Trabajo de diploma para optar por el título

Ingeniero en Ciencias Informáticas

TÍTULO: ESTANDARIZACIÓN DEL DIAGNÓSTICO DE LAS PRUEBAS DEL MÓDULO ANATOMÍA PATOLÓGICA

Autor: Leonardo Carlos Fuentes Nasiff

Tutores: Ing. Nadiezka Milán Cristo

Ing. Yeinier Ferrás Cecilio

La Habana, julio de 2011

"Año 53 de la Revolución"

DATOS DE CONTACTO

Tutores:

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

Correo electrónico: nmilan@uci.cu

Ing. 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

RESUMEN

El presente trabajo tiene como objetivo diseñar e implementar las funcionalidades que dan respuestas a los procesos de gestión de la información que se obtiene durante la realización de los diagnósticos anatomopatológicos dentro del área Anatomía Patológica de las instituciones hospitalarias. También su integración a las funcionalidades del proceso realizar análisis, ya existentes en el módulo Anatomía Patológica del Sistema de Información Hospitalaria alas-HIS.

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

Se prevé que el sistema facilite el trabajo con los diagnósticos de las pruebas realizados en esta área, obtener los datos de una forma homogénea y estructurada, y permita además generar reportes estadísticos de determinadas enfermedades.

ÍNDICE

Índice de tablas

Índice de figuras

Introducción

La aplicación y utilización de los sistemas de información en el ámbito médico ha demostrado grandes mejoras tanto en la optimización de los recursos vinculados a la salud como a la calidad de atención. Estos surgen a partir de la década del 70 y posteriormente dan lugar a los Sistemas de Información Hospitalaria, los cuales, buscan elevar la calidad de la atención del paciente, de los servicios brindados en las instituciones hospitalarias y aplicar la información obtenida en diversas áreas como son: la clínica, la de investigación, la docente y la de administración. Además de reducir los costos y elevar la productividad.

Los Sistemas de Información Hospitalaria tienen una serie de componentes encargados de brindar una correcta representación de toda la información relativa a los procesos hospitalarios. Esto históricamente se ha logrado mediante el uso de clasificaciones estándares, utilizadas para minimizar la problemática propia del lenguaje natural.

El uso de una terminología estándar posibilita una representación más acabada de la gestión hospitalaria, además de abrir las puertas a la interoperabilidad entre los Sistemas de Información de este tipo, posibilitando el intercambio de información entre instituciones de salud. (1)

La Universidad de las Ciencias Informáticas cuenta con un Departamento de Sistemas de Gestión Hospitalaria el cual tiene en desarrollo un Sistema de Información Hospitalaria llamado alas HIS. Este sistema cuenta con el módulo de Anatomía Patológica. Está compuesto por 13 procesos de los cuales 5 cuentan con funcionalidades que permiten el registro o visualización del los diagnósticos anatomopatológicos.

En la actualidad su implementación permite registrar los diagnósticos a apreciación de los usuarios finales del sistema, es decir, los profesionales ingresan los diagnósticos mediante una, o varias líneas de texto narrativo, donde tienen absoluta libertad para describir dichos diagnósticos en una variada cantidad de caracteres. Por esta razón la información no es homogénea ni está estructurada, además no se garantiza un correcto entendimiento entre el resto de los módulos del sistema. Esta situación imposibilita que los informes puedan ser consultados por otras aplicaciones y que se generen estadísticas específicas de determinadas enfermedades.

Analizando la situación anterior se define como **problema a resolver**: ¿Cómo estandarizar los diagnósticos de las pruebas del módulo de Anatomía Patológica del Sistema de Información Hospitalaria alas HIS?

El **objeto de estudio** lo constituyen los estándares para diagnósticos de pruebas en los sistemas de gestión hospitalaria. El **campo de acción** está enmarcado en los estándares para diagnósticos de pruebas en el módulo de Anatomía Patológica del Sistema de Información Hospitalaria alas HIS.

El **objetivo de la investigación** es estandarizar los diagnósticos de las pruebas del módulo de Anatomía Patológica del Sistema de Información Hospitalaria alas HIS, permitiendo la compatibilidad con el resto de los módulos de dicho sistema.

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

1. Analizar los procesos asociados al área de Anatomía Patológica del Sistema de Información Hospitalaria alas HIS e identificar aquellos en los que se registran diagnósticos a las pruebas.

2. Aplicar la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.

3. Evaluar las tendencias actuales en el mundo, sobre los estándares definidos para el diagnóstico de las pruebas realizadas en el área Anatomía Patológica.

4. Actualizar el diseño del sistema según los estándares definidos para el diagnóstico de las pruebas realizadas en el área Anatomía Patológica.

5. Implementar las funcionalidades según los estándares definidos para el diagnóstico de las pruebas realizadas en el área Anatomía Patológica para los procesos:

- ✓ Estudiar muestra para diagnóstico definitivo.
- ✓ Generar informes.
- ✓ Entregar informes.
- ✓ Iniciar autopsia.
- ✓ Estudiar muestra para diagnóstico definitivo de autopsia.

6. Garantizar la integración de las nuevas funcionalidades con el sistema alas HIS.

7. Probar las funcionalidades implementadas.

8. Liberar el módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS con las nuevas funcionalidades implementadas.

Se prevé con la aplicación del estándar, la homogenización de los diagnósticos de las pruebas del módulo de Anatomía Patológica garantizando un correcto entendimiento entre el resto de los módulos del sistema. Además de obtener los informes finales de los estudios realizados con una estructura uniforme, facilitando la comprensión de los mismos a los usuarios finales.

El presente documento está estructurado por cuatro capítulos los cuales se relacionan a continuación:

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

Capítulo 2 Descripción de la arquitectura: 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: Centrada en el modelado detallado y la construcción de la aplicación.

Capítulo 4 Modelo de prueba: Se analizan los métodos de prueba a usar y se describen los casos de uso correspondientes.

Capítulo1 Fundamentación teórica

El presente capítulo tiene como puntos esenciales a tratar, los conceptos básicos relacionados con el análisis de estándares existentes en el área de Anatomía Patológica. Son analizados a nivel nacional e internacional, así como las tecnologías, herramientas y metodologías que se utilizan en el desarrollo de la investigación justificando el uso de cada una de ellas.

1.1 *Conceptos básicos relacionados con el campo de acción*

Estándar

Palabra de origen inglés, tiene varios significados: en tecnología y otros campos, un estándar es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad.

Estandarización o Normalización

Redacción y aprobación de normas que se establecen para garantizar el acoplamiento de elementos construidos independientemente. Según la ISO (Organización Internacional para la Estandarización) la Normalización es la actividad que tiene por objeto establecer, ante problemas reales o potenciales, disposiciones destinadas a usos comunes y repetidos, con el fin de obtener un nivel de ordenamiento óptimo en un contexto dado, que puede ser tecnológico, político o económico.

Prueba de Papanicolaou

También conocida como la prueba Pap o examen de citología, es una forma de examinar células recolectadas del cuello uterino y la vagina, la misma puede demostrar la presencia de infección, inflamación, células anormales o cáncer.

1.2 *Antecedentes*

Un Sistema de Información Hospitalaria consta de varios módulos donde cada uno está orientado a un área específica en las instituciones hospitalarias. Dentro de las áreas que poseen dichas instituciones se encuentra Anatomía Patológica; la cual permite registrar el flujo de información desde que la solicitud de análisis que ingresa al área hasta que se obtiene el diagnóstico final a través de los diferentes estudios que se realizan sobre las muestras.

En muchos países existen empresas dedicadas y patentadas para la producción de estos tipos de sistemas, a continuación se muestran características fundamentales de algunos productos.

1.2.1 AvantPat

Es un sistema que almacena, organiza y gestiona la información relacionada con los estudios que se realizan en los departamento 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 video, permite al patólogo diseñar nuevos estudios o agregar campos a los prediseñados en las plantillas.

