Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de diploma para optar por el título

Ingeniero en Ciencias Informáticas

TEMA: Diseño e implementación del módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS

TÍTULO: Implementación de los procesos generar informes y procesar muestras del módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS

Autores: Ismael Bell Sánchez

Lisandra Estupiñán Pérez

Tutores: Ing. Yeinier Ferrás Cecilio

Ing. Francisco Rodríguez Torres

Ciudad de La Habana, julio de 2010

"Año 52 de la Revolución"

DATOS DE CONTACTO

Tutores:

Ing. Yeinier Ferrás Cecilio

Ing. Francisco Rodríguez Torres

Síntesis de los tutores

Ing. Yeinier Ferrás Cecilio: Graduado de ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en julio del 2007. Actualmente posee la categoría docente de Profesor Instructor y ha impartido las siguientes asignaturas: Física (curso 2007-2008), Práctica Profesional (curso 2008-2009), actualmente imparte Aplicaciones Informáticas en el Sector de la Salud. Pertenece al Centro Especializado en Soluciones de Informática Médica y dentro de este labora como jefe de módulo en el Departamento de Gestión Hospitalaria.

Correo electrónico: yferras@uci.cu

Ing. Francisco Rodríguez Torres: 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.

Correo electrónico: frtorres@uci.cu

RESUMEN

El siguiente trabajo tiene como objetivo desarrollar las funcionalidades para procesar muestras y generar informes, en los laboratorios de este departamento de Anatomía Patológica. Dichas funcionalidades forman parte del módulo de Anatomía Patológica del Sistema de Información Hospitalaria alas-HIS.

Para la realización del mismo se utilizaron herramientas definidas en la arquitectura establecida por el departamento de Sistema de Gestión Hospitalaria. La metodología RUP para el diseño del sistema, Java como lenguaje de programación, Eclipse Ganymede como entorno de desarrollo, PostgreSQL 8.3 como Sistema Gestor de Bases de Datos. Así como, el uso de Hibernate como herramienta ORM para la persistencia de los datos y el framework Seam para la lógica del negocio, entre otras tecnologías.

Con el sistema se prevén beneficios como: eliminar el procesamiento manual del flujo de información que se genera durante el proceso antes mencionado, reduciendo el tiempo que se ocupa actualmente durante el procesamiento de dichas muestras en un hospital. Estandarizar y agilizar la generación del informe final o diagnóstico definitivo, además disminuir los costos por concepto de inversión que tendría el país con la adquisición de un sistema de esta envergadura.

INDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

El funcionamiento de cualquier institución, depende de la eficiencia con que se utilicen sus recursos. La maquinaria, el dinero, las personas y la información forman parte de estos; pero sin un control de esta última, todos los demás quedarían aislados e inmanejables. En un hospital se manipulan notables volúmenes de información, por lo que su tratamiento, mejora considerablemente la labor se realiza con esta.

Durante la década del 70, surgen los primeros sistemas de información para la salud. Se introdujeron casi exclusivamente por necesidades financieras y de gestión económica de los centros hospitalarios. Los sistemas mencionados se centraban en almacenar datos demográficos del paciente y mezclarlos con datos de costos para producir facturas. Al evolucionar dieron lugar a sistemas médicos llamados PACS, RIS y Sistemas de Información Hospitalaria (HIS, siglas en inglés). En la década de los 90, los HIS se movieron en dirección a centrarse sobre el paciente y estar orientados clínicamente; estos sistemas no veían al paciente como una colección de números o episodios, sino como un flujo continuo de datos. Se conseguía más información desde el punto de vista asistencial y mejoraban la comunicación médico-paciente.

Los HIS, son sistemas de información orientados a satisfacer las necesidades de almacenamiento, procesamiento e interpretación de los datos médico-administrativos generados. Además, constituyen un apoyo para las actividades en los niveles operativos, estratégicos y tácticos de cualquier institución hospitalaria.

Este tipo de sistema permite también, la optimización de los recursos humanos y materiales, además minimiza los inconvenientes y morosidades que enfrentan los pacientes. A partir de un HIS se pueden obtener reportes e informes estadísticos, en dependencia del área o servicio que los requieran. Esto da lugar a la retroalimentación en el desempeño de la atención de salud y como consecuencia aumenta o mejora la calidad de vida de los pacientes y de los servicios de salud que se prestan.1

[1] Sistema de Información Hospitalaria. México D.F.: Universidad Autónoma de México. D. R. Facultad de Medicina,(2003). p 7. Obtenido el 03 11 ,2003, de: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.

Está conformado por módulos donde, cada uno de ellos representa un área específica dentro de un hospital. En cada uno de los módulos se recoge información, quedando esta almacenada y centralizada en una base de datos, permitiendo ser consultada desde otros módulos.

Dentro del conjunto de áreas presentes en un hospital se encuentra Anatomía Patológica cuyo funcionamiento actual consiste en el examen anatómico de un cadáver, conocido como autopsia y en el procesamiento histológico de las muestras obtenidas a través de las biopsias (muestras de tejidos) y las citologías (muestras de fluidos).

A partir de estas se realiza una interpretación diagnóstica definitiva que permite al médico indicarle al paciente el tratamiento a seguir. El procesamiento de dichas muestras describe el flujo de información que se genera a partir de que se recoge la muestra, se envía al laboratorio, se hacen los exámenes correspondientes, se da un diagnóstico definitivo y se generan los informes correspondientes.

Este proceso genera información de suma importancia, el cual tiene que estar en constante movimiento viajando con las muestras y actualizándose según lo exámenes que se le realicen a esta. Actualmente en un hospital esto se hace de forma manual, lo cual trae una serie de inconvenientes pues toda la información recogida es archivada para su posterior consulta, ya sea para estudio científico, docente o evaluación de la calidad de la labor médica realizada, tarea que se torna lenta por el volumen de información almacenada. Esta puede en ocasiones ser ilegible por el deterioro a raíz del tiempo y las condiciones del lugar en que se encuentra.

También existen varios inconvenientes en el momento de confeccionar los informes, quien se encarga de realizar este documento es la secretaria, luego se lo envía al patólogo, para que lo revise y firme, si está incorrecto este lo regresa hasta que esté bien confeccionado, y si está correcto la secretaria realiza dos copias una para archivar y otra para recepción. Lo que trae como consecuencia una demora innecesaria y resulta desfavorable para el solicitante que está en espera de un resultado.

Por lo antes expuesto se identifica como **problema a resolver:** ¿Cómo facilitar la gestión de información durante el procesamiento de las muestras y generación de informes en el área de Anatomía Patológica de las instituciones hospitalarias?

En correspondencia con el problema, el **objeto de estudio** lo constituyen los procesos 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 procesamiento de una muestra en los laboratorios del área de Anatomía Patológica de las instituciones hospitalarias.

El **objetivo de la investigación** es realizar la implementación de los procesos generar informe y procesar muestra 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 de la investigación**:

- Evaluar las tendencias actuales de los Sistemas de Información Hospitalaria en el ámbito internacional.
- Asimilar la arquitectura definida por el Departamento de Sistema de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
- Asimilar las pautas para el desarrollo de los entregables y el diseño (definidos por el mismo departamento).
- Analizar los patrones de diseño a utilizar.
- Obtener los artefactos correspondientes a los flujos de trabajo, “Diseño” e “Implementación” y “Pruebas”.
- Implementar: procesar muestra y generar informes, según el documento de casos de uso del sistema.
- Obtener el acta de liberación otorgada por calidad interna, de la documentación y de las funcionalidades implementadas.

El presente documento está constituido por cuatro capítulos, que incluyen todo lo relacionado a la investigación realizada, diseño e implementación de la aplicación propuesta, así como el modelo de prueba de la misma. Además, cuenta con introducción, conclusiones y anexos.

Capítulo 1 Fundamentación Teórica: En este capítulo se detallarán cada uno de los aspectos relacionados con el sistema a implementar, así como el análisis de soluciones existentes relacionadas con el mismo, en la nación e internacionalmente. Explica además un estudio detallado de las tecnologías actuales, lo que posibilitará decidir sobre qué herramientas utilizar para el diseño e implementación del sistema.

Capítulo 2 Descripción de la arquitectura: Se describe la arquitectura definida, 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 la modelación detallada y la construcción de la aplicación.

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

Capítulo 1: Fundamentación teórica

El presente capítulo tiene como objetivo fundamental mostrar los conceptos básicos relacionados con la gestión de la información en el área de Anatomía Patológica. Haciendo el análisis de los sistemas análogos que existen en la actualidad tanto en el mundo como en el ámbito nacional así como las tecnologías, herramientas y metodologías que se utilizan en el desarrollo de software.

1.1 Conceptos básicos relacionados con el dominio del problema

La Anatomía Patológica se encarga del estudio de los órganos y los tejidos con el fin de determinar las causas y efectos de ciertas enfermedades, es uno de los pilares fundamentales en una institución hospitalaria, de tal manera que todo hospital cuenta con esta. Los resultados que se arrojan durante las pruebas que allí se realizan son indispensables para facilitar la labor en cualquiera de las áreas aledañas dentro del hospital, logrando así un funcionamiento óptimo en el mismo.

Esta área se divide en dos procesos generales: realizar autopsia conocido también como el análisis anatómico de un cadáver y realizar análisis cuyo objetivo es el procesamiento histológico de las muestras que se obtienen a partir de las biopsias y citologías, este se realiza con el objetivo de conocer las patologías presentes en un paciente vivo.

Los análisis correspondientes son realizados a partir de una solicitud que llega a la recepción. La misma viene acompañada de una muestra, puede presentarse en forma de tejidos o fijadas en láminas. Luego el patólogo se encarga de emitir la descripción macroscópica, posteriormente se obtienen los casettes que son sometidos a un proceso de corte fino para obtener bloques que serán archivados, y láminas que serán examinadas. Este material es recibido por el histotecnólogo, el cual actualiza el registro de derivados con las cantidades de material recibido. A partir de esto se comienza el estudio microscópico de la muestra en sus nuevas formas.

El proceso concluye con el diagnóstico microscópico, que genera un informe final, el cual es transcrito por la secretaria que le envía al patólogo un borrador para su revisión. Si este informe no presenta problemas en su estructura, es firmado y entregado a la secretaria quien se encarga de archivar una copia y entregar otra en recepción. En caso contrario el informe es entregado nuevamente a la secretaria para su inmediata corrección.

Los informes son el producto final donde se plasma el resultado del estudio de las muestras por cualquiera de los procedimientos anteriormente mencionados, estos contienen la opinión médica experta basada en observaciones, integrada con los datos clínicos del momento.

Se procesa una muestra cuando un paciente o solicitante requieren de los servicios de Anatomía Patológica para conocer si padece o no una afección. Para esto se lleva cabo el proceso anteriormente mencionado donde pueden distinguirse varios tipos de biopsias:

- Insicional
- Exicional
- Por trocal
- Transoperatoria o por congelación

Esta última consiste en un examen que se le realiza a la muestra durante una operación en curso, donde el patólogo tiene que dar un diagnóstico en 20 minutos para detener la intervención quirúrgica o continuar con la misma. Lo que demuestra una medida de cuán importante es la realización de este tipo de análisis.

1.2 Sistemas automatizados existentes vinculados con el campo de acción

Un HIS es un conjunto organizado de elementos, los cuales interactúan entre sí para procesar información y distribuirlas de manera adecuada en función de los objetivos de una organización, en este caso, un hospital y específicamente en el área de Anatomía Patológica.

Pat-win

Sistema integrado a un HIS que funciona con una red de comunicaciones intrahospitalaria íntegra en el Hospital Son Llàtzer ubicado en Palma de Mallorca, España. Permite realizar una gestión completa de los departamentos anatomopatológicos. Proporciona principalmente funcionalidades para: 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. Este proyecto tiene como objetivo la incorporación de la imagen microscópica (biopsias y citologías) y microscópica en la historia clínica y en el circuito de almacenamiento de imágenes de la Fundación Hospital Son Llàtzer.

