostrándose la relación existente entre las clases de diseño y la arquitectura del sistema.

[22] Java.util Public interface: List. http://www.docjar.com/

Capítulo 4. Modelo de prueba

El presente capítulo muestra el modelo de prueba. Para ello se realiza una caracterización de las pruebas de caja negra como método a utilizar. Se definen y describen los casos de prueba.

4.1 Prueba de caja negra

Las pruebas de un software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. La creciente percepción del software como un elemento del sistema y la importancia de los costes asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas.

Cualquier producto de ingeniería puede ser probado conociendo la funcionalidad específica para la cual fue diseñado, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa. Este enfoque se denomina prueba de caja negra.

La prueba de caja negra se centra principalmente en los requisitos funcionales del software permitiendo obtener un conjunto de condiciones de entrada que ejerciten estos requisitos y se ignora la estructura de control. No son más que las pruebas que se realizan sobre la interfaz del software.

Para desarrollar dicha prueba existen varias técnicas, entre ellas están:

- Técnica de la partición de equivalencia.
- Técnica del análisis de valores límites.
- Técnica de grafos de causa-efecto.

Se utilizará la técnica de la Partición de equivalencia que divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Un caso de prueba es un conjunto de entradas de pruebas donde se describen escenarios y se identifican variables. Estas últimas representa a un conjunto de estados válidos o inválidos para las condiciones de entrada para demostrar que las funciones del software son operativas.

Descripción de los casos de pruebas

Caso de prueba: Registrar datos de entrada del cadáver

Escenarios del Registrar datos de entrada del cadáver	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Registrar datos de entrada del cadáver	Registrar datos de entrada del cadáver satisfactoriamente.	Se selecciona la opción Registrar datos de entrada del cadáver. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Muestra la interfaz Ver detalles de la entrada del cadáver. Ver DCP: **Ver detalles de la entrada del cadáver.**
EC 2: Cancelar operación	Cancelar la opción de registrar datos de entrada del cadáver.	Se selecciona la opción Registrar datos de entrada del cadáver. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista de seleccionar historia clínica.
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 Registrar datos de entrada del cadáver. 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 Registrar datos de entrada del cadáver. Se introducen los datos incorrectos. Se selecciona la opción Aceptar.

		Muestra un indicador sobre los campos incorrectos.

Tabla 3.3 CP_Registrar datos de entrada del cadáver

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Registrar datos de entrada del cadáver	Cancelar operación	Existen datos incompletos	Existen datos incorrectos
Variable 1 (Fecha de ingreso)	V		I	I
Variable 2 (Hora de ingreso)	V		I	I
Variable 3 (Nombre de la persona que entrega)	V		I	I
Variable 4 (Apellido1 de la persona que entrega)	V		I	I
Variable 5 (Apellido2 de la persona que entrega)	V		I	I
Variable 6 (Cédula de identificación)	V		I	I
Variable 7 (Establecimiento)	V		I	I
Variable 8 (Nombre de la persona que recibe)	V		I	I
Variable 9 (Apellido1 de la persona que recibe)	V		I	I
Variable 10 (Apellido2 de la persona que recibe)	V		I	I

Variable 11 (Cédula de identificación)	V		I	I
Botón 1 (Aceptar)	NA		NA	NA
(Cancelar)		NA		
Respuesta del Sistema	Registra los datos de entrada del cadáver y se visualiza la interfaz ver detalles.	Regresa a la vista de seleccionar historia clínica.	Muestra un indicador sobre los campos incompletos.	Muestra un indicador sobre los campos incorrectos.
Resultado de la Prueba				

Tabla 3.4 SC_Registrar datos de entrada del cadáver

Caso de prueba: Modificar datos de entrada del cadáver

Escenarios del Modificar datos de entrada del cadáver	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Modificar datos de entrada del cadáver	Modificar datos de entrada del cadáver satisfactoriamente.	Muestra la interfaz para modificar datos de entrada del cadáver. Se modifican los datos correspondientes. Se selecciona la opción Aceptar. Se modifican los datos de entrada del cadáver. Muestra la interfaz Ver detalles de datos de entrada del cadáver. Ver DCP **Ver detalles de la entrada del cadáver**.
EC 2: Cancelar operación	Cancelar la opción de Modificar datos de entrada del cadáver.	Muestra la interfaz para modificar datos de entrada del cadáver. Se introducen 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.	Muestra la interfaz para modificar datos de entrada del cadáver. 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.	Muestra la interfaz para modificar datos de entrada del cadáver. Se introducen los datos incorrectos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incorrectos.