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

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

Engloba cinco estudios básicos prediseñados como son: citología vaginal, citología de líquidos, punción aspirativa con aguja fina, biopsia y necropsia, los cuales cuentan con dinamismo entre sus campos de forma tal que el usuario tiene la posibilidad de adicionar nuevos campos a los mismos. Cada estudio posee cuatro secciones: datos generales, laboratorio, galería de imágenes y una sección que se recogen los resultados anatomopatológicos: descripción macroscópica, microscópica, resultado anatomopatológicos final, entre otros. (2)

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

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

1.2.2 Sarcap

Desarrollado en el ISMM "Dr. Luis Díaz Soto" y posteriormente aplicado en otros departamentos de Anatomía Patológica del país, un Sistema Automatizado de Registro y Control de Anatomía Patológica, con el objetivo de registrar y controlar de forma automatizada la información obtenida de las biopsias y autopsias, para facilitar su procesamiento e interrogación y poder ser utilizada en la enseñanza y en la confección de estudios e informes científicos, asistenciales y administrativos.

El 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 descripción, según la Clasificación Internacional de Enfermedades de la OMS y el SNOMED.

El SARCAP 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. Con él, se ha creado además el Banco de Datos de Autopsia Nacional, logro que sea propósito aún de países con gran desarrollo. (3)

1.2.3 Pat-win

PAT-win, es para anatomía patológica la solución que permite realizar una gestión integrada de los departamentos anatomopatológicos. Puede ser utilizada dentro del sistema de información de un hospital o de forma individual en servicios o laboratorio de esta área de la salud. La gestión permite principalmente registrar las muestras que llegan al departamento, especificar las diferentes técnicas, realizar descripciones macroscópicas y microscópicas, incluir observaciones realizadas, así como asignar los resultados diagnósticos, gestionando las salidas, las impresiones de informes y el control de costos. PAT- win, con más de diez años en el mercado, es el sistema de información líder y con más cobertura en el mercado español. (4)

Características:

- ✓ Integración con el sistema de información hospitalaria del centro que permite mantener un historial único de pacientes.
- ✓ Multicentro: permite integrar información de varios servicios de anatomía patológica (base de datos única).

- ✓ Multinforme: permite trabajar con estudios independientes de un mismo paciente y poder recuperar toda la información en un único informe.
- ✓ Compatibilidad con los sistemas operativos Windows 95, 98, 2000, XP.
- ✓ Uso de las bases de datos más extendidas: Informix, Oracle, MS SQL Server, Sybase.
- ✓ Interfaz gráfica de fácil manejo y muy intuitivo, con posibilidad de utilizar atajos de teclado.
- ✓ Posibilidad de gestión y confección de todo tipo de estudios (biopsias, citologías generales, ginecológicas, PAAF, autopsias, inmunohistoquímica, casos consulta) fácil acceso a otros estudios de pacientes.
- ✓ Módulo de gestión de imágenes (opcional)
- ✓ Corrector ortográfico integrado
- ✓ Gran número de listados y estadísticas.

Funcionalidades

- ✓ Incluyen módulo de laboratorio (listas de trabajos bloques, microtomías, tinción). Gestiona el trabajo pendiente y realizado por cada técnico de laboratorio.
- ✓ Gestor/Editor de informes integrado, disminuyendo los tiempos de acceso a los informes al evitar el uso aplicaciones externas.
- ✓ Permite el uso de sistemas de reconocimiento de voz para confección de descripciones macroscópicas, microscópicas y diagnósticos.
- ✓ Almacén integrado (posibilidad de integración con el almacén hospitalario).
- ✓ Gestión de modificaciones en informes emitidos. Notas adicionales con posibilidad de configuración (nota imprimible, no imprimible y tipos de notas), quedando constancia del patólogo que ha realizado la modificación y cuándo la ha llevado a cabo.

1.2.4 NovoPath

Es una aplicación para la completa gestión de los Servicios de Anatomía Patológica, fabricado por Vitro S.A., es el más veterano de los sistemas basados en Windows. Trabaja en entornos de red y puede ser conectado a otros equipos del laboratorio como cámaras digitales, marcadores (impresoras) de casetes y/o portas, inmunoteñidores, escáner, lectores de código de barras, etc. Dispone de módulos adicionales como los módulos de reconocimiento de voz y de visualización de informes en web. Se integra además con todos los Sistemas de Información Hospitalaria que se encuentran en el mercado. Permite directamente la gestión de imágenes (captura, almacenamiento, visualización) y la gestión de todo tipo de estudios anatomopatológicos (citología, biopsias, autopsias): (5)

- ✓ Gestión de patología quirúrgica.
- ✓ Gestión de citología.
- ✓ Gestión de autopsias.
- ✓ Otros estudios especiales.

1.3 *Tecnologías utilizadas en el proceso de desarrollo.*

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

1.3.1 CIE 10

Es la décima versión de la *Clasificación internacional de enfermedades* y determina los códigos utilizados para clasificar las enfermedades y una amplia variedad de signos, síntomas, hallazgos anormales, denuncias, circunstancias sociales y causas externas de daños y/o enfermedad. Cada condición de salud puede ser asignada a una categoría y recibir un código de hasta seis caracteres de longitud (en formato de X00.00). Cada una de tales categorías puede incluir un grupo de enfermedades similares. Se desarrollo en 1992 y su propósito fue rastrear estadísticas de mortalidad.

Se utiliza a nivel internacional para fines estadísticos relacionados con morbilidad y mortalidad, los sistemas de reintegro y soportes de decisión automática en medicina. Además permite almacenar y consultar información diagnóstica para fines clínicos y patológicos, lo cual sirve de base para la compilación de estadísticas nacionales de la mortalidad y la morbilidad. (6)

1.3.2 Snomed-CT

Es una terminología clínica, multilingüe y codificada que cuenta con una gran amplitud, precisión e importancia en el mundo. Es un producto que nace de la fusión entre Snomed RT (*Snomed Reference Terminology*), creada por el College of American Pathologist (CAP) y el Clinical Terms Version 3 (CTV3). Esta fusión ha dado lugar a una terminología de referencia que permite a los profesionales de la salud de todo el mundo representar la información clínica de forma precisa e inequívoca, en formato multilingüe. Actualmente esta terminología es mantenida por la International Health Terminology Standars Development Organization (IHTSDO). (7)

1.3.3 Código de Bethesda

El Sistema Bethesda es un sistema de información sobre diagnósticos de citología, que se utiliza para informar los resultados de la prueba de Papanicolaou. Se originó en Bethesda, Maryland en 1988, en un seminario-taller organizado por el Instituto Nacional de Cáncer de los Estados Unidos. Bethesda tiene como objetivo principal comunicar al médico solicitante la mayor información posible para ser utilizada en el manejo del paciente, a través de un informe descriptivo en el que se incluyan los aspectos citológicos a nivel hormonal, morfológico y microbiológico.

Su última actualización fue en abril del 2001 y es además el más utilizado a nivel internacional. (8)

1.3.4 Java Enterprise Edition 5 (JEE5)

Es la plataforma que provee Sun Microsystem para dar soporte y desarrollar software para las empresas. El gran éxito de java como plataforma para el desarrollo de aplicaciones se encuentra en esta especificación, que no es más que un conjunto de librerías que establecen un estándar para lograr un producto altamente calificado. Con el lanzamiento de esta edición, la especificación dio un gran paso hacia la excelencia, pues incorporó nuevas tecnologías de punta que no estaban contempladas anteriormente y que indiscutiblemente le incorporan a la plataforma gran robustez y simplicidad a la hora de trabajar, suficientes razones para no dudar en establecerla como base para el desarrollo.

1.3.5 Java Virtual Machine

No es más que la base sobre la que se ejecutan todas las aplicaciones javas, encargándose de interpretar todo el código java y convertirlo al lenguaje nativo del sistema operativo en uso, esta es la razón por la que todas las aplicaciones javas son altamente portables.

1.3.6 Capa de vista.

XHTML

Es un lenguaje de marcas establecido por la World Wide Web Center que combina el Lenguaje de Marcas extensible (XML, por sus siglas en inglés) y el Lenguaje de Marcas de Hipertexto (HTML, por sus siglas en inglés) en un mismo formato. Permite extender etiquetas propietarias, además de tener una codificación más rigurosa que HTML. Es el leguaje de marcas por defecto que utiliza JSF. (9)

Java Script

Es un lenguaje script e interpretado que se utiliza para incorporar interactividad en las páginas del sistema, este permite capturar los eventos dentro de las páginas para ejecutar alguna acción así como ser responsable de controlar algunas validaciones a nivel de cliente.