Este sistema tiene como principales 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.
- 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.
- Posibilidad de gestión y confección de todo tipo de estudios como: biopsias, citologías generales, ginecológicas.
- Módulo de gestión de imágenes.
- Corrector ortográfico integrado.
- Gran número de listados y estadísticas. 2

X-HIS

Sistema de Información Hospitalaria desarrollado por la empresa iSOFT, desarrollado en España, instalado en el Hospital Regional materno-infantil de Monterrey, México y otros lugares de Latinoamérica como Perú. 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 cómodo. También permite tras la identificación previa del

[2] IT Trade, SA.de CV. (2010). IT TRADE Soluciones en TI. Obtenido el 01 02, 2010, de http://www.ittrade.com.mx/pdf/PATwin_080108.pdf

paciente que el usuario del sistema pueda navegar por todo su historial clínico, a través de un navegador asistencial, sin necesidad de salir y entrar en diferentes aplicaciones.

Este 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, radiología, fisioterapia y uno de gran importancia para la labor de diagnóstico que se realiza dentro de los hospitales, Anatomía Patológica. Este 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.

Kewan HIS

También llamado Kewan-Cosmosalud, es un sistema que ha sido instalado en hospitales de España. 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 (HIS) y sistemas departamentales. Forma parte de un ERP 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. En general gestiona la información de varios departamentos como son, la Central de Esterilización, Morgue, Radiodiagnóstico Y Anatomía Patológica, este último con el objetivo de llevar el control de las solicitudes y resultados de diagnósticos. 3

[33] Infomed. (2007). Informatica en salud 2007. Obtenido el 01 03, 2010, de "redes y conocimientos en acción": http://www.informatica2007.sld.cu/Members/jherreroypf/la-experiencia-kewan-cosmosalud/

SARCAP

Realizado en Cuba en 1985, un Sistema Automatizado de Registro y Control de Anatomía Patológica, conocido por SARCAP. Está compuesto por 2 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.

La interacción del usuario con el sistema se realiza a través de un menú que posee las mismas opciones para los 2 subsistemas. Facilita el conocimiento de los diagnósticos realizados tanto clínicos como anatomopatológicos y en especial, las causas de muerte (directa, intermedia, básica 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 este, han permitido la realización de trabajos científicos con el aporte de sólida argumentación a diversas líneas de investigación entre las que se encuentran: infecciones, daño multiorgánico, edema pulmonar de permeabilidad, muerte por hechos violentos, cáncer.

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. Resulta un instrumento idóneo para la máxima utilización de los conocimientos que aportan los métodos de la Anatomía Patológica (autopsia y biopsia) en el control, evaluación y aseguramiento de la calidad del trabajo médico.[4]

[4] Infomed. (2010). Biblioteca virtual en salud. Obtenido el 01 03, 2010, de http://bvs.sld.cu/revistas/mil/vol24_2_95/mil10295.htm

AvanPat

Surge como una respuesta a la necesidad que el desarrollo creciente de la medicina cubana, en áreas de investigación como la anatomía patológica, demanda en cuanto a la fluidez de la información, asesoramiento, procesamiento e intercambio de datos, incluyendo imágenes. La aplicación de la tele patología y otras técnicas informáticas se convierten en una realidad potencial.

Es un sistema que almacena, organiza y gestiona la información relacionada con los estudios que se realizan en los departamentos de Anatomía Patológica, cuenta con un conjunto de plantillas prediseñadas para estudios (citologías de líquidos, vaginales y punción aspirativa con aguja fina, biopsia y necropsias) con más de 350 campos incluyendo imágenes y vídeo, permite al patólogo diseñar nuevos estudios o agregar campos a los prediseñados.

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.

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 anatomopatológicos final, entre otros.5

Tiene como objetivo crear un sistema de información para el trabajo administrativo e investigativo de los departamentos del área que se pueda obtener de forma libre con el propósito de facilitar el trabajo de los departamentos de la misma, unificar el formato de intercambio de información entre patólogos y establecer un adecuado flujo de información entre otros sectores de la salud.

[5] Infomed. (2004). IV Congreso virtual hispanoamericano de Anatomía Patológica . Obtenido el 01 04, 2010, de http://conganat.uninet.edu/6CVHAP/autores/trabajos/T067/index.html

Sistemas para hospitales				
Nombre del HIS	**País**	**AP**	**Herramientas para el desarrollo (Libres)**	**Propietario**
Pat-win	**España**	**X**	**---**	**X**
x-His	**España**	**X**	**---**	**X**
Kewan HIS	**España**	**X**	**---**	**X**
SARCAP	**Cuba**	**X**	**X**	**---**
AvanPat	**Cuba**	**X**	**X**	**---**

Tabla 1.1 Sistemas que gestionan los procesos de biopsias y citologías de AP

1.3 Tecnologías actuales a considerar

Teniendo en cuenta las características del producto que se va desarrollar y el estado de arte de los Sistemas de Información Hospitalarias 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.

Metodología de desarrollo

RUP

El Proceso Unificado de Software, es un proceso genérico que puede ser utilizado para varios tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos. Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. Se basa en componentes, lo que significa que el sistema en construcción estará integrado por componentes de software interconectados por medio de interfaces bien definidas.

Esta metodología identifica nueve flujos de trabajo, que van desarrollándose durante las fases de inicio, elaboración, construcción y transición; seis de ingeniería entre los que se encuentran: modelado de negocio, requerimientos, diseño e implementación, prueba, despliegue y tres de apoyo que son: configuración y cambios, entorno y gestión de proyecto. Durante el flujo de diseño, se crea y valida la arquitectura. En la implementación se define la organización del código, se implementan los objetos en forma de componentes, se prueban los componentes desarrollados y se integran los componentes en un sistema ejecutable. Para esto se generan artefactos, con el objetivo de guiar el desarrollo del producto, para los cuales RUP utiliza el Lenguaje de Modelado Unificado (UML) en la preparación de todos los planos del sistema. De hecho, UML es una parte integral del proceso unificado. Los aspectos distintivos de este están capturados en tres conceptos clave: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental.

Características

Proceso dirigido por casos de usos: Son técnicas de captura de requisitos que parten de la importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Los casos de usos representan los requisitos funcionales del sistema. En RUP estos no son sólo una herramienta para especificar los requisitos del sistema, también guían su diseño, implementación y prueba, por lo que constituyen un elemento integrador y una guía del trabajo, además se proporcionan un hilo conductor, permitiendo establecer la trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

Proceso centrado en la arquitectura: Es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. En la arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, es relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además, la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La misma se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados entre otras.

Proceso iterativo e incremental: Es donde el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo que el equilibrio entre casos de uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. Cuenta con una secuencia de iteraciones en la que cada una aborda una parte de la funcionalidad total, pasando por cada uno de los flujos de trabajo relevante y refinando la arquitectura. 6

Lenguajes

UML

Es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos .Un artefacto es una información que es utilizada o que es producida mediante un proceso de desarrollo de software, Unified Modeling Language o UML como también es conocido, fue desarrollado por Grady Booch y Jim Rumbaugh en 1995, con el objetivo de crear un lenguaje para el modelado y que permitiera ser usado tanto por máquinas como por personas, además, establecer un acoplamiento explícito de los conceptos y los artefactos ejecutables.

Cuenta con una serie de características que lo consolidan como un lenguaje estándar en el análisis y diseño de sistemas de computo como por ejemplo, los mecanismos de extensibilidad como estereotipos, valores etiquetados y restricciones, patrones y colaboraciones, diagramas de actividades para reingeniería de procesos de negocio, refinamiento para manejar relaciones entre niveles de abstracción, clara separación de tipos, clases e instancias, además de contar con una serie de interfaces y componentes que lo hacen muy útil dentro de la comunidad de desarrollo.

Ventajas

Con respecto a otros lenguajes, facilita un mayor rigor en las especificaciones, por lo que se hace más fácil entender lo que se quiere, permite realizar verificaciones y validaciones a los diferentes modelos que

[6] Ivanex. (2008). Sistema de selección de las rutas optimas . Obtenido el 01 05, 2010, de http://ivanex.wikidot.com/metodologia

se realizan, además de que se pueden automatizar determinados procesos y generar códigos a partir de los diferentes modelos o viceversa, por lo que proporciona que se tengan versiones tanto de los modelos como del código actualizadas, con lo que se puede mantener la visión en el diseño de más alto nivel, de la arquitectura de un proyecto.

Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo 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 o memoria.

Características:

Las características principales que ofrece Java respecto a cualquier otro lenguaje de programación, son:

Simple: Ofrece toda la funcionalidad de un lenguaje potente, se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje, mediante el reciclador se permite liberar bloques de (Java)de los lenguajes C y C++ como es el caso de: aritméticas de punteros, registros, definiciones de tipo, macro entre otras.

Orientado a objetos: Trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

Distribuido: Construido con extensas capacidades de interconexión TCP/IP, proporciona librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Robusto: Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución, obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error, maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria, proporciona comprobación de punteros, de límites de array, excepciones, verificación byte-code, entre otros.

Arquitectura neutral: Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará.

Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en de ningún modo la máquina en que ha sido generado.

Multithreaded: Consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real, aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente, aún supera a los entornos de flujo único de programa tanto en facilidad de desarrollo como en rendimiento. 7

Ventajas:

Reduce en un 50% los errores más comunes de programación, reducción del tiempo de desarrollo de aplicaciones, es multiplataforma, las aplicaciones en Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus, es interpretado y dinámico, entre otras cosas.

Herramientas

Entorno de desarrollo: Eclipse

Es una plataforma de desarrollo basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. Es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados: plugin. Contiene una serie de perspectivas. Cada perspectiva proporciona una serie de funcionalidades para el desarrollo de un tipo específico de tarea. Por ejemplo la perspectiva Java combina un conjunto de vistas que permiten ver información útil cuando se está escribiendo código fuente, mientras que la perspectiva de depuración contiene vistas que muestran información útil para la depuración de los programas Java. Es uno de los proyectos de código abierto más interesantes y más usados para el desarrollo de aplicaciones, entre sus versiones se encuentra el Eclipse Ganymede, el cual es la versión consecutiva del Europa que ostenta entre sus características fundamentales:

[7] Ministerio del poder popular para ciencia, tecnología e industrias intermedias. (2010). Biblioteca digital. Obtenido el 01 05, 2010, de http://www.biblioteca-digital.net.ve/wordpress/?p=507

- Búsqueda de texto mejorada.
- Mejoras en el reemplazo de texto.
- Cerrar tabs con el botón del medio o con el scroll del mouse.
- Soporte del compilador Java para múltiples CPU.
- Mejoras en el debugger.
- Asistente para la conversión a StringBuffer.

Plataforma: Java EE

Java Plataform, Enterprise Edition o Java EE 6 es el estándar de la industria para la informática empresarial de Java, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de niveles distribuidos. [8]

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 UI y el mínimo y solamente necesario XML. Se integra con librerías de controles de código abierto basadas en JSF como RichFaces e ICEFaces.

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 también el concepto de workspaces permitiendo que el usuario tenga en varias 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

[8] Oracle corporation y sus filiales. (2010). Oracle. Obtenido el 01 05, 2010, de http://java.sun.com/javaee/

(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.

Servidor de aplicaciones: JBoss AS

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.

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.
- **Drools IDE:** Editor de ficheros de reglas, debugueo e inspección de reglas.
- **JBossWS Tools:** Desarrollo, invocación, inspección y pruebas de webservices sobre http con la adición y soporte de características JBossWS.

JSF

JavaServer Faces o sus siglas en ingles 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.

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 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).

Librería: Rich Faces 3.2

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

Librería: Ajax4jsf

Ajax4jsf es una librería 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 automáticas al servidor, 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.

Herramienta cliente para la administración de bases 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. [9]

Herramienta para la administración de bases de datos: EMS

EMS SQL Management Studio es una solución completa para la administración de bases de datos y el desarrollo. Es una herramienta potente y fácil que integra componentes para la gestión de bases de datos.