Tabla 3.5 CP_Modificar datos de entrada del cadáver

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Modificar datos de entrada del cadáver satisfactoriamente	Cancelar operación	Existen datos incompletos	Existen datos incorrectos
Variable 1 (Fecha de ingreso)	V		I	I
Variable 2 (Hora de ingreso)	V		I	I
Variable 3 (Nombre de la persona que entrega)	V		I	I
Variable 4 (Apellido1 de la persona que entrega)	V		I	I
Variable 5 (Apellido2 de la persona que entrega)	V		I	I

Variable 6 (Cédula de identificación)	V		I	I
Variable 7 (Establecimiento)	V		I	I
Variable 8 (Nombre de la persona que recibe)	V		I	I
Variable 9 (Apellido1 de la persona que recibe)	V		I	I
Variable 10 (Apellido2 de la persona que recibe)	V		I	I
Variable 11 (Cédula de identificación)	V		I	I
Botón 1 (Aceptar)	NA		NA	NA
(Cancelar)		NA		
Respuesta del Sistema	Modifica la datos de entrada del cadáver 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				

Tabla 3.6 SC_Modificar datos de entrada del cadáver

Caso de prueba: Eliminar datos de entrada del cadáver

Escenarios del eliminar datos de entrada del cadáver	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Eliminar datos de entrada del cadáver	Eliminar datos de entrada del cadáver satisfactoriamente.	Se muestra el mensaje: "Se eliminará la datos de entrada del cadáver seleccionada. Al seleccionar Sí se perderán todos los datos. ¿Desea continuar?". Selecciona la opción Si. Muestra la interfaz de seleccionar historia clínica.
EC 2: Cancelar operación	Cancelar la opción de eliminar datos de entrada del cadáver.	Se muestra el mensaje: "Se eliminará la datos de entrada del cadáver seleccionado. Al seleccionar Sí se perderán todos los datos. ¿Desea continuar?". Se selecciona la opción de No. Se regresa a la vista de ver detalles de la entrada del cadáver.

Tabla 3.7 CP_Eliminar datos de entrada del cadáver

Id del escenario	EC 1	EC 2
Escenario	Eliminar datos de entrada del cadáver satisfactoriamente	*Cancelar operación*
Botón 1 (Si)	NA	
Botón 2 (No)		NA
Respuesta del Sistema	Muestra la interfaz de seleccionar historia clínica.	Regresa a la vista de ver detalles de la entrada del cadáver.
Resultado de la Prueba		

Tabla 3.8 SC_Eliminar datos de entrada del cadáver

Capítulo 4: Modelo de prueba

En este capítulo se realizó el modelo de prueba. Haciendo uso de la técnica Partición de equivalencia se utilizó el método de prueba: Caja negra, obteniéndose los casos de prueba como artefactos generados del flujo de trabajo.

Conclusiones

En correspondencia con las exigencias del módulo Anatomía Patológica y los objetivos trazados en la investigación se concluye:

- Los sistemas analizados a pesar de brindar soluciones avanzadas no fueron los más factibles a utilizar por concepto de costos, requerimientos y flexibilidad.
- La arquitectura, tecnologías y herramientas definidas, favorecieron el desarrollo de la aplicación web, con una interfaz sencilla, robusta y flexible que gestiona correctamente la información generada por los procesos analizados.
- La implementación de las funcionalidades correspondientes al proceso realizar autopsia del módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS, facilitará la gestión de información en esta área de las instituciones hospitalarias.
- Las pruebas exploratorias efectuadas a las funcionalidades del proceso realizar autopsia del módulo de Anatomía Patológica del Sistema de Información Hospitalaria alas HIS, garantizaron la calidad de la solución propuesta.

Recomendaciones

Una vez cumplido el objetivo general del trabajo, se puede constatar de que nuevas ideas han surgido durante el desarrollo del mismo, dando margen a que en el futuro se pueda implementar un sistema con nuevas funcionalidades, que posibilite un control más extensible de la información generada por el área de Anatomía Patológica dentro de las instituciones hospitalarias, por lo que se recomienda:

- Estandarizar los diagnósticos emitidos por los patólogos para apoyar la generación de reportes estadísticos.

Bibliografía

1. Ajax4jsf Developer Guide. 2007.
2. Allen, Dan. Seam in Action. 2008.
3. Anatomía Patológica: La gran desconocida de las especialidades médicas. Oviedo Ramirez, MI., Ortiz Ruiz, E., Monzones, C. 13, España : s.n., Junio 2008, Enfermería Global.
4. Cristian Bauer, Gavin King. Java Persistence with Hibernate. 2005.
5. García Linares, A.J., Reche Martínez, D. y Richarte Reina, J.M. xHIS: Un nuevo concepto en Sistemas de Información Hospitalarios. Dpto. I+D+I. Novasoft Sanidad. Grupo Novasoft : s.n.
6. Herrero Santoja, Joaquín. La experiencia Kewan - Cosmosalud.
7. Ing. Nadiezka Milán Cristo, Ing. Yeinier Ferras Cecilio. Sistema de Anatomía Patológica para instituciones hospitalarias. UCI : s.n.
8. Javid Jamae, Peter Johnson. JBoss in Action. 2009.
9. Jose Alain Senra Gutiérrez, Elier Broche Cristo. Sistema de gestión tecnológica hospitalaria V 1.0. Instituto Superior Politécnico “José Antonio Echeverría” : s.n., 1999-2000.
10. Josué Díaz, Iliana Giménez, Julio López, Cesar Narváez, Richard Rivas. [En línea] [Citado el: 20 de noviembre de 2009.] http://unesr.files.wordpress.com/2007/03/tesis_01-b.pdf.
11. JuanaG. Laboratorio Anatomía Patológica. [En línea] 19 de diciembre de 2008. [Citado el: 2 de marzo de 2010.] http://laboratorioanatomapatolgica.blogspot.com/2008/12/laboratorio-anatoma-patolgica.html.
12. La autopsia (Autopsy). [En línea] [Citado el: 12 de enero de 2010.] http://www.healthsystem.virginia.edu/uvahealt/adult_path_sp/autopsy.cmf.
13. La tecla de escape. [En línea] 2006. [Citado el: 27 de Marzo de 2010.] http://latecladeescape.com/w0/basico/que-es-la-complejidad-de-un-algoritmo.html.
14. Larman, Craig. UML y Patrones. Introdución al análisis y diseño. 2004.