Java Server Face (JSF)

Es un framework (marco de trabajo) orientado a la construcción de interfaces gráficas de usuario (GUI), este marco separa el comportamiento de la presentación, teniendo su propio server como controlador, esto pone de manifiesto el uso del patrón de diseño modelo-vista-controlador, lo que hace más simple su utilización. Otra de las facilidades que brinda su uso, es proporcionar un modelo basado en componentes y dirigido por eventos, filosofía muy similar a las aplicaciones GUI standalone. JSF permite la creación de nuevos componentes por el desarrollador. (10)

Facelets

Es un marco de trabajo orientado a JSF para el trabajo con plantillas (templates), este permite contar con un tiempo cero para el desarrollo de los componentes de interfaces de usuario (UI) y permite separar dichos componentes en diferentes archivos, con soporte para el Lenguaje de Expresión (EL) para las cuestiones de validaciones sin la necesidad de configuraciones en archivos XML. Una de las ventajas que se explotan en el sistema de este framework es que no depende de un contenedor web.

Ajax4JSF

Es una librería de código abierto que se integra totalmente a la arquitectura de JSF. Hereda de los componentes JSF y los dota de funcionalidad AJAX de forma limpia y sin necesidad de utilizar código java script. Mediante este framework se puede alargar el ciclo de vida de JSF, recargar componentes sin la necesidad de hacerlo en la página completa además de hacer peticiones asíncronas al servidor.

Funciona a través de eventos que generan sus propios componentes, generando peticiones al contenedor AJAX sin la necesidad de preocuparse por crear el XMLHttpRequest, que no es más que el objeto que se obtiene de los contenedores web para hacer peticiones asincrónicas al servidor. (11)

Richfaces

Es una biblioteca de componentes JSF y un avanzado framework para la integración de AJAX con facilidades en la capacidad de desarrollo de aplicaciones de negocio. Sus componentes vienen listos para su uso out-of-the-box, por lo que los desarrolladores pueden ahorrar tiempo y de inmediato aprovechar las características de los componentes para crear aplicaciones web, que proporcionan mejoras en gran medida en la experiencia del usuario, haciéndola más fiable y rápida.

Incluye un fuerte apoyo para la skinnability (cambio de apariencia) de aplicaciones JSF y aprovecha al máximo los beneficios de este incluyendo la validación y conversión de instalaciones, junto con la gestión estática y dinámica los recursos. (12)

Seam UI

Serie de controles JSF altamente integrables con JBoss Seam. Adicionan varias mejoras a JSF, desde validación, expresiones Extended EL, integración de la navegación en la interfaz de usuario basada en pageflows o procesos del negocio.

1.3.7 Capa de control.

Seam

Es un poderoso y moderno framework creado para unificar todas las tecnologías estándares JSF, EJB3, JPA, además de BPM (Business Process Management).Fue creado desde el inicio para eliminar la complejidad a nivel desde arquitectura hasta una Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés), permitiendo la creación de complejas aplicaciones web basadas en POJOs, componentes de UI y XML. Se integra con librerías de controles de código abierto basadas en JSF como Richfaces e ICEFaces.

Integra, además, el concepto de workspaces (entornos de trabajo) permitiendo que el usuario tenga en varios tabs o ventanas del navegador actividades del negocio con contextos completamente aislados. Seam integra transparentemente la administración de procesos del negocio vía jBoss jBPM, haciendo muy fácil implementar y optimizar complejas colaboraciones e interacciones con el usuario. (13)

1.3.8 Capa de modelo.

PostgreSQL 8.3

Es un sistema de gestión de base de datos relacional orientada a objetos de código abierto. Algunas de sus principales características son: (14)

- ✓ Implementación del estándar SQL92/SQL99.
- ✓ Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetario, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- ✓ Incorpora una estructura de datos array.
- ✓ Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores.
- ✓ Soporta el uso de índices, reglas y vistas.
- ✓ Incluye herencia entre tablas.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- ✓ Permite mantener la integridad referencial.

Hibernate

Es una capa de persistencia objeto/relacional y un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y mejorada brinda la posibilidad de generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySql; además es open source (código abierto).

Permite expresar consultas en una extensión de SQL (HQL), así como en SQL nativo ó utilizando criterios orientado a objetos. Brinda filtros para trabajar con datos históricos, regionales ó condicionados por permisos. Puede ser usado para desarrollar y distribuir aplicaciones de de forma

gratuita. Posee una arquitectura de doble capa por lo que puede ser usado en un entorno agrupado. Soporta la generación automática de claves primarias. Realiza la persistencia de forma transparente sin procesamiento de código de bytes. Se integra con J2EE. (15)

Enterprise Java Bean 3 (EJB 3)

La necesidad de construir un sistema suficientemente robusto, obliga a identificar las tecnologías que mayor simplicidad ofrecen a la hora de implementar toda la lógica de negocio. EJB en su versión tres, se convierten en toda una plataforma que mediante POJOS permiten obtener servicios, ejecutando pequeñas porciones de código en el contenedor EJB, en este caso Jboss Aplication Server. Se puede contar con servicios de seguridad, persistencia, transacciones, mensajería.

En el caso de la persistencia, debido al auge de los motores de Mapeo de Objetos Relacionales (ORM, por sus siglas en inglés), surge una nueva especificación: Librería de Persistencia de Java (JPA, por sus siglas en inglés). Esta persiste automáticamente entidades basándose en la técnica de las ORM, define tres características principales: algunas metadatas (configuraciones) para establecer el mapeo objeto-relacional, un EntityManager para realizar las operaciones CRUD y un lenguaje de persistencia (JPQL).

1.3.9 Metodologías de desarrollo.

Proceso Racional Unificado (RUP)

Es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (16)

El proceso de software propuesto por RUP tiene tres características esenciales:

1. Guiado por los casos de uso: Los casos de uso no son sólo una herramienta para especificar los requisitos del sistema sino que constituyen un elemento integrador y una guía del trabajo, de su diseño, implementación y prueba. Además éstos no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

2. Centrado en la arquitectura: La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo.

3. Iterativo e incremental: Propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

Lenguaje Unificado de Modelado (UML)

Es un lenguaje que se utiliza para especificar, visualizar, construir y documentar un sistema de software orientado a objetos (OO). UML es gratuito, accesible a todos, y conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelado de sistemas grandes y complejos.

Además tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue. (17)

1.3.10 Lenguaje de programación.

Java

Es un lenguaje de programación desarrollado por la compañía Sun Microsystems, que utiliza el paradigma de la Programación Orientada a Objetos (POO). Es un lenguaje interpretado e independiente a la arquitectura que posee una amplísima biblioteca estándar de clases predefinidas. Además las aplicaciones Java pueden ser ejecutadas indistintamente en cualquier plataforma sin necesidad de recompilar, tienen un amplio espectro: programación tradicional, distribuida, GUI, web.

Con Java se pueden programar aplicaciones web dinámicas, con acceso a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. Este lenguaje es utilizado de manera horizontal en el desarrollo del sistema, pues puede estar presente en las diferentes capas de la aplicación.

1.3.11 Herramientas.

Eclipse

El término Eclipse identifica a la comunidad de software libre que lleva el desarrollo de la plataforma Eclipse. Esta comunidad tienen el objetivo de proporcionar una plataforma robusta, escalable y de calidad para el desarrollo de software con el Entorno de Desarrollo Integrado (IDE, Integrated Development Environment) Eclipse. Este IDE cuenta con un editor de código, un compilador/intérprete y un depurador. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación además de brindar la posibilidad de añadir a la plataforma base de Eclipse extensiones (plugins) para extender sus funcionalidades.

Jboss Tools

Es un plugin creado por la compañía Red Hat que se integra a la plataforma base de Eclipse para extender las funcionalidades de su desarrollo. Es toda una suite de herramientas que brinda un soporte desde eclipse a diferentes tecnologías que permiten un desarrollo óptimo para los programadores. Entre los módulos con los que cuenta se pueden apreciar:

- ✓ JBoss AS Tool: Provee cuatro servidores, con gran facilidad para arrancar, parar y depurar un proyecto de eclipse, además de presentar optimas características de despliegue.
- ✓ Richfaces VE: Editor visual que propone Exadel, que le da soporte a HTML y JSF además de brindar soporte para los componentes específicos de JBoss Richfaces.
- ✓ JBPM Tool: Editor para JBPM (Java Business Process Managment) que permite construir y desplegar este workflow.
- ✓ Struts Tool: Herramienta que da soporte al framework web Struts.
- ✓ JBossWS Tool: Herramienta para el trabajo con JBoss WebServices.
- ✓ JBoos ESB Tooling: Herramienta que permite el trabajo con los Enterprise Service Bus.
- ✓ Portlet Tool: Soporte para Portlet.
- ✓ Seam Tool: Incluye soporte para seam-gen, permitiendo todo lo relacionado a la refactorización del código.