JRE

JRE es el acrónimo de Java Runtime Environment 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 las 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 incluye, entre otros, un compilador para Java. [10]

Framework Hibernate para acceso a los datos

Hibernate es una herramienta de mapeo objeto-relacional para la plataforma Java que permite desarrollar clases persistentes basándose lenguaje orientado a objetos - incluyendo la asociación, herencia,

[9] Ubuntu. (2010). Guia documentada para ubuntu. Obtenido el 01 07, 2010, de http://www.guia-ubuntu.org/index.php?title=PgAdmin_III

[10] Java Utilidades y programación. (2007). Obtenido el 01 07, 2010, de http://www.tipete.com/userpost/descargas-gratis/megapost-java-utilidades-y-programaci%C3%B3n

polimorfismo, la composición y las colecciones. Expresa consultas en su propia extensión de SQL portátiles (HQL), así como en SQL nativo, o con criterios orientados a objeto.

Herramienta para el modelado: Visual Paradigm

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

Características:

Es multiplataforma, permite realizar cualquier tipo de diagrama entiéndase diagrama de estado, de actividades, de despliegue, de actores del negocio y sistema, de flujo de datos, entre otros, tiene soporte para el lenguaje de modelado UML, permite importar y exportar ficheros XML, además de ser robusto y tener gran usabilidad y fiabilidad.

Ventajas:

Ingeniería de ida y vuelta, ingeniería inversa Java, generación de código, modelo a código, diagrama a código, diagramas EJB, visualización de sistemas EJB, diagramas de flujo de datos, modelado colaborativo con CVS y subversión.

Con el estudio realizado a los sistemas existentes se llega a la conclusión de que algunos presentan inconvenientes como:

- Son software propietario, por lo que su código fuente no es libre y no se le pueden realizar modificaciones.
- Poseen un costo alto en el mercado; para disponer de ellos hay que pagar licencia, actualizaciones y soporte técnico lo que necesariamente provoca que no sea factible su adquisición.
- Son sistemas que han sido desarrollados para satisfacer las especificaciones del lugar donde han sido desplegados.

- En general se encargan los procesos del área pero no específicamente del procesamiento unísono de las muestras de biopsias y citologías.
- Presentan déficit en la generación de informes.

Esto hace inevitable desarrollar un software flexible, que no sea costoso pues deberá ser desplegado en una amplia red de instituciones y con el objetivo de obtener una aplicación de coste mínimo, se definieron las siguientes herramientas que conforman el ambiente de desarrollo para el módulo en cuestión:

- Gestor de base datos: PostgreSQL 8.3
- Lenguaje de programación: Java
- Metodología de Desarrollo de SW: RUP
- Herramienta de Modelado: Visual Paradigm
- IDE Desarrollo: Eclipse

Capítulo 2: Descripción de la arquitectura

En el presente capítulo se describen los requisitos no funcionales, se describe la arquitectura propuesta para el desarrollo así como la vista de despliegue y como será tratado el tema de la seguridad en el sistema que está en desarrollo .Se exponen las estrategias de codificación donde se estipulan los estándares y estilos a utilizar. Partiendo del resultado del análisis de posibles implementaciones de componentes o módulos ya existentes que pueden ser rehusados se propone la estrategia de integración.

2.1 Descripción de requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Existen múltiples categorías para clasificar los requisitos no funcionales, siendo las siguientes las presentes en el Sistema de Información Hospitalaria alas HIS.

Usabilidad

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

Usuarios normales: 20 días

Usuarios avanzados: 30 días

Fiabilidad

- En los servidores de los hospitales 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.

- 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 entre usuarios, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo con 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 entre las 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 base de batos, independientemente de que para el sistema, este elemento ya no exista.
- El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

Eficiencia

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

Seguridad

- Se mantendrá seguridad y control entre usuarios, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo con 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 entre las 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 registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.
- 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 base de datos, 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.

Rendimiento

El sistema minimizará el volumen de datos en las peticiones y optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda. Respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos. Se deberá usar siempre que sea posible el patrón Singleton destruir referencias que ya no estén siendo usadas, optimizar el trabajo con cadenas, entre otras buenas prácticas que ayudan a mejorar el rendimiento.

Soporte

Seguridad de acceso y administración de usuarios

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos.

Monitoreo de funcionamiento

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

Respaldo, recuperación de base de datos y réplica

Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo y recuperar la base de datos a partir de los respaldos realizados.

Auditoría

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

Configuración de parámetros

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

Restricciones de diseño

- La capa de presentación contendrá todas las vistas y la lógica de la presentación. El flujo web se manejará de forma declarativa y basándose en definiciones de procesos del negocio.

- La capa del negocio mantendrá el estado de las conversaciones y procesos del negocio que concurrentemente pueden estar siendo ejecutados por cada usuario.
- La capa de acceso a datos contendrá las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

Requisitos de 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 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 de comunicación

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

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

Se posibilitará el uso de ayudas dinámicas y tutoriales en línea sobre el funcionamiento del sistema.

Requerimientos de hardware

Estaciones de trabajo

En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS, las que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, se recomienda IE 7, Firefox 2 o versiones superiores. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 GHz 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 disc 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.

Requerimientos de software

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

El sistema deberá disponer de un navegador web, estos pueden ser IE 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

Portabilidad

El producto podrá ser utilizado bajo los sistemas operativos Linux o Windows

2.2 Descripción de la arquitectura

Una de las tareas más importantes en el desarrollo de cualquier sistema de información 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 basándose en los 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.

Una de las tareas de la arquitectura de software también es la elección de patrones a utilizar para el desarrollo de software, los patrones no son más que una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos.

También expresan una organización estructural para un sistema de software, proveen un conjunto de subsistemas predefinidos e incluyen reglas y lineamientos para conectarlos.

La arquitectura definida para el desarrollo del sistema está 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). Representado por las clases autogeneradas, personalizadas y componentes Hibernate que permite las operaciones con datos de una base de datos aprovechando las ventajas de la orientación a objetos.

Vista: Maneja la visualización de la información. Compuesta por páginas XHTML, las cuales contienen formularios que mediante controles JSF y RichFaces que recogen, validan y muestran los datos que el usuario provee y solicita en cada una de las operaciones que realiza.

Controlador: Controla el flujo entre la vista y el modelo (los datos). Integrado por las clases controladoras (beans) que encierran la lógica del negocio del módulo, permiten el manejo de las acciones que el usuario realiza sobre la vista modificando los datos y las entidades en el modelo para ser mostrada nuevamente al usuario. Se utiliza Seam como Framework de integración que une las capas, modelo y vista.

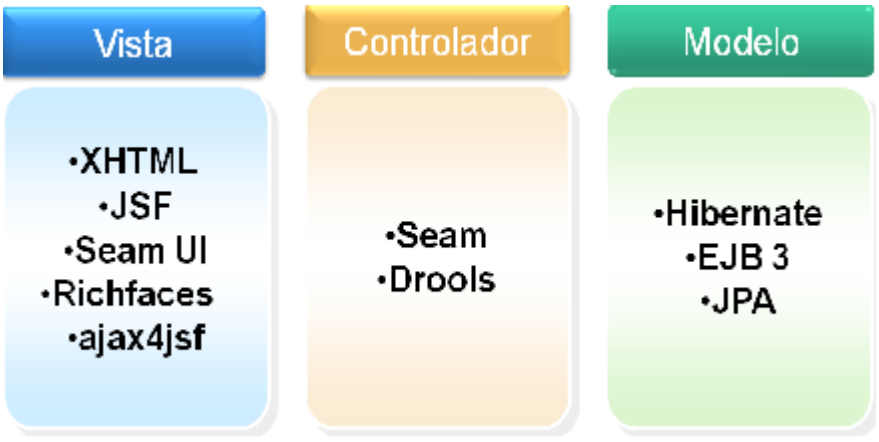

Figura 2.1 Visión general de la arquitectura

La elección de este modelo proporciona que múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes, ya que la vista se halla separada del modelo y no exista dependencia directa del modelo con respecto a la vista.

Además, permite cambiar con mayor rapidez los requerimientos de interfaz de usuario dado que el modelo no depende de las vistas agregar nuevas opciones de presentación no afecta al modelo.

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

La reutilización de código no es más que una técnica o mecanismo de programación que garantiza que una parte o totalidad de una aplicación existente pueda emplearse en la construcción de otra. De esta forma, se elimina la repetición de código, economiza el tiempo y el costo de desarrollo.

El área de AP es un servicio indispensable un hospital pues contribuye a un estudio más completo de las enfermedades, sin la misma, las demás áreas no podrían funcionar de manera óptima pues las propias se nutren de los resultados que esta les ofrece. Áreas como Bloque quirúrgico, Hospitalización y Consulta externa realizan solicitudes de biopsias y citologías, así como Emergencias y Hospitalización realizan solicitudes de autopsias; solicitudes que también pueden ser realizadas por alguna institución externa.

El módulo cuenta con dos procesos fundamentales para exponer los resultados de análisis correspondientes que son solicitados desde otras áreas: realizar análisis y realizar autopsia. Dentro de los subprocesos que conforman el primero de estos se encuentra: procesar muestras, el cual es reutilizado con las muestras de autopsias donde se obtienen las láminas, material para emitir un diagnóstico final en el proceso de realizar autopsias.

De forma general en el módulo se reutilizan las funcionalidades que brinda la clase Bitácora del módulo de Configuración para llevar a cabo el registro de las acciones que realiza el usuario: actualizar, modificar, ver, liberar, aceptar y crear.

Para la generación de informes se empleará la clase ReportManager del módulo Común que permitirá mostrar información en varios formatos (pdf, html, rtf) dentro de una página XHTML.

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, desde la autorización a directorios, páginas, controles, opciones del menú, servicios del negocio, está basada en reglas. Ninguna

de estas reglas del negocio 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 con 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 ids o llaves enviadas por 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).

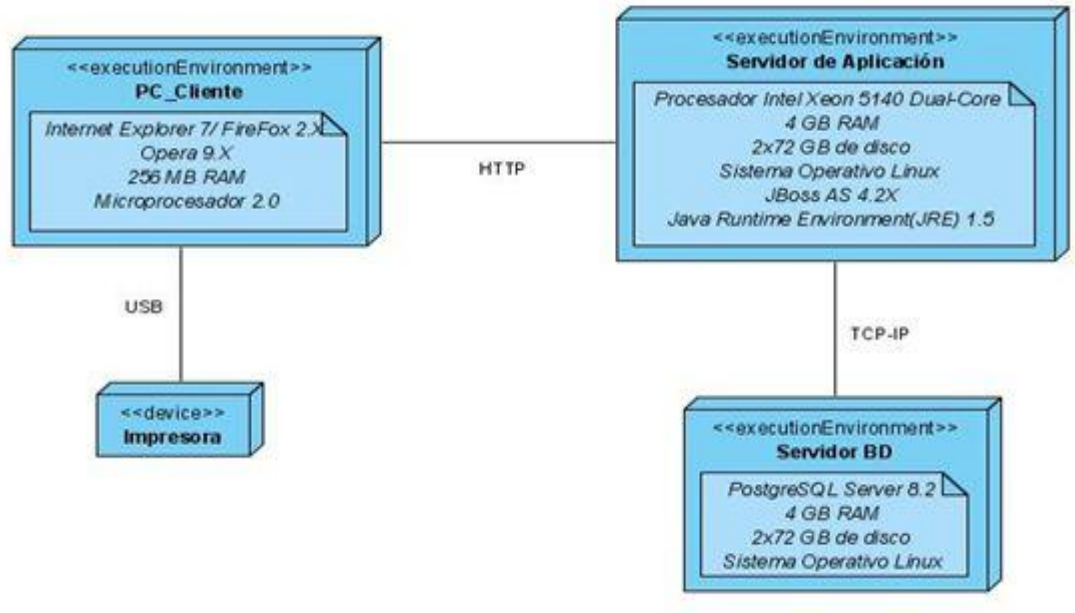

Figura 2.2 Vista de Despliegue.