15. M.M. García Bonafé, A. Contestí, N. Pérez, A. Gámez. Informatización electrónica de un servicio de Anatomía Patológica y su integración en el sistema informático global hospitalario. Hospital Son Llàtzer. Palma de Mallorca : s.n.
16. Marcial García Rojo, Francisco Javier Pardo Mindán. Anatomía Patológica (Patología) en la Historia de Salud Electrónica.
17. Milán Cristo, Nadiezka. Glosario de Términos. Anatomía Patológica. Ciudad de la Habana : s.n., 2009.
18. Milán Cristo, Nadiezka. IH-SW-DE-020 ALAS-HIS_Anatomía Patológica_Glosario de términos. UCI : s.n., 2008.
19. Milán Cristo, Nadiezka. Resumen por procesos del módulo Anatomía Patológica. UCI : s.n., 2008.
20. Ministerio de la Informática y las Comunicaciones de Cuba. [En línea] 2002. [Citado el: 20 de noviembre de 2009.] http://www.mic.gov.cu/hinfosoc.aspx.
21. Obregón Martin, Liuvyn. IH-SW-DR-090 ALAS-HIS_Requerimientos suplementarios. UCI : s.n., 2008.
22. Pérez García, Alejandro. Hibernate Validator, y como definir las validaciones sobre los objetos de negocio. España : s.n., 2008.
23. Pressman, Roger S. Ingeniería de software, un enfoque práctico. 2005.
24. Red Hat. RichFaces developer Guide. 2007.
25. Registro nacional de autopsias en Cuba. Utilización del SARCAP. José Hurtado de Mendoza Amat, Reynado Álvarez Santana. 1, España : s.n., 2004, Vol. 37.
26. Rosa María Coro Antich, Joaquín Galarraga Inza, Héctor Gómez Suárez, María Xiomara Gil Gil. NEUPAT: Proyecto de informatización para Laboratorio de Neuropatología. Octubre 2006.
27. Salvador Oliván, José Antonio. Sistemas de informacion hospitalarios: el C.M.B.D. Universidad de Zaragoza : s.n.
28. The postgreSQL Global Develoment Group. PostgreSQL 8.3.6 Documentation. 2008.
29. UI Develoment with JaveServer Faces.

30. Velázquez Carralero, Alejandro Mario. IH-SW-DR-091 ALAS-HIS_Documento de Arquitectura del Sistema. UCI : s.n., 2008.

31. VI Congreso virtual hispanoamericano de Anatomía Patológica. [En línea] 2003. [Citado el: 20 de noviembre de 2009.] http://conganat.uninet.edu/6CVHAP/autores/trabajos/T067/index.html.

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.

Autopsia: Examen anatómico de un cadáver. Examen analítico minucioso.

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

Biopsia: Muestra de tejido tomada de un ser vivo, con fines diagnósticos. Resultado del examen de esta muestra.

Citología: Parte de la biología que estudia la célula. Procedimiento diagnóstico basado en el examen de las células contenidas en un exudado o trasudado.

DDL: Scripts de creación de la base de datos.

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

ICEFaces: Biblioteca de componentes JSF Ajax.

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

Muestra: Porción de tejido o líquido extraído de la persona a la cual se le realizará una biopsia, citología o autopsia.

PACs: Sistema de archivo y comunicación de imágenes.

Patólogo: Doctor que se especializa en el análisis de tejido bajo el microscopio y diagnostica enfermedades.

Médico especialista en identificar enfermedades (patologías) estudiando las células y tejidos en el microscopio.

Pieza quirúrgica: Tipo de biopsia de mayor tamaño, a menudo comprende órganos e incluyen toda la lesión.

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

Protocolo de autopsia: Plan escrito y detallado de una autopsia. Recoge el resumen de historia clínica, protocolo macroscópico, protocolo microscópico, correlación clínica-patológica y diagnostico final.

RIS: Sistema de Información Radiológico.

Servicio de apoyo: Servicio especializado que brinda los medios de diagnóstico para apoyar la determinación o corroboración del diagnóstico médico.

Smalltalk: Lenguaje orientado a objetos.

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