PgAdmin III

Es una popular plataforma de administración y desarrollo para el gestor de bases de datos PostgreSQL, permite a los usuarios hacer desde simples consultas SQL hasta desarrollar bases de datos complejas. Esta herramienta fue diseñada para responder a las necesidades de todos los usuarios, cuenta con una interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente su 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 y soporte para el motor de replicación Slony-I.

Visual Paradigm para UML

Es una herramienta CASE profesional de licencia gratuita que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

Este software de modelado UML ayuda a una más rápida y mejor construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Soporta ingeniería inversa, generador de informes, editor de figuras, integración IDE con Eclipse, NetBeans y otros. Además entre sus ventajas se incluyen el modelado colaborativo con CVS y subversión, la generación de documentación y de código base para diferentes lenguajes de programación como Java y su exportación como HTML.

Por lo antes expuesto se hace necesario propiciarle al sistema desarrollado las funcionalidades según los estándares definidos para el diagnóstico de las pruebas realizadas en esta área. Con el objetivo de obtener una aplicación flexible y de costo mínimo, se definieron las siguientes herramientas que conforman el ambiente de desarrollo del módulo de Anatomía Patológica y los estándares a utilizar para el desarrollo de este sistema:

- ✓ Entorno de desarrollo: Eclipse.
- ✓ Gestor de Base de Datos: PostgreSQL 8.3.
- ✓ Herramienta de modelado: Visual Paradigm.

- ✓ Clasificación Internacional de Enfermedades. (CIE).
- ✓ Sistema de Bethesda.

En el capítulo fueron analizados varios sistemas pero se demostró que no son viables ya que presentan varios inconvenientes como:

- ✓ Son software propietario, su código fuente no es libre y no se le pueden realizar modificaciones.
- ✓ Han sido desarrollados para satisfacer las especificaciones de lugares donde van a ser desplegados.
- ✓ No son multiplataforma y generalmente son aplicaciones de escritorio lo cual aumenta los gastos por la necesidad de equipos de mayor potencia.
- ✓ Poseen un alto costo, para adquirirlos hay que pagar licencia, soporte técnico y actualizaciones lo que provoca que no sean factibles.

Capítulo2 Descripción de la arquitectura

El presente capitulo describe la arquitectura definida, los requerimientos no funcionales, la estrategia de integración a otros servicios, la seguridad a implementar en el sistema así como la estrategia de codificación.

2.1 *Requerimientos no funcionales*

Los requerimientos no funcionales son cualidades que el producto debe tener. A través de estos se especifican propiedades del sistema como restricciones de ambiente y desarrollo, rendimiento, dependencias de plataformas y mantenimiento. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. (18)

2.1.1 Usabilidad

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

Para alcanzar la categoría de Usuarios normales, serán necesarios 20 días de preparación, obteniendo un nivel Elemental asociado al dominio del sistema y el uso eficiente del mismo.

Para alcanzar la categoría de Usuarios avanzados, serán necesario 30 días de preparación, obteniendo un nivel avanzado asociado al dominio del problema y el uso eficiente del mismo.

Los usuarios podrán lograr sus objetivos con un mínimo esfuerzo y obteniendo los resultados máximos.

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

El entendimiento del sistema por parte del usuario es estimulado debido a la estructura concebida para la organización de la información.

2.1.2 Fiabilidad

Todas las acciones que se realizan serán registradas, llevando el control de las actividades de cada usuario.

La información que se haya ingresado al sistema no 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 a partir de respaldos o salvas realizadas.

El sistema contendrá un control de cambios a determinados campos de información, de forma tal que sea posible determinar cuáles han sido las modificaciones que se le han realizado.

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

2.1.3 Eficiencia

Se minimizará además el volumen de datos en las peticiones y se optimizara el uso de recursos críticos como la memoria. Para ello se establecerá como regla guardar en la memoria caché datos y recursos altamente demandados.

2.1.4 Soporte

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

2.1.5 Seguridad de acceso y administración de usuarios

Se mantendrá la seguridad y el control a nivel de usuario, esto garantiza el acceso sólo a los niveles establecidos de acuerdo a la función que realizan. Además las contraseñas sólo podrán cambiarlas el propio usuario o el administrador del sistema.

2.1.6 Respaldo y recuperación de datos

Las copias de la base de datos hacia otros dispositivos de almacenamiento externo estarán permitidas, además de recuperar la base de datos a partir de los respaldos realizados.

2.1.7 Auditoría

El chequeo de las operaciones y acceso de los usuarios al sistema estarán permitidas.

Debe existir un registro de trazas que almacene todas las transacciones realizadas en el sistema, indicando: usuario que realizó la transacción, tipo de operación que se realizó, fecha y hora en que se realizó la operación e información contenida en el registro modificado o accedido.

2.1.8 Configuración de parámetros

Está permitido establecer parámetros de configuración del sistema y actualización de nomencladores.

2.1.9 Réplica

Está permitido realizar réplica de la base de datos de los hospitales con el Centro de Datos, esta puede hacerse de forma manual y automatizada a través de la red.

2.1.10 Restricciones del diseño

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

2.1.11 Interfaz

Interfaces de usuarios

Los datos en las ventanas del sistema estarán claros y bien estructurados, además de permitir una correcta interpretación de la información. La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización. La entrada de datos detectada claramente e informada al usuario si es incorrecta. Todos los textos y mensajes en pantalla aparecerán en idioma seleccionado.

Interfaces de comunicación

Se utilizará el estándar HL7 para el intercambio electrónico de datos entre aplicaciones. El sistema usará el formato estándar WSDL para la descripción de los servicios web. Se implementarán mecanismos de encriptación de datos para el intercambio de información con sistemas externos, utilizará además mecanismos de compactación de los datos que se intercambiarán con sistemas externos con el objetivo de minimizar el tráfico en la red y economizar el ancho de banda.

2.1.12 Rendimiento

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

2.1.13 Hardware

Estaciones de trabajo

Se escogieron estaciones de trabajo de 256 MB de memoria RAM y un microprocesador de 2.0 Hz con sistema operativo Linux, ya que el sistema necesita para las estaciones una capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, se recomienda Internet Explorer 7, Firefox 2 o versiones superiores.

Servidores

La solución está conformada 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, 2x72 GB de disco duro y sistema operativo Linux. Servidores de Aplicaciones: 2DL380 G5, Procesador Intel® Xeon® 5140 Dual-Core, 4GB de memoria, 2x72 GB de disco duro y sistema operativo Linux. Servidores de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual-Core, 2 GB de memoria, 2x72 GB de disco y sistema operativo Linux.

2.1.14 Software

El sistema es compatible con los sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss AS y PostgreSQL). Contará además con un navegador web, estos pueden ser Internet Explorer 7, Opera 9, Google Chrome 1 y Firefox 2, o versiones superiores de estos.

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 mantenimiento,

auditable, flexible e interacción con otros sistemas de información, teniendo en cuenta las limitaciones derivadas de las tecnologías disponibles para implementarlos. (2)

Un patrón arquitectónico es un patrón de alto nivel que fija la arquitectura global de una aplicación. Definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Algunas características deseables en todo patrón son:

- ✓ Consistencia: todo el modelo presentado debe tener una consistencia entre sus distintos componentes.
- ✓ Familiaridad: debe ser conocido y usado por la comunidad de interés.
- ✓ Simplicidad: de fácil utilización, no debe agregar complejidad al diseño.

La arquitectura definida se basa en el patrón arquitectónico Modelo-Vista-Controlador, el cual separa los datos, las interfaces de usuario y la lógica de control en tres componentes distintos. Cada componente agrupa en módulos toda la aplicación, donde la vista engloba lo referente a las interfaces de usuario, el modelo comprende las clases de acceso a datos, y el controlador toda la lógica del negocio.