2.6 Estrategias de codificación. Estándares y estilos a utilizar

El establecimiento de un estándar de programación trae consigo que todos los desarrolladores den soluciones de implementación con un estilo homogéneo, como si el código fuese escrito por un único programador. El mantenimiento de un producto software en muchos casos no es llevado a cabo por el autor original de este; una estrategia de codificación hace el software más legible, permitiendo a los desarrolladores entenderlo a fondo y más rápido. Para el desarrollo del sistema se utilizara el estándar de codificación definido en las especificaciones del lenguaje Java: Java Code Conventions.

```
/**
 * This is about <code>ClassName</code>.
 * {@link com.yourCompany.aPackage.Interface}
 * @author author
 * @deprecated use <code>OtherClass</code>
 */
public class ClassName<E> implements
Interface<String> {
        enum Color { RED, GREEN, BLUE };
        /* This comment may span multiple
lines. */
        static Object staticField;
        // This comment may span only this line
        private E field;
        // TASK: refactor
        @SuppressWarnings(value="all")
        public int foo(Integer parameter) {
                abstractMethod();
                int local= 42*hashCode();
                staticMethod();
                return bar(local) + parameter;
        }
}
```

Figura 2.3 Ejemplo del IDE de desarrollo.

El idioma a utilizar será el español y las palabras no se acentuarán. A continuación se muestran los estándares de codificación a utilizar para el desarrollo del sistema:

Notación Pascal: Se empleará para las clases donde que la palabra de inicio del identificador comienza con mayúscula. Si el identificador está compuesto por más de una palabra entonces estás deben comenzar con mayúsculas.

Ejemplo: **RegistroDerivadosArchivo**

Notación Camello: Se emplea para denotar variables, parámetros y métodos. Especifica que la palabra de inicio del identificador comienza con minúscula. Si el identificador está compuesto por más de una palabra entonces estás deben comenzar con mayúsculas, excepto la primera.

Ejemplo: **cantidadLaminasExistencia**

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

Comentarios, separadores, líneas, espacios en blanco y márgenes

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 y clases.

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.

Bases de datos, tablas, esquemas y campos

Apariencia de las tablas

Los nombres de las Bases de datos deben escribirse en minúscula, en caso de que sea un nombre compuesto se separará por underscoard. Se utilizará abreviatura en caso de que sea necesario.

Ejemplo: diagnostico_anatomapat_definitivo.

Tablas que representen relaciones

El nombre a emplear para estas tablas de relación debe comenzar con el nombre de la primera tabla, luego uderscoard, seguido de ‘in’ continuado de uderscoard y por último el nombre de la segunda tabla.

Ejemplo: cambio_celular_in_diag_micro_cx_gin

Tablas que representen nomencladores

Ver: Apariencia de las tablas.

Apariencia de los campos

Todas las letras en minúscula.

Nombre de los campos

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

Ejemplo:

id_muestra

En este capítulo se ha descrito la arquitectura del sistema donde se explica brevemente como será utilizado el patrón Modelo Vista Controlador, además los requisitos no funcionales del sistema a desarrollar. Se analizaron los aspectos de seguridad. La Vista de Despliegue muestra la distribución física del sistema y sus conexiones. Se establecieron estándares de codificación y estilos a utilizar para darle al código una mejor comprensión y mantenimiento. Para agilizar el desarrollo y eliminación de código redundante se analizaron los componentes a reutilizar. Y por último el perfil de integración del Área Temática de Hospitales.

Capítulo 3: Descripción y análisis de la solución propuesta

Este capítulo tiene como objetivo principal analizar el diseño propuesto por el analista y a partir de esto hacer una valoración sobre el mismo, obtener los diagramas de clases del diseño e interacción, hacer la descripción de las nuevas clases u operaciones necesarias, obtener el modelo de datos, hacer una breve valoración de las técnicas de validación de una base de datos, la descripción de las tablas que se utilizarán por el módulo y la vista de implementación.

3.1 Valoración crítica del diseño propuesto por el analista

En la ingeniería de software el diseño es uno de los flujos más importantes, en él se desarrollan, revisan y documentan los refinamientos progresivos de la estructura de datos, arquitectura, interfaces y datos procedimentales de los componentes del software. Para esto se apoya en el modelo de diseño.

Este es una abstracción del modelo de implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución, puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.11

En el diseño elaborado se plantea la utilización de patrones de diseño lo que constituye una buena práctica de programación porque pueden ser rehusables en diferentes problemas, en distintas circunstancias por lo que ahorran tiempo y suponen mejoras en el software.

Seam asegura la persistencia de datos mediante la abstracción que brinda Hibernate y gestor de bases de datos a utilizar. El componente EntityManagerFactory es uno de los principales en a arquitectura que se encarga de crear los objetos EntityManager e implementa el patrón de diseño Abstract Factory para permitir el acceso a la base de datos. Además, el EntityManager como interfaz principal del JPA es el responsable de realizar las llamadas operaciones del CRUD. Por otra parte, se utiliza también el patrón ActiveRecord por parte de Hibernate donde una fila en la tabla de la base de datos se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado.

[11] Ministerio del Poder Popular para Ciencia, t. e. (2010). MeRinde. Obtenido el 03 09, 2010, de http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=92&Itemid=296.

Hibernate implementa también patrones de mapeo y comportamiento, entre los primeros se encuentran: Identity field que guarda un campo de identificación de la base de datos en un objeto para mantener la identidad entre un objeto en memoria y una fila de una tabla en una base de datos, Foreign Key Mapping que mantiene una asociación entre objetos para referencias de llave foránea en tablas de la base de datos, o lo que es lo mismo mapear relaciones de uno a mucho, Association Table Mapping que guarda una asociación entre objetos como una tabla con las llaves foráneas a las tablas que esos objetos están asociados. O lo que es lo mismo mapear relaciones de mucho a mucho.

Entre los de comportamiento se usan: Identity map el cual mantiene un registro de todos los objetos que fueron cargados de la base de datos dentro de una transacción de negocio. Lo que evita tener en memoria dos representaciones distintas del mismo objeto y el Lazy que interrumpe por el momento el proceso de carga de objetos relacionados dejándolo listo, para cuando sus datos sean requeridos, él pueda cargarse automáticamente. También se maneja el uso del patrón Query object que interpreta la estructura de objetos y los traduce a consultas SQL.

Los patrones anteriormente mencionados son utilizados durante la etapa de implementación para asegurar la reutilización de código y el correcto manejo de las instancias. Garantizando así que la misma se optimice el trabajo y de resultados satisfactorios para el desarrollador.

Como patrones de diseño se utilizaron patrones GRASP, describen los principios fundamentales de diseño de objetos, plantea que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo y basándose en lo que propone este patrón, a cada clase se le asignaron las tareas que podían realizar según la información que poseían, además se crearon instancias de otras clases en correspondencia con la responsabilidad dada, poniéndose de manifiesto los patrones experto y creador para obtener un diseño de mayor cohesión y el encapsulamiento de la información.

El diseño propuesto cumple con los patrones de bajo acoplamiento y alta cohesión que se encargan respectivamente de que existan pocas dependencias entre clases y de que cada elemento del diseño realice una única labor dentro del sistema no desempeñada por el resto de los elementos.

Entre los elementos que componen el modelo de diseño están contenidos los diagramas de clases de diseño y los diagramas de interacción los primeros describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación y los segundos muestran un patrón de interacción

entre objetos. Describen cómo grupos de objetos colaboran para conseguir algún fin, mostrando a dichos objetos así como los mensajes entre ellos dentro del caso de uso.

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 su paquete correspondiente, haciendo uso de las clases que se encuentran dentro del paquete del repositorio que a su vez contiene dos subpaquetes, uno para las sesiones y otro para las entidades. El primero está formado por las clases controladoras del proceso y por aquellas que han sido autogeneradas o personalizadas mientras que el paquete de las entidades está compuesto por las clases autogeneradas y personalizadas.

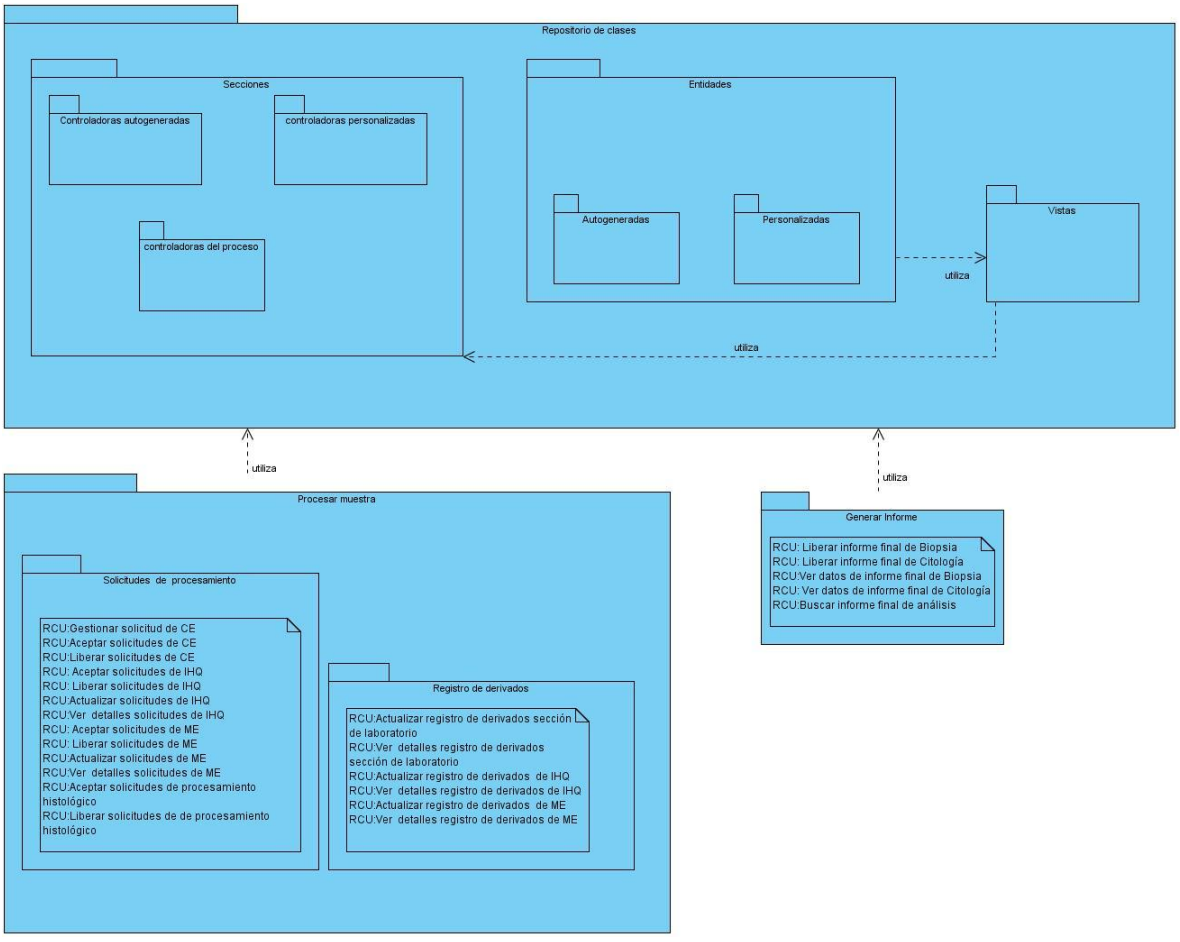

Figura 3.1 Diagrama de paquetes

A continuación se describen las realizaciones de los casos de uso conteniendo cada una, el diagrama de clases de diseño y los distintos diagramas de secuencia correspondientes:

Diagramas de clases del diseño

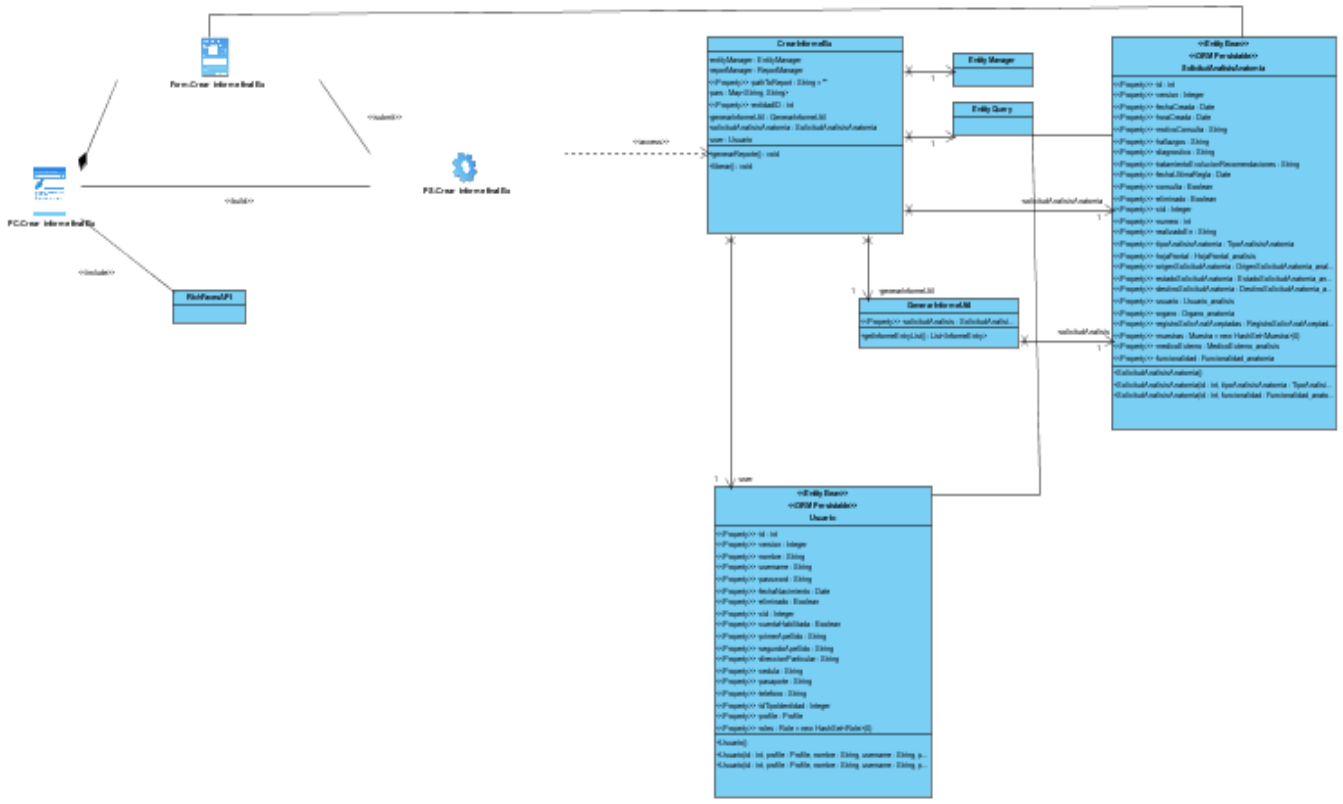

Figura 3.2 DCD_Crear informe final de Biopsia

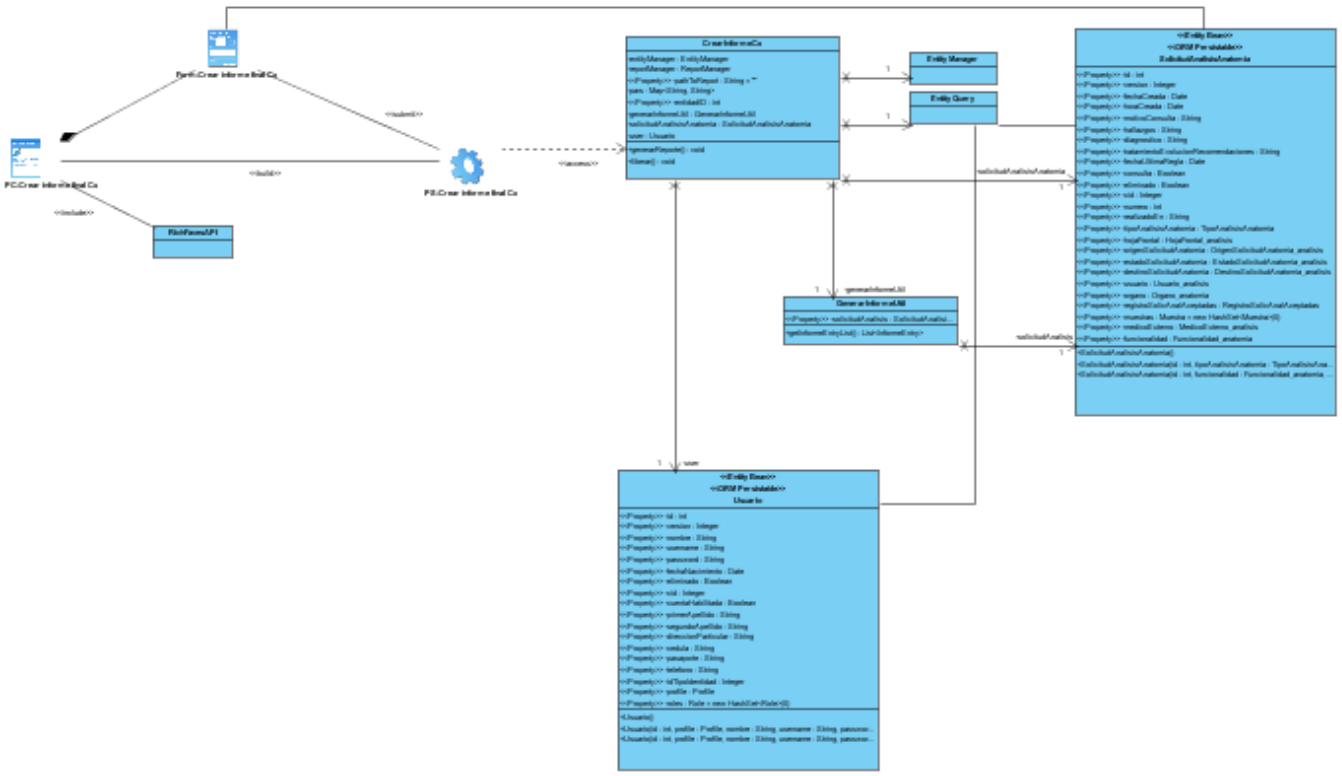

Figura 3.3 DCD_Crear informe final de Citología

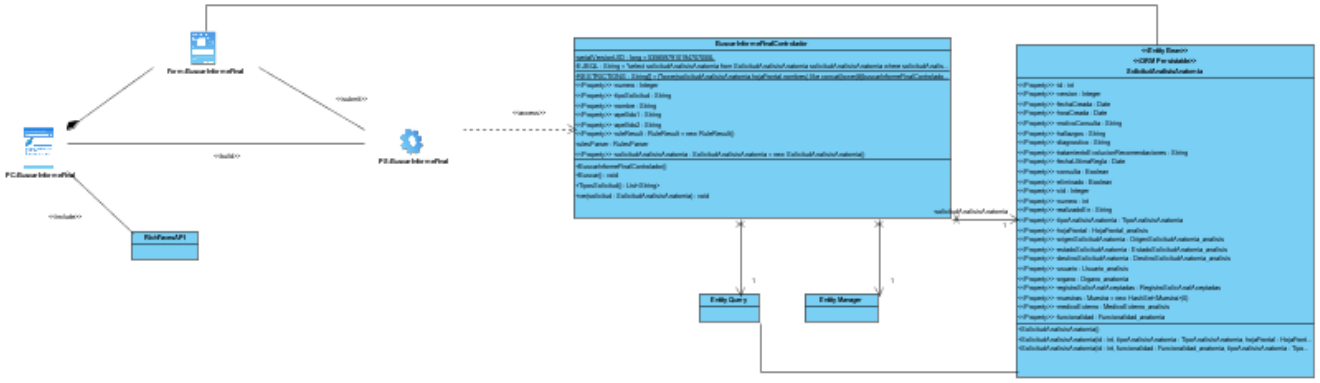

Figura 3.4 DCD_Buscar informe final de análisis

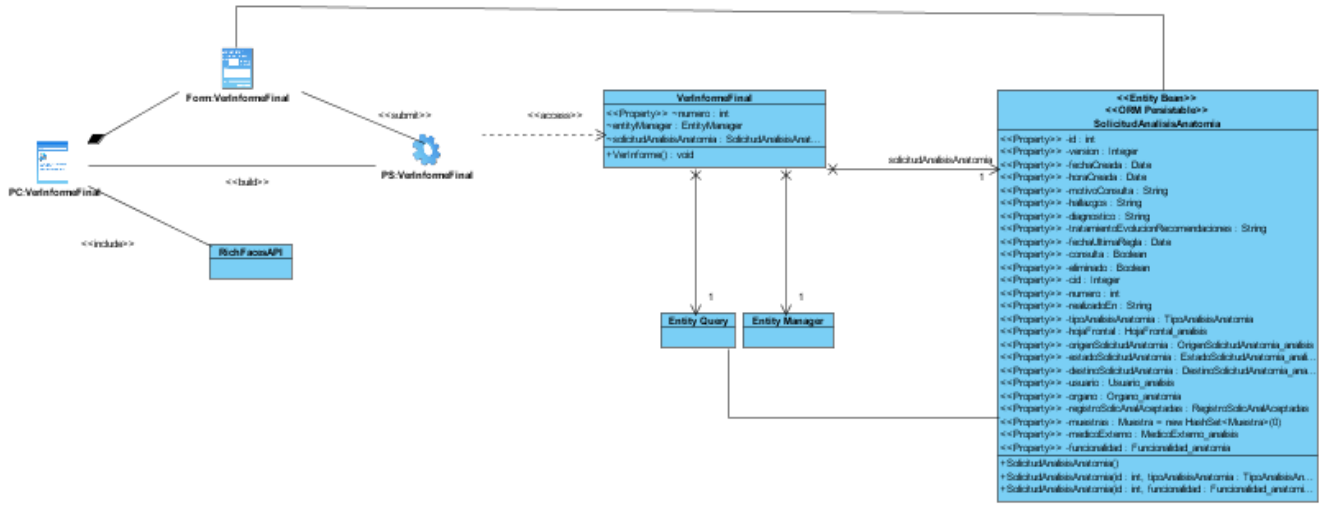

Figura 3.5 DCD_ Ver datos de informe final

Diagramas de secuencia

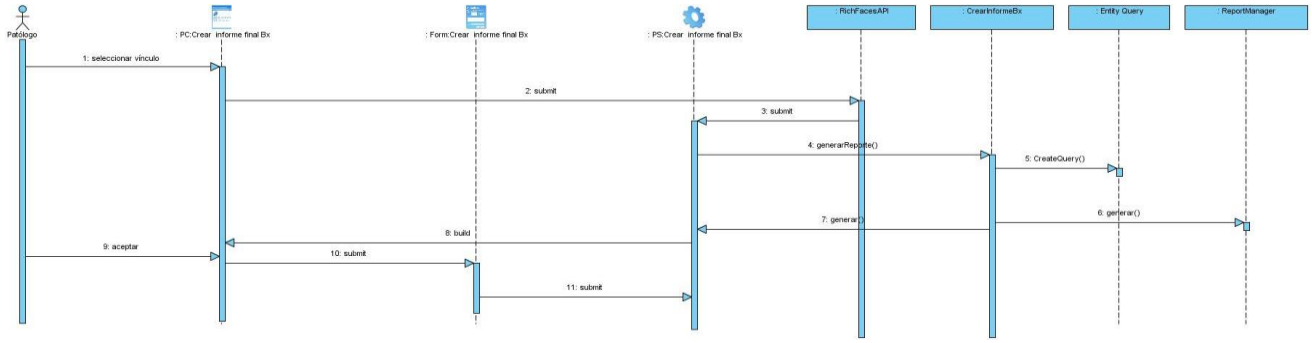

Figura 3.6 DS_Crear informe final de Biopsia

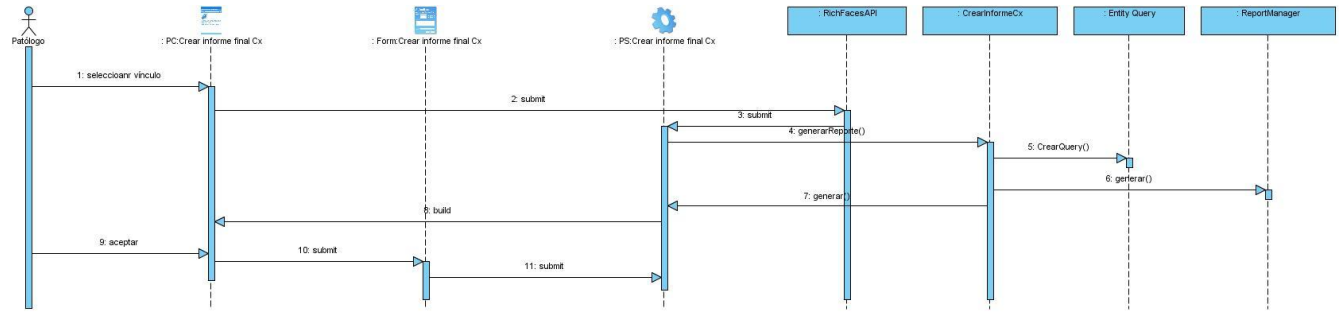

Figura 3.7 DS_Crear informe final de Citología

2

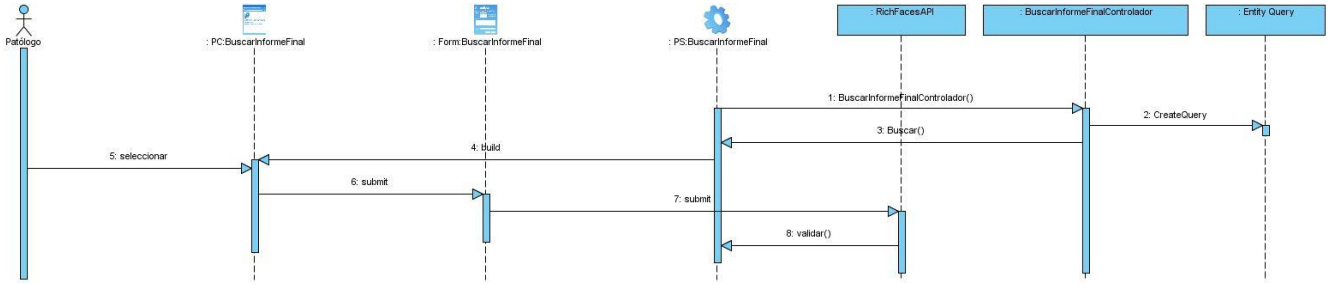

Figura 3.8 DS_Buscar informe final de análisis

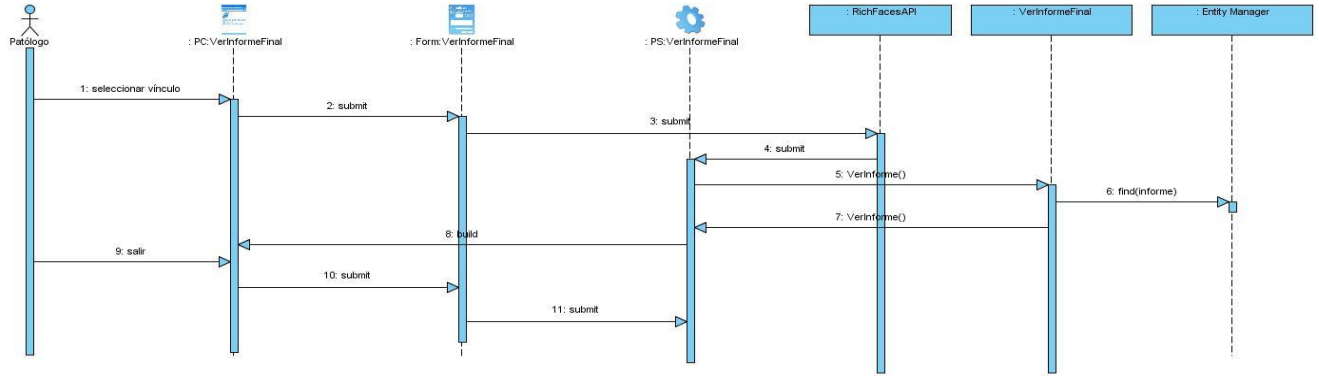

Figura 3.9 DS_Ver datos de informe final

3.2 Descripción de las nuevas clases u operaciones necesarias

Nombre: BuscarInformeFinalControlador	
Tipo de clase controladora	
Atributo	**Tipo**
serialVersionUID	long
EJBQL	String
RESTRICTIONS	String[]
solicitudAnalisisAnatomia	SolicitudAnalisisAnatomia
numero	Integer

tipoSolicitud	String
nombre	String
apellido1	String
apellido2	String
ruleResult	RuleResult
rulesParser	RulesParser
Para cada responsabilidad:	
Nombre:	BuscarInformeFinalControlador()
Descripción:	Contructor de la clase
Nombre:	Buscar()
Descripción:	Busca el informe final
Nombre:	ver(SolicitudAnalisisAnatomia solicitud)
Descripción:	Permite ver el informe final

Tabla 3.1 Buscar informe final controlador

Nombre: CrearInformeBx	
Tipo de clase controladora	
Atributo	**Tipo**
user	Usuario
generarInformeUtil	GenerarInformeUtil
entityManager	EntityManager
reportManager	ReportManager
pathToReport	String
pars	Map<String, String>

entidadID	int
solicitudAnalisisAnatomia	SolicitudAnalisisAnatomia
Para cada responsabilidad:	
Nombre:	generarReporte()
Descripción:	Genera el reporte
Nombre:	liberar()
Descripción:	Libera el reporte generado

Tabla 3.2 Crear informe biopsia