Modelo: Es el componente encargado del acceso a datos, representando las estructuras de datos del sistema. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.

Vista: Define la interfaz de usuario, la información presentada al usuario. Una vista puede ser una página web o una parte de una página.

Controlador: Responde a eventos y actúa como intermediario entre el modelo, la vista y cualquier otro recurso necesario para generar una página.

Ventajas de la utilización del patrón Modelo-Vista-Controlador (MVC):

- ✓ Facilidad para realizar cambios en la aplicación puesto que cuando se realiza un cambio de bases de datos, programación o interfaz de usuario solo se maneja uno de los componentes.

- ✓ Se puede modificar uno de los componentes sin conocer cómo funcionan los otros.

2.3 *Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser re usados*

La reutilización de código consiste en las técnicas y el comportamiento que garantizan que la totalidad o una parte de una aplicación informática existente se puedan emplear en la cimentación de otro programa. Esto garantiza el aprovechamiento del trabajo anterior, se economiza el tiempo y se reduce la redundancia.

La vía más fácil de reutilizar código es copiarlo parcial o totalmente desde un programa antiguo al que se está desarrollando, pero es demasiado trabajoso mantener muchas copias del mismo en código. Por esto es recomendable dejar el código reutilizable en un lugar común y llamarlo desde donde se necesite.

En el módulo de Anatomía Patológica se reutilizan algunos componentes que se utilizan de manera general por todos los módulos del Sistema de Información Hospitalaria alas HIS, estos componentes son: la clase Bitácora del módulo de Configuración para llevar a cabo el registro de todas las acciones que realiza el usuario: aceptar, modificar, ver, liberar, actualizar y crear; la clase ActiveModule, la cual brinda información sobre que módulo y entidad se encuentra el usuario que está utilizando el sistema; la clase ReportManager del módulo Común la cual se utiliza para la generación de informes.

2.4 *Seguridad*

En cualquier sistema de información la seguridad es de vital importancia ya que de ella depende la integridad de los datos. Para garantizar la seguridad en el área de Anatomía Patológica del Sistema de Gestión Hospitalaria alas HIS, toda la autorización a directorios, controles, páginas, opciones del menú y servicios del negocio está basada en reglas, las cuales no están contenidas en el código de la aplicación, sino en ficheros no compilados, los cuales le proveen a la aplicación un mayor dinamismo al no requerir recompilar en caso que cambie alguna. Esto puede llevarse a cabo por el evento de integración del motor de reglas JBoss Rules que brinda el Framework de Seguridad de Jboss Seam.

Por otra parte, toda la información ingresada al sistema nunca será eliminada físicamente de la base de datos, permitiendo al sistema recuperarse de posibles errores en su manejo.

Se realiza además el control a nivel de usuario y contraseñas, garantizando el acceso sólo a los niveles establecidos de acuerdo a la función que realiza cada usuario. Las contraseñas sólo pueden

ser cambiadas por el propio usuario o por el administrador del sistema. Se registra además las actividades realizadas por los usuarios en cada momento en una especie de bitácora, almacenándose la fecha, la hora, el usuario y la actividad que realizó.

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 modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

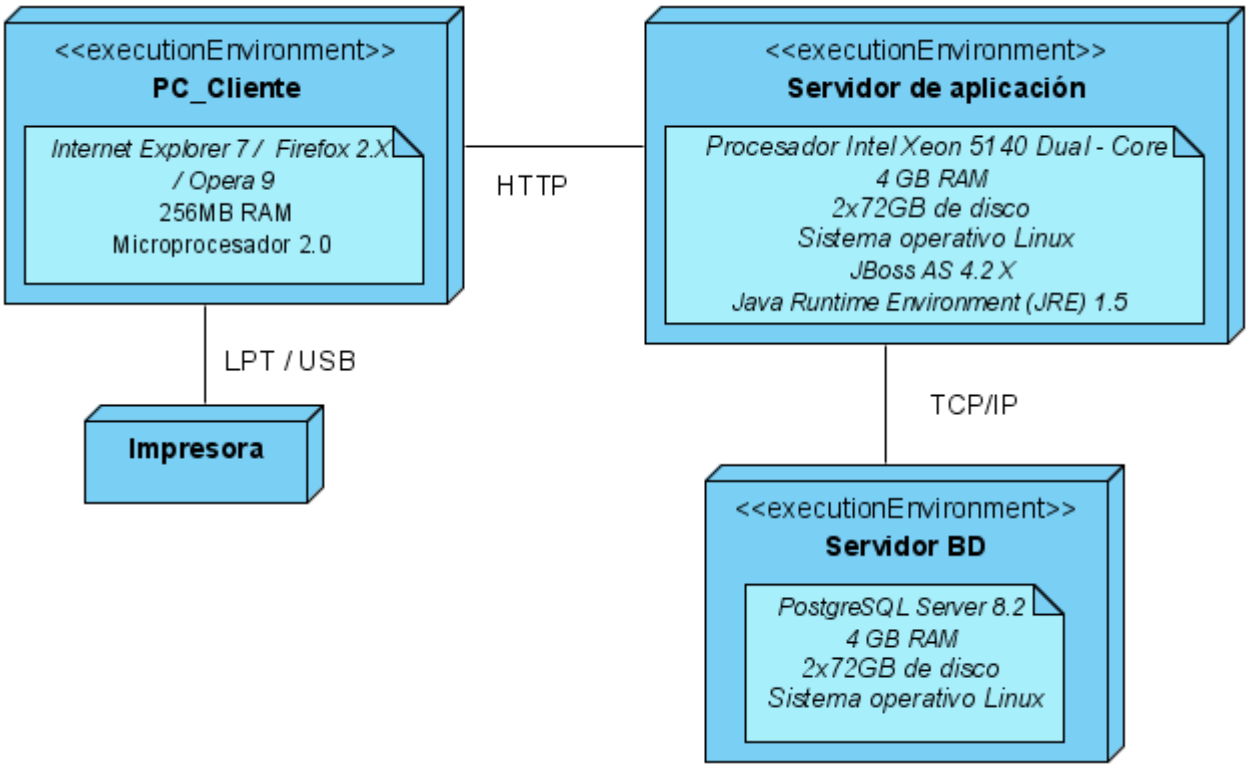

Figura 2.1. Diagrama de despliegue

2.6 *Estrategias de codificación. Estándares y estilos a utilizar*

Un estándar de codificación comprende todos los aspectos de la generación de código. Su utilización sirve como punto de referencia para los programadores a la hora de desarrollar las aplicaciones, mantiene un único estilo de programación, contribuye a mejorar el proceso de codificación, haciéndolo más eficiente y además eleva la posibilidad de mantener del código.

En el desarrollo del sistema alas HIS se utiliza el estándar de codificación definido en las especificaciones del lenguaje de programación Java: Java Code Conventions. A continuación se mostrarán los estándares de codificación a utilizar en el desarrollo del sistema:

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

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

Inicio y fin del 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, elsem for, while, do while, switch, foreach.

Aspectos Generales: El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador, ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito 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 un línea propia.

Comentarios, separadores, líneas, espacio en blanco y márgenes: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.

Líneas en blanco: Se emplean antes y después de métodos, clases y estructuras. Se recomienda dejar una línea en blanco antes y después 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.

Clases y Objeto: Nombrar las clases e instancias de forma estándar para todas la aplicaciones.

Apariencia de clases y objetos: Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en casos de que sea un nombre compuesto se empleará notación PascalCasing*. Para el caso de las instancias se comenzara con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.

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

Apariencia de las funciones: Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que realiza la función. Se empleará la notación PascalCasing*. Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set.

Declaración de parámetros en funciones: Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos (tabla 1.1).

Bases de Datos, Tablas, esquemas y Campos

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

Tablas que representen relaciones: El nombre a emplear para estas tablas de la relación debe comenzar con el prefijo tr, seguido de underscoard y el nombre será la concatenación del nombre de las dos tablas que la generaron separados por underscoard, todo en minúscula.

Tablas que representen nomencladores: El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre será corto y descriptivo, todo en minúscula.

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

Ejemplo: ‘id_producto’;

Sentencias SQL: Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.