Nombre: CrearInformeCx	
Tipo de clase controladora	
Atributo	**Tipo**
user	Usuario
generarInformeUtil	GenerarInformeUtil
entityManager	EntityManager
reportManager	ReportManager
pathToReport	String
pars	Map<String, String>
entidadID	int
solicitudAnalisisAnatomia	SolicitudAnalisisAnatomia
Para cada responsabilidad:	
Nombre:	generarReporte()
Descripción:	Genera el reporte
Nombre:	liberar()

Descripción:	Libera el reporte generado

Tabla 3.3 Crear informe citología

Nombre: VerInformeFinal	
Tipo de clase controladora	
Atributo	**Tipo**
numero	int
solicitudAnalisisAnatomia	SolicitudAnalisisAnatomia
entityManager	EntityManager
Para cada responsabilidad:	
Nombre:	VerInforme()
Descripción:	Permite ver el informe final

Tabla 3.4 Ver informe final

3.3 Modelo de datos

Una de las características fundamentales de los sistemas de bases de datos, es que proporcionan cierto nivel de abstracción de datos al ocultar las características sobre el almacenamiento físico, que la mayoría de usuarios no necesita conocer.

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Contienen también un conjunto de operaciones básicas para la realización de consultas y actualizaciones de datos. [12]

[12] Marqués Andrés, M. M. (2001). Sistemas de Bases de Datos. Obtenido el 06 de 03 de 2010, de http://www3.uji.es/~mmarques/f47/apun/node32.html.

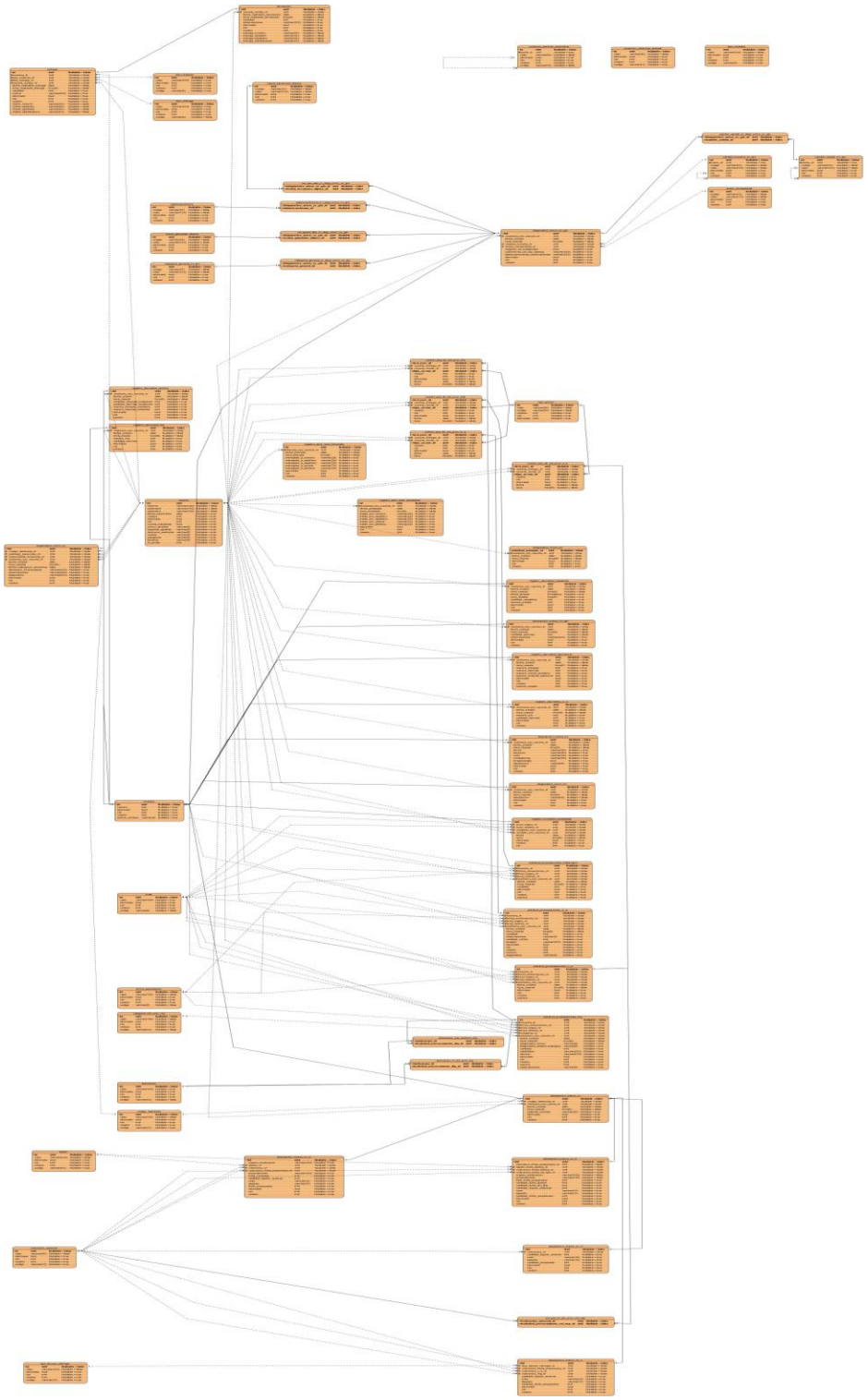

Figura 3.10 Modelo de datos

3.4 Valoración de las técnicas de validación

Para realizar las validaciones de entradas de usuarios se propone la utilización del Frameworks como JSF que proporcionan ayuda para definir las validaciones en la vista lo que tiene como inconveniente que repite las validaciones una y otra vez, pero Hibernate Validator propone resolver este problema de forma tal, define vez las validaciones en nuestros objetos de negocio e invoca esas validaciones desde el punto que nos interese.