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

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

Tipo Datos	Prefijo	Ejemplo
Int	i	iCantPacientes
flota	f	fPesoPaciente
double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	e	eSexo
byte	b	bCantDiasPaciente
sbyte	sb	sbEdadPaciente
short	sh	shVariableShort
ushort	us	usVariableUshort
uint	ui	uiVariableUint
long	l	lVariableLong
ulong	ul	ulVariableUlong
decimal	dc	dcVariableDecimal

Objetos	o	oPacienteHistorico
Objetos de tipo Struct	st	stUnaStruct

Tabla 2.1 Notación de los tipos de datos

En este capítulo se describió la arquitectura del sistema donde se expone una breve explicación sobre la utilización del patrón Modelo Vista Controlador, se analizaron los aspectos de seguridad. Se mostró su distribución física y sus conexiones mediante la Vista de Despliegue y se establecieron estándares de codificación y estilos a utilizar para garantizar una mejor comprensión y mantenimiento al código. Para agilizar el desarrollo y eliminación de código redundante se analizaron los componentes a reutilizar.

Capítulo3 Descripción y análisis de la solución propuesta:

El siguiente capítulo muestra la descripción y el análisis de la solución propuesta, a partir de esto se realiza una valoración de las técnicas de validación de una base de datos. Además se describen las nuevas clases u operaciones que se utilizarán por el módulo: obtener el modelo de datos, obtener los diagramas de clase de diseño e interacción, así como la vista de implementación.

3.1 *Valoración crítica del diseño propuesto por el analista*

Uno de los flujos más importantes en la ingeniería de software es el diseño, en este flujo se desarrollan, revisan y documentan los refinamientos progresivos de la estructura de datos, arquitectura, interfaces, y datos procedimentales de los componentes del software. En este flujo además se describen las clases, la organización de las clases en paquetes y subsistemas, uno de sus artefactos es el Modelo de Diseño.

Este último puede contener los diagramas, las clases, paquetes, subsistemas, interfaces, atributos, relaciones entre casos de usos, entre otros, los cuales pueden considerarse para el sistema en desarrollo. Además describe las clases sirviendo como abstracción para la implementación.

En la elaboración de este modelo, se tomaron en cuenta patrones de diseño, lo que está catalogado como una buena práctica de programación ya que estos patrones pueden ser reusables al aplicarse a diversos problemas en distintas circunstancias, esto ahorra tiempo y logra que el software sea más seguro, eficiente y dinámico.

El framework Seam garantiza la persistencia de los datos por medio de la abstracción que brinda Hibernate y el gestor de bases de datos a utilizar.

El patrón de diseño Abstract Factory, el cual se encarga del acceso a la base de datos, se encuentra implementado por el componente EntityManagerFactory, este se encarga de crear los objetos EntityManager, los cuales son la interfaz principal del JPA y son a su vez los responsables de realizar las operaciones CRUD sobre un conjunto de objetos persistentes.

Hibernate es el responsable además de la implementación de patrones de comportamiento y mapeo como Identity field, Foreing Key Mapping y el patrón Association Table Mapping. El primero se encarga de guardar un campo de identificación de la base de datos en un objeto para mantener la identidad entre una fila de una tabla de una base de datos y dicho objeto. El segundo se encarga de la asociación entre objetos para referencias de llave foránea entre tablas de la base de datos, y el tercer patrón es el responsable del mapeo entre las relaciones de mucho a mucho. Por último se utiliza

también por parte de Hibernate el patrón ActiveRecord el cual asocia filias únicas de la base de datos con objetos del lenguaje de programación que se está utilizando.

A continuación se mostrarán todos los diagramas utilizados de la aplicación que se propone:

- ✓ Diagrama de interacción: Son artefactos que muestran gráficamente las relaciones entre los objetos, incluyendo los mensajes que se pueden enviar entre ellos. Pueden utilizarse para modelar un flujo de control particular de un caso de uso y los aspectos dinámicos de los sistemas. Estos diagramas se clasifican en diagramas de colaboración y diagramas de secuencia.

- ✓ Diagrama de secuencia: Muestra las interacciones expresadas en función del tiempo, así como los objetos que participan y los mensajes que se intercambian a lo largo del tiempo.

- ✓ Diagrama de clases del diseño: Es donde se crea el diseño conceptual de la información que se manejara en el sistema, describen además la estructura de un sistema mostrando sus clases.

Respondiendo a la arquitectura definida 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, este se encuentra estructurado en dos subpaquetes:

- ✓ Sesiones: Contiene las clases controladoras propias del proceso, las controladoras personalizadas y las controladoras autogeneradas.

- ✓ Entidades: Contiene las entidades personalizadas y las autogeneradas de la base de datos. Las primeras pueden presentar relaciones de composición o herencia con las autogeneradas, y las segundas se obtienen mediante el proceso de ingeniería inversa de la base de datos, utilizando para ello el mapeo objeto-relacional de Hibernate.

A continuación se muestra el Diagrama de Paquetes del Diseño de la aplicación propuesta:

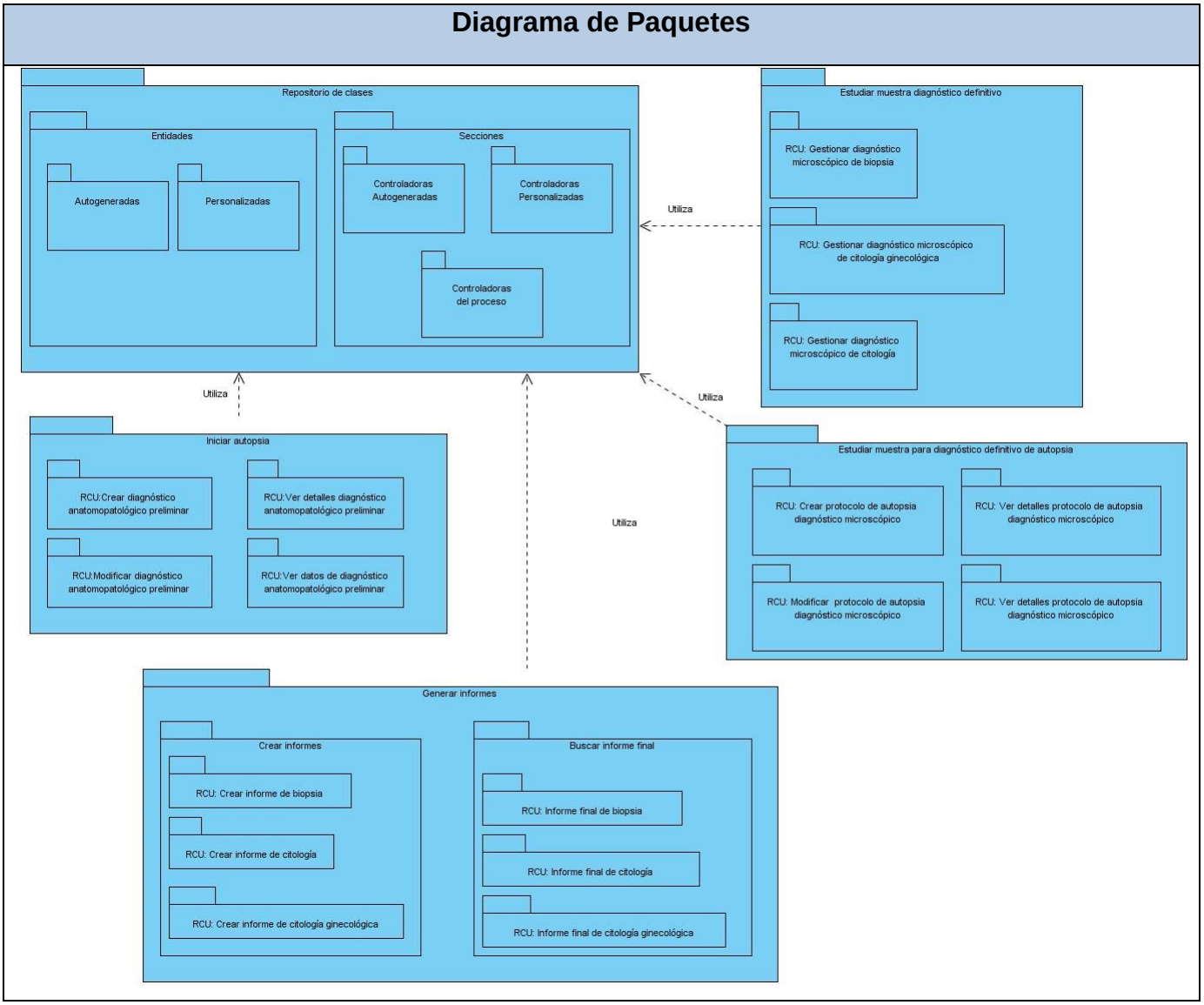

Figura 3.1. Diagrama de Paquetes.

A continuación se describen algunas de las realizaciones de los casos de usos del sistema propuesto, relacionados al proceso Estudiar muestra diagnóstico definitivo.

DCD_ Gestionar diagnóstico microscópico de biopsia

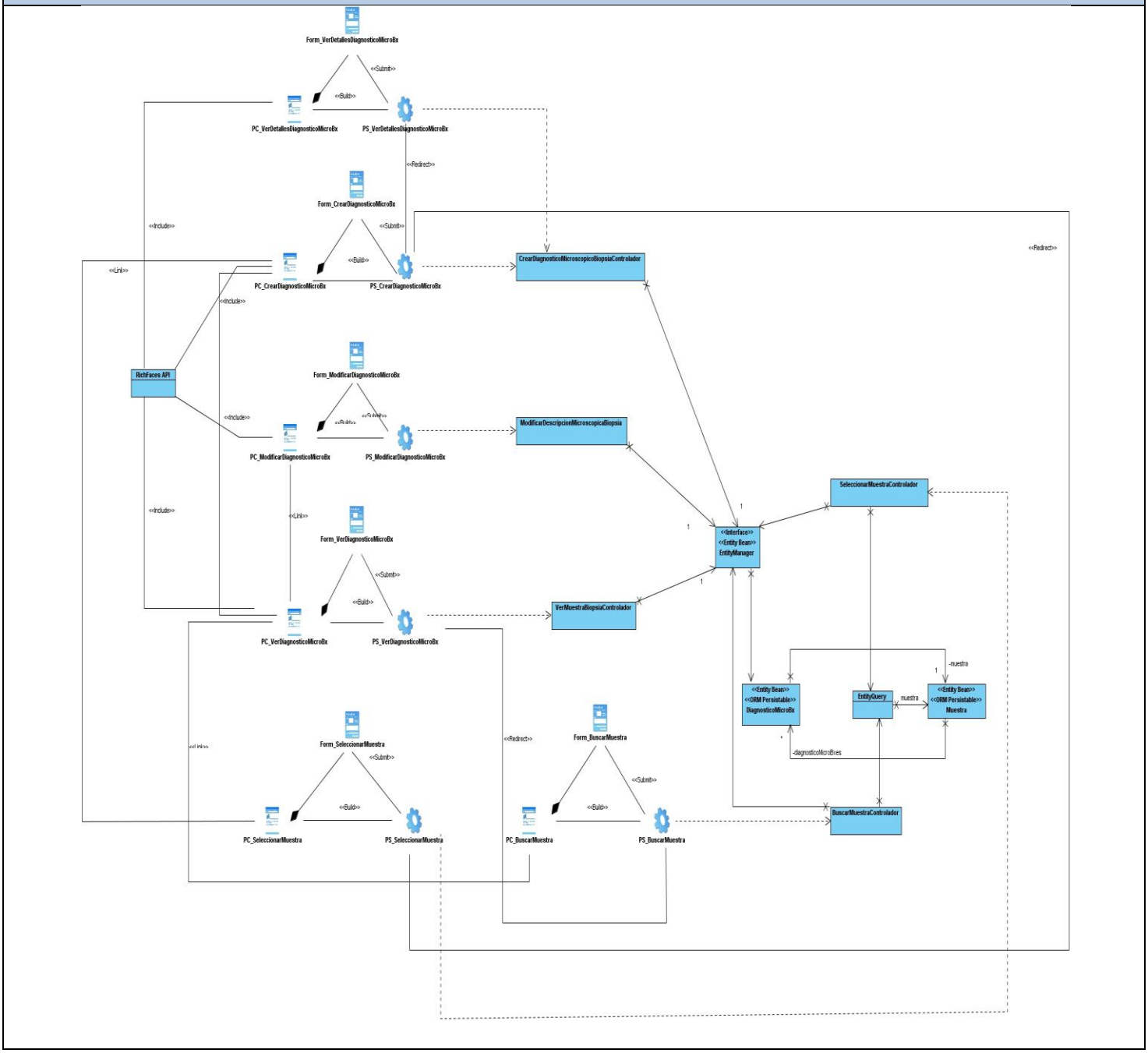

Figura 3.2. DCD Gestionar diagnóstico microscópico de biopsia.

DCD_ Gestionar diagnóstico microscópico de citología

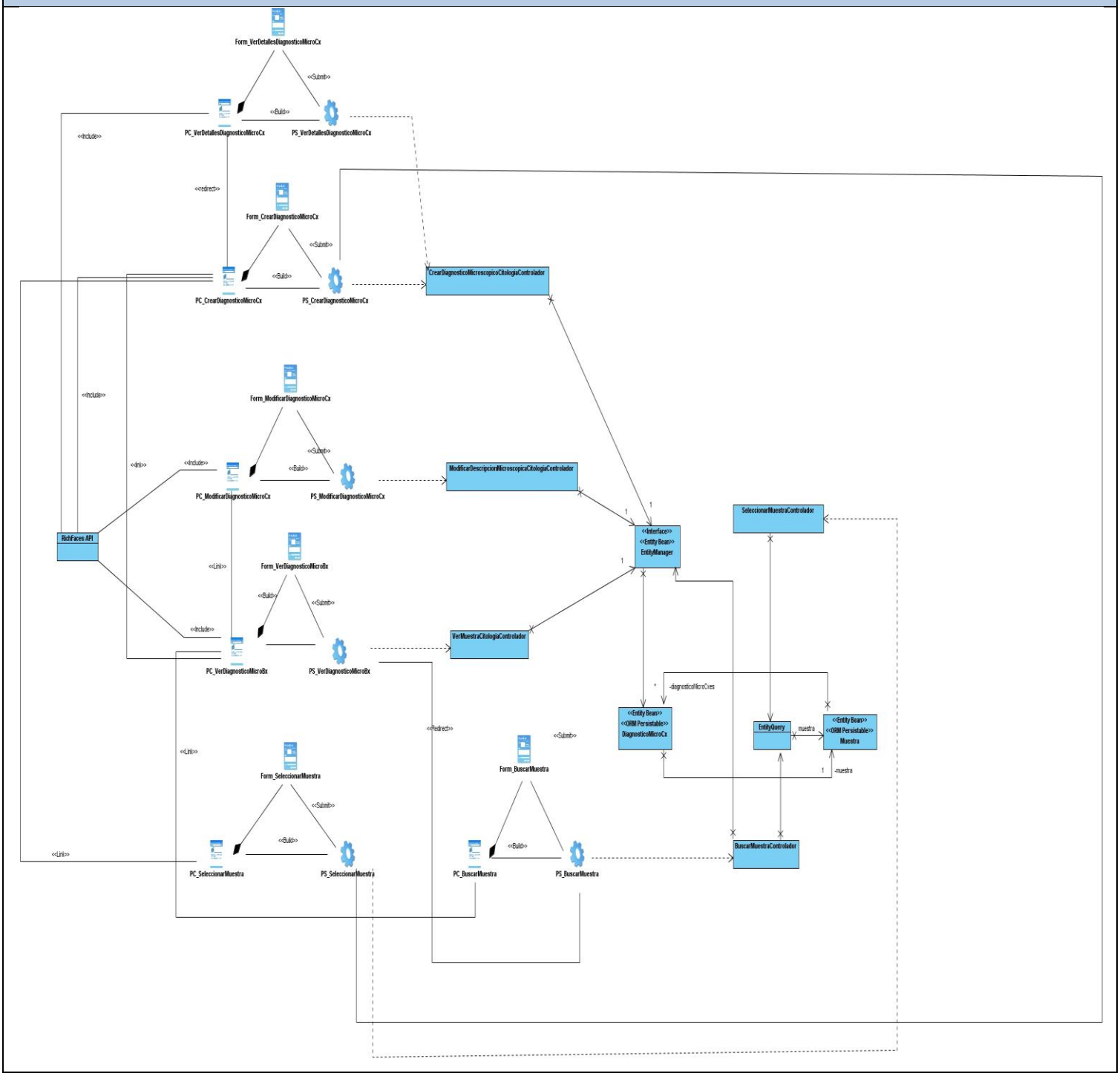

Figura 3.3. DCD Gestionar diagnóstico microscópico de citología.