Una de las funciones más importantes de una base de datos relacional es preservar la integridad de sus datos, esta garantiza la calidad de los datos que se almacenan. Cuando estos son modificados con las sentencias insert, delete o update la integridad de los datos puede perderse, porque pueden añadirse datos no validos o pueden modificarse datos existentes.

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de pautas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Este proceso sigue una serie de reglas para cada fase, para evitar la redundancia de datos, problemas de actualización de los datos en las tablas y proteger la integridad de los mismos dentro de la base de datos.

Descripción de las tablas

Nombre: usuario		
Descripción: Tabla para registrar los datos pertenecientes a los usuarios del sistema		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla
nombre	varchar(150)	Nombre del usuario
username	varchar(150)	Nombre con que se identifica
password	varchar(150)	contraseña
fecha_nacimiento	date	Fecha de nacimiento de usuario
version	integer	Atributo para persistir o actualizar la entidad

eliminado	bit	Permite verificar si el usuario ha sido eliminado
cid	integer	Identificador de modificaciones registradas en la bitácora
cuenta_habilitada	bit	Atributo para crear la cuenta
primer_apellido	varchar	Primer apellido del usuario
segundo_apellido	varchar	Segundo apellido del usuario
direccion_particular	varchar	Dirección del usuario
cedula	varchar	Numero de cédula del usuario
pasaporte	varchar	Numero de pasaporte del usuario
telefono	varchar	Número de teléfono del usuario
id_profile	integer	Id creado para ese usuario

Tabla 3.5 Usuario

Nombre: muestra		
Descripción: Tabla para registrar los datos de la muestra		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la tabla
numero	integer	Numero de la muestra
eliminado	bit	Atributo para eliminar
cid	integer	Identificador de modificaciones registradas en la bitácora
version	integer	Atributo para persistir o actualizar la entidad
motivo_rechazo	varchar	Motivo de rechazo de la muestra

Tabla 3.6 Muestra

3.5 Vista de Implementación

Esta vista muestra la organización del código y el código actual de ejecución. Contiene una visión general del modelo de implementación y su organización partiendo de módulos en paquetes y capas. También se describe la asignación de paquetes y clases de la vista lógica a los paquetes y módulos de la vista de implementación. Es un subconjunto del modelo de implementación.

Diagrama de componentes

Los componentes identifican objetos físicos que hay en tiempos de ejecución, de compilación o de desarrollo y tienen identidad propia con una interfaz bien definida.

Un diagrama de componentes modela los aspectos físicos de un sistema, la vista de implementación estática de un sistema y los elementos físicos que residen en un nodo tales como: ejecutables, tablas, librerías, archivos y documentos. Muestra un conjunto de componentes y sus relaciones. Está compuesto por:

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

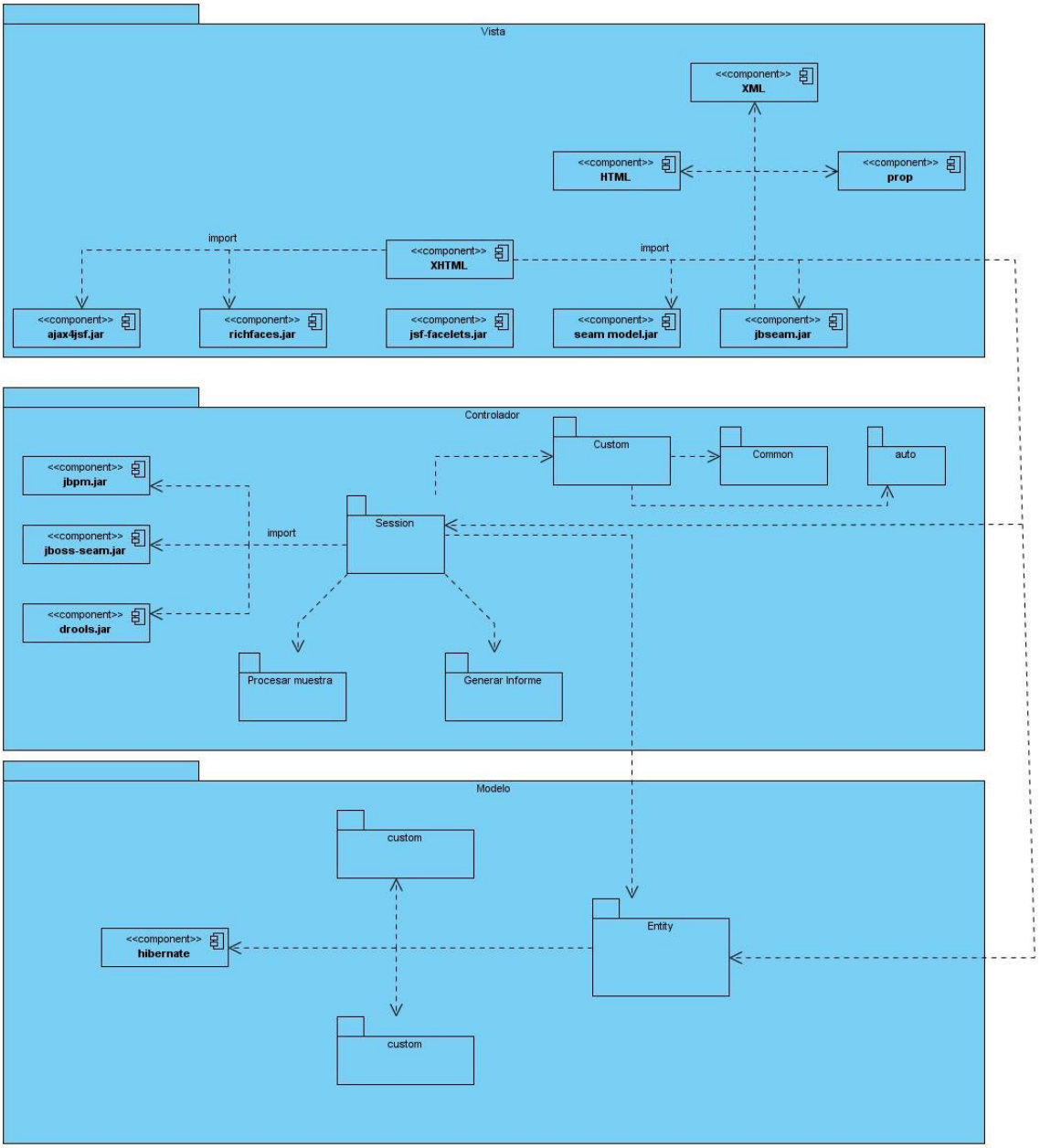

Figura 3.11 Diagrama de componentes

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

Un algoritmo se define como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas o también como un conjunto de instrucciones que realizadas en orden que conducen a obtener la solución de un problema. Y la complejidad de estos es una métrica teórica que proporciona una idea de cuánto va a tardar un algoritmo en resolver dicho problema.

Para el desarrollo de los procesos generar informes y procesar muestra del módulo Anatomía Patológica del sistema de información hospitalaria alas HIS se implementan algoritmos de alta complejidad. Dentro de los que se encuentran:

- Buscar informe final
- Ver datos de informe final de biopsia
- Ver datos de informe final de citología
- Liberar informe final para solicitudes de análisis
- Buscar solicitud de procesamiento
- Seleccionar solicitud de procesamiento
- Crear imágenes para cortes finos de microscopía electrónica.

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.[13]

[13] Algoritmia.net. (2008). Algoritmia.net. Obtenido el 01 07, 2010, de http://www.algoritmia.net/articles.php?id=10

Entre las implemementaciones de estructuras de datos que contiene el paquete Java.util se encuentran las listas las cuales son un conjunto de elementos con un orden concreto. Estas son interfaces que están compuestas por varios métodos y el ArrayList es una clase que implementa la interfaz.

3.8 Descripción de las clases que se utilicen para representar computacionalmente esta 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).

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. [14]

En este capítulo se realizó una valoración del diseño propuesto por el analista del sistema, se hizo la descripción de las clases de los casos de uso correspondientes así como los diagramas de clase de análisis e interacción para facilitar la implementación por parte del desarrollador, además de mostrar el modelo de datos, la descripción de las tablas con las que interactúa el modulo de Anatomía Patológica y la vista de Implementación.

[14] Java.util Public interface: List.Obtenido el 01 07, 2010, de http://www.docjar.com/

Capítulo 4: Modelo de prueba

En este capítulo se presenta 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.

Cuando se considera que un módulo está terminado se realizan las pruebas sistemáticas, el objetivo de estas es buscar fallos a través de un criterio específico, estos criterios se denominan "pruebas de caja negra y de caja blanca".

4.1 Pruebas de caja negra

Se conocen también como pruebas de caja opaca, pruebas funcionales, pruebas de entrada-salida y pruebas inducidas por los datos. Son complementarias a las pruebas de caja blanca y se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales. El probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra son aquellas que se enfocan directamente en el exterior del módulo, sin importar el código, son pruebas funcionales en las que se trata de encontrar fallas en las que este no se atiene a su especificación, como ser interfaz con el usuario, apariencia de los menús, control de las teclas, entre otros.

Este tipo de pruebas no es aplicable a los módulos que trabajan en forma transparente al usuario. Entre los errores habituales que pueden encontrarse tras aplicar este método están: funciones inexistentes o incorrectas, errores de interface, errores de iniciación, comienzo o finalización y errores de rendimiento.

Para llevar a cabo dichas pruebas existen varias técnicas, entre ellas están:

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

Se utilizará la técnica de la partición de equivalencia con modificaciones en el diseño de los casos de pruebas. Estos, son un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

Lo que caracteriza un escrito formal de caso de prueba es que hay una entrada conocida y una salida esperada, los cuales son formulados antes de que se ejecute la prueba. La entrada conocida debe probar una precondición y la salida esperada debe probar una pos condición. Formalmente, los casos de prueba escritos consisten principalmente en tres partes con subdivisiones:

Introducción o visión general contiene información general acerca de los Casos de Prueba que consta de:

- Un identificador único para futuras referencias, por ejemplo, mientras se describe un defecto encontrado.
- El dueño o creador, que es el nombre del analista o diseñador de pruebas, quien ha desarrollado pruebas o es responsable de su desarrollo.
- Versión, que es la actual definición del caso de prueba.
- El nombre del caso de prueba, que debe ser un título entendible por personas, para la fácil comprensión del propósito del caso de prueba y su campo de aplicación.
- Un identificador de requerimientos el cual está incluido por el caso de prueba. También aquí puede ser identificador de casos de uso o especificación funcional.
- El propósito, que contiene una breve descripción del objetivo de la prueba, y la funcionalidad que chequea.
- Las dependencias entre los CU.

Actividades de los casos de prueba:

- El ambiente de prueba o configuración, que contiene información acerca de la configuración del hardware o software en el cual se ejecutará el caso de prueba.
- La inicialización, que es donde se describen acciones, que deben ser ejecutadas antes de que los casos de prueba se hayan inicializado. Por ejemplo, no se deben abrir algún archivo.

- La finalización, que describe acciones, que deben ser ejecutadas después de realizado el caso de prueba. Por ejemplo si el caso de prueba estropea la base de datos, el analista debe restaurarla antes de que otro caso de prueba sea ejecutado.
- Las acciones o pasos a realizar para completar la prueba.
- La descripción de los datos de entrada

Resultados

- Los resultados esperados, que contienen una descripción de lo que el analista debería ver tras haber completado todos los pasos de la prueba.
- Los resultados reales, que contienen una breve descripción de lo que el analista encuentra después de que los pasos de prueba se hayan completado. Esto se sustituye a menudo con un Correcto o Fallido. Si un caso de prueba falla, frecuentemente la referencia al defecto implicado.

Descripción de los casos de pruebas

Liberar informe final de biopsia.

Caso de prueba: Sección 1: Liberar informe con microscopía electrónica.

Escenarios del Informe con microscopía electrónica	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de biopsia con microscopía electrónica	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica, el diagnóstico microscópico y el diagnóstico de microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada. Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica, el diagnóstico microscópico y el diagnóstico de microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.1 SC 1: Liberar informe final para biopsia con microscopía electrónica.

Caso de prueba: Sección 2: Liberar Informe con inmunohistoquímica.

Escenarios del Informe con inmunohistoquímica	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de biopsia con inmunohistoquímica	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica, el diagnóstico microscópico y diagnóstico de inmunohistoquímica asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada. Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica, el diagnóstico microscópico y diagnóstico de inmunohistoquímica asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.2 SC 2: Liberar Informe para biopsia con inmunohistoquímica

Caso de prueba: Sección 3: Liberar Informe con inmunohistoquímica y microscopía electrónica.

Escenarios del Informe con inmunohistoquímica y microscopía electrónica	**Descripción de las funcionalidades**	**Flujo Central**
EC 1: Liberar informe final de biopsia con inmunohistoquímica y microscopía electrónica	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica, el diagnóstico microscópico, diagnóstico de inmunohistoquímica y microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada.

		Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica, el diagnóstico microscópico, diagnóstico de inmunohistoquímica y microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.3 SC 3: Liberar Informe para biopsia con inmunohistoquímica y microscopía electrónica

Caso de prueba: Sección 4: Liberar Informe de estudios complementarios.

Escenarios del Informe de estudios complementarios	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de biopsia con estudios complementarios	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica y el diagnóstico microscópico asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada. Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de biopsia. Se muestra la descripción macroscópica y el diagnóstico microscópico asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.4 SC 4: Liberar Informe para biopsia de estudios complementarios

Id del escenario	EC 1	EC 2
Escenario	Liberar informe final de biopsia	Cancelar operación
Botón 1 (Aceptar)	NA	
Botón 2 (Cancelar)		NA
Respuesta del Sistema	Regresa a la vista anterior.	Regresa a la vista anterior.
Resultado de la Prueba		

Liberar informe final de citología

Caso de prueba: Sección 1: Liberar Informe con microscopía electrónica.

Escenarios del Informe con microscopía electrónica	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de citología con microscopía electrónica	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica, el diagnóstico microscópico y el diagnóstico de microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada. Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica, el diagnóstico microscópico y el diagnóstico de microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Cancelar.

		Regresa a la vista anterior.

Tabla 4.5 SC 1: Liberar Informe para citología con microscopía electrónica

Caso de prueba: Sección 2: Liberar Informe con inmunohistoquímica.

Escenarios del Informe con inmunohistoquímica	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de citología con inmunohistoquímica	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica, el diagnóstico microscópico y diagnóstico de inmunohistoquímica asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada. Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica, el diagnóstico microscópico y diagnóstico de inmunohistoquímica asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.6 SC 2: Liberar Informe para citología con inmunohistoquímica

Caso de prueba: Sección 3: Liberar Informe con inmunohistoquímica y microscopía electrónica.

Escenarios del Informe con inmunohistoquímica y microscopía electrónica	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de citología con inmunohistoquímica y microscopía electrónica	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica, el diagnóstico microscópico, diagnóstico de inmunohistoquímica y microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada.

		Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica, el diagnóstico microscópico, diagnóstico de inmunohistoquímica y microscopía electrónica asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.7 SC 3: Liberar Informe para citología con inmunohistoquímica y microscopía electrónica

Caso de prueba: Sección 4: Liberar Informe de estudios complementarios.

Escenarios del Informe de estudios complementarios	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Liberar informe final de citología de estudios complementarios	Muestra el informe con los datos que corresponde según el tipo.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica y el diagnóstico microscópico asociado a la muestra en el formato del informe. Selecciona la opción Aceptar. Actualiza el estado de la muestra asociada. Regresa a la vista anterior.
EC 2: Cancelar operación	Cancelar la opción de liberar informe final.	Se muestra la interfaz Liberar informe final de citología. Se muestra la descripción macroscópica y el diagnóstico microscópico asociado a la muestra en el formato del informe. Selecciona la opción Cancelar. Regresa a la vista anterior.

Tabla 4.8 SC 4: Liberar Informe para citología de estudios complementarios

Id del escenario	EC 3	EC 4
Escenario	Liberar informe final de citología	Cancelar operación
Botón 3 (Aceptar)	NA	
Botón 4 (Cancelar)		NA
Respuesta del Sistema	Regresa a la vista anterior.	Regresa a la vista anterior.
Resultado de la Prueba		

En este capítulo se realizó el modelo de prueba. Utilizando el método prueba de caja negra la técnica de partición de equivalencia. Se obtuvieron los casos de prueba como artefactos generados de este flujo de trabajo, siendo estos una guía para las pruebas efectuadas.

Conclusiones

Se concluye:

- Los sistemas analizados a pesar de brindar soluciones avanzadas no fueron los más factibles a utilizar por concepto de costos, requerimientos, flexibilidad y funcionalidades implementadas.
- Se definieron las herramientas, tecnologías y arquitectura utilizadas para la construcción de la aplicación web de forma que favoreciera las necesidades de los usuarios.
- Se desarrolló la implementación de los procesos generar informe y procesar muestra, con los cuales se facilita la gestión de la información en el departamento Anatomía Patológica de las instituciones hospitalarias.
- Se desarrollaron las pruebas exploratorias, garantizando con ellas el correcto funcionamiento de los procesos implementados.

Recomendaciones

Con el objetivo de hacer más eficiente el sistema que se implementó y brindar un mejor servicio en el área de Anatomía Patológica para sus trabajadores se recomienda:

- Obtener un modelo estándar para la generación de informes de autopsias.
- Rediseñar las funcionalidades de manera que se minimice el acceso por parte de los usuarios la funcionalidad seleccionar muestra.

Bibliografía

1. Ajax4jsf Developer Guide. 2007.
2. Allen, Dan. Seam in Action. s.l.: Manning Publication, 2008.
3. BPS Business Publications Spain S.L. (2010). computing.es. Obtenido el 01 07, 2010,de http://www.computing.es/Noticias/200807300007/iSOFT-Espana-gestionara-un-Sistema-de-Informacion-Regional-en-Mexico.aspx
4. Compañia ISOFT. (2006). Hospital Nuestra señora de Fátima, modelo en la integración de las soluciones de ISOF con el sistema de reconocimiento de habla Speech Magic de Philips. Soluciones extenciables y adaptables a cualquier entorno sanitario Obtenido el 01 07, 2010, de ftp://ftp.scansoft.com/nuance/casestudies/cs_es_nuestrasenoradefatima.pdf
5. Java, Equipo manual de. (s.f.). Manual de Java. Obtenido el 01 07, 2010, de http://www.manual-Java.com/
6. Javid Jamae, Peter Johnson. JBoss in Action. s.l.: Manning Publications, (2009).
7. Jboss comunity. (s.f.). Hibernate. Obtenido el 01 07, 2010, de http://www.hibernate.org/
8. latina, O. (2009). Osmosis latina. Obtenido el 01 07, 2010, de http://www.osmosislatina.com/lenguajes/uml/basico.htm
9. Mauricio. (2006). My opera. Obtenido el 01 07, 2010, de http://my.opera.com/pelican0/blog/show.dml/564829
10. Masadelante.com. (2010). Masadelante.com. Obtenido el 01 07, 2010, de http://www.masadelante.com/faqs/plug-in.
11. Microsoft Corporation. (2010). MSDN . Obtenido el 01 07, 2010, de http://msdn.microsoft.com/es-es/library/default.aspx
12. Milán Cristo, Nadiezka. Resumen por procesos del módulo Anatomía Patológica. UCI : s.n., 2008.

13. MYSQL. (2003). MYSQL Hispano. Obtenido el 01 07, 2010, de http://www.mysql-hispano.org/page.php?id=16.

14. Pergaminovirtual.com. (2009). Pergamino virtual Buscador hispano. Obtenido el 01 07, 2010, de http://www.pergaminovirtual.com.ar/definicion/

15. Pozo, P. (s.f.). Clikear.com. Obtenido el 01 07, 2010, de http://www.clikear.com/manuales/uml/diagramasinteraccion.aspx

16. Red Hat. RichFaces developer Guide. 2007.

17. Rodríguez, C. L. (2003). Ejemplo de desarrollo software utilizando la metodologia RUP. Obtenido de http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/index.html

18. S.L., F. d. (2008). Web Taller. Obtenido el 01 07, 2010, de http://www.webtaller.com/construccion/lenguajes/

19. The Eclipse Foundation. (2010). Ganymede. Obtenido el 01 07, 2010, de http://www.eclipse.org/ganymede/

20. The postgreSQL Global Develoment Group. PostgreSQL 8.3.6 Documentation. 2008.

21. UI Development with JavaServer Faces.

22. Valencia, D. I. (2004). Eclipse(2.1)Java.

23. WordPress.com., B. d. (s.f.). El mundo informático. Obtenido el 01 07, 2010, de http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/

Glosario de términos

Bases de datos: es un “almacén” que permite guardar grandes cantidades de información de forma organizada para que luego se pueda encontrar y utilizar fácilmente.se define como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Biopsia: método de estudios de la Anatomía Patológica que se realiza a partir de fragmentos de un ser vivo para que el médico de un diagnóstico de una enfermedad determinada.

Citología: método de estudios de la Anatomía Patológica que se realiza a partir de fluidos de un ser vivo para que el médico de un diagnóstico de una enfermedad determinada.

CRUD: Funciones básicas Crear, leer, modificar y eliminar.

CVS (Concurrent Versions System): aplicación informática que mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto y permite que distintos desarrolladores potencialmente situados a gran distancia colaboren.

ERP: Enterprise Resource Planning. Sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa.

Fichero o formato de archivo informático: es una manera particular de codificar información para almacenarla en un archivo informático.

Histotecnólogo o técnico en Anatomía Patológica: es el profesional capacitado para procesar cualquier tipo de material biológico y volverlo apto para un estudio microscópico y eventualmente macroscópico, sea con fines diagnósticos, docentes, de investigación.

ID: Atributo identificador de una tabla en la base de datos.

JPA o Java Persistence API: Proporciona un estándar para gestionar datos relacionales en aplicaciones Java SE o Java EE. Su objetivo es no perder las ventajas que proporciona la orientación a objetos al interactuar con una base de datos.

Memory-leaks: es un error de software que ocurre cuando un bloque de memoria reservada no es liberada en un programa de computación. Comúnmente ocurre porque se pierden todas las referencias a esa área de memoria antes de haberse liberado.

Necropsia: es el procedimiento técnico y científico de disección anatómica sistemática de un animal después de su muerte para dilucidar la causa de la misma. Es igual a un examen post mórtem o autopsia. La diferencia entre los dos términos es que la necropsia se realiza para confirmar las causas de la defunción en un hospital y la autopsia para averiguar las causas cuando fallecen de forma súbita y sin enfermedad aparente.

Patología: es la parte de la medicina encargada del estudio de las enfermedades en su más amplio sentido, es decir, como procesos o estados anormales de causas conocidas o desconocidas.

Plug-ins: Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Registro de Derivados: documento donde se recoge la cantidad de material que se utiliza en los análisis que se le realizan a las muestras es actualizado durante el procesado de las mismas para levar el control de lo que se manejó durante el proceso.

RIS (Radiology Information System) y PACS (Picture Archiving and Communications System): Nuevas tecnologías de imágenes médicas digitales, para el almacenamiento, distribución, presentación y administración de las imágenes que se toman generalmente en los servicios de radiología de los establecimientos de salud.

Singleton: Patrón cuya intención consiste en tener una instancia única de una clase y proporcionar un punto de acceso global a la misma.

Subversión: Sistema para el control de versiones que maneja los archivos y las carpetas de un proyecto y sus modificaciones en el transcurso del tiempo.