DCD_ Gestionar diagnóstico microscópico de citología ginecológica

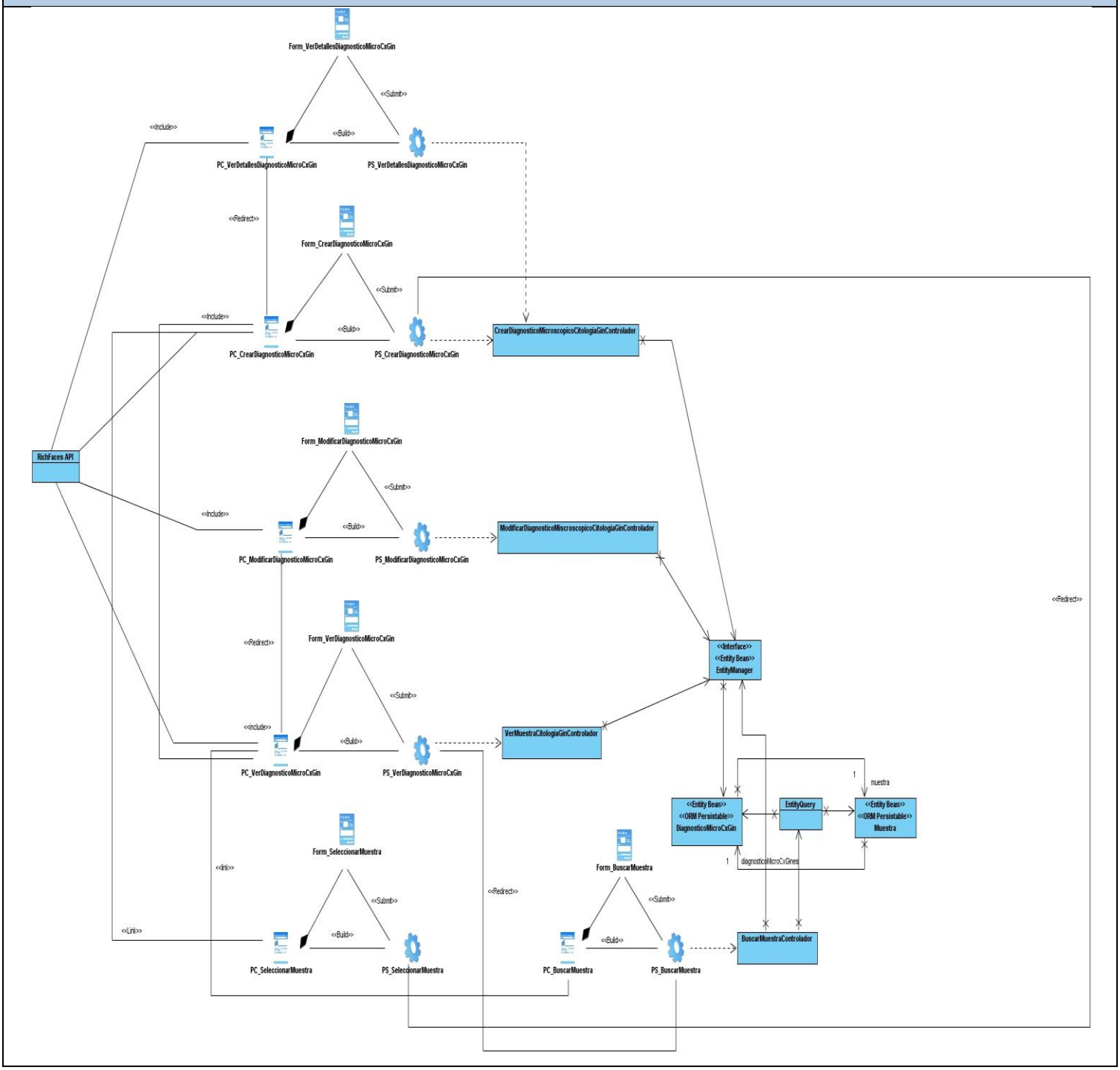

Figura 3.4. DCD Gestionar diagnóstico microscópico de citología ginecológica.

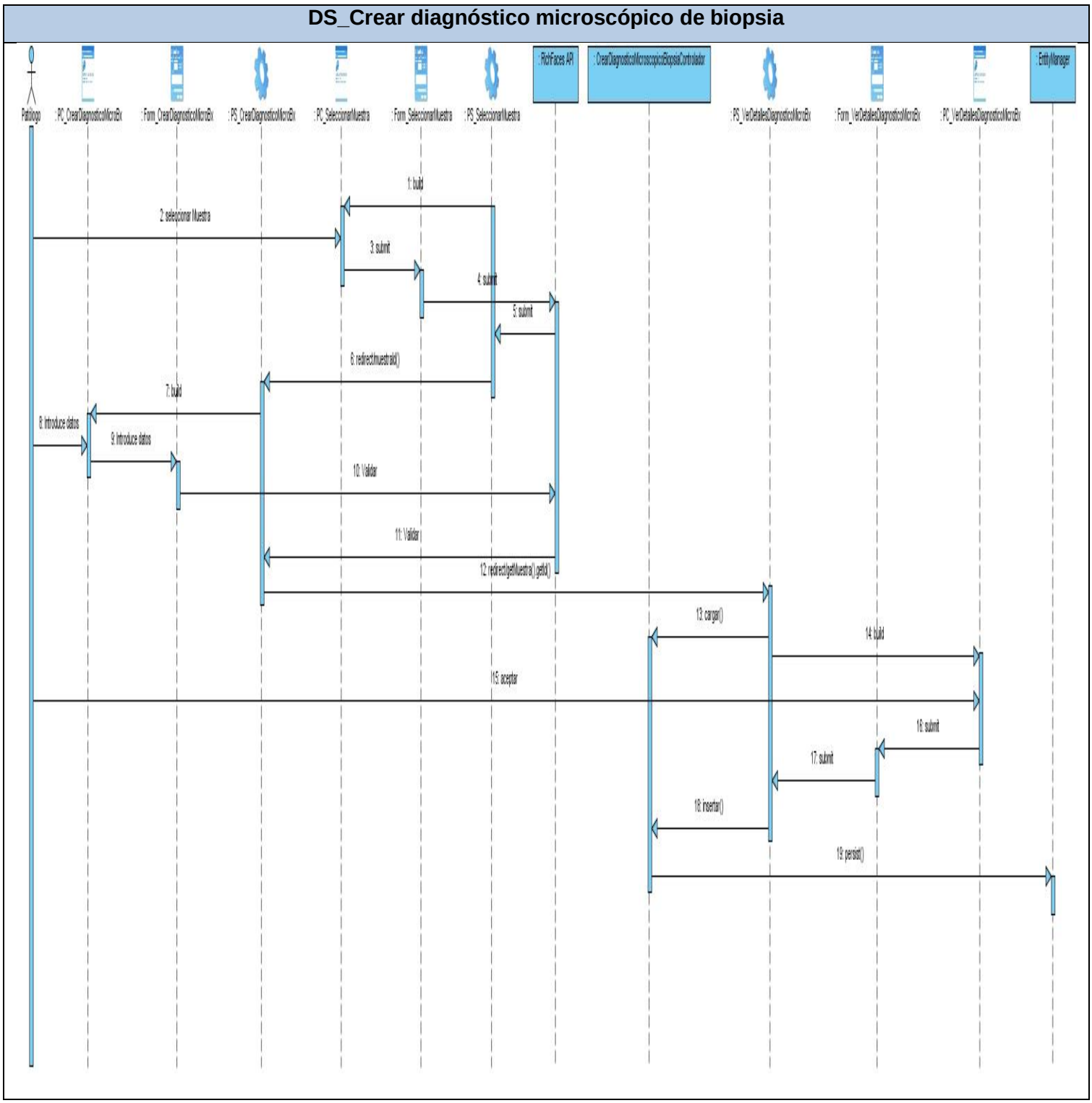

Figura 3.5. DS Crear diagnóstico microscópico de biopsia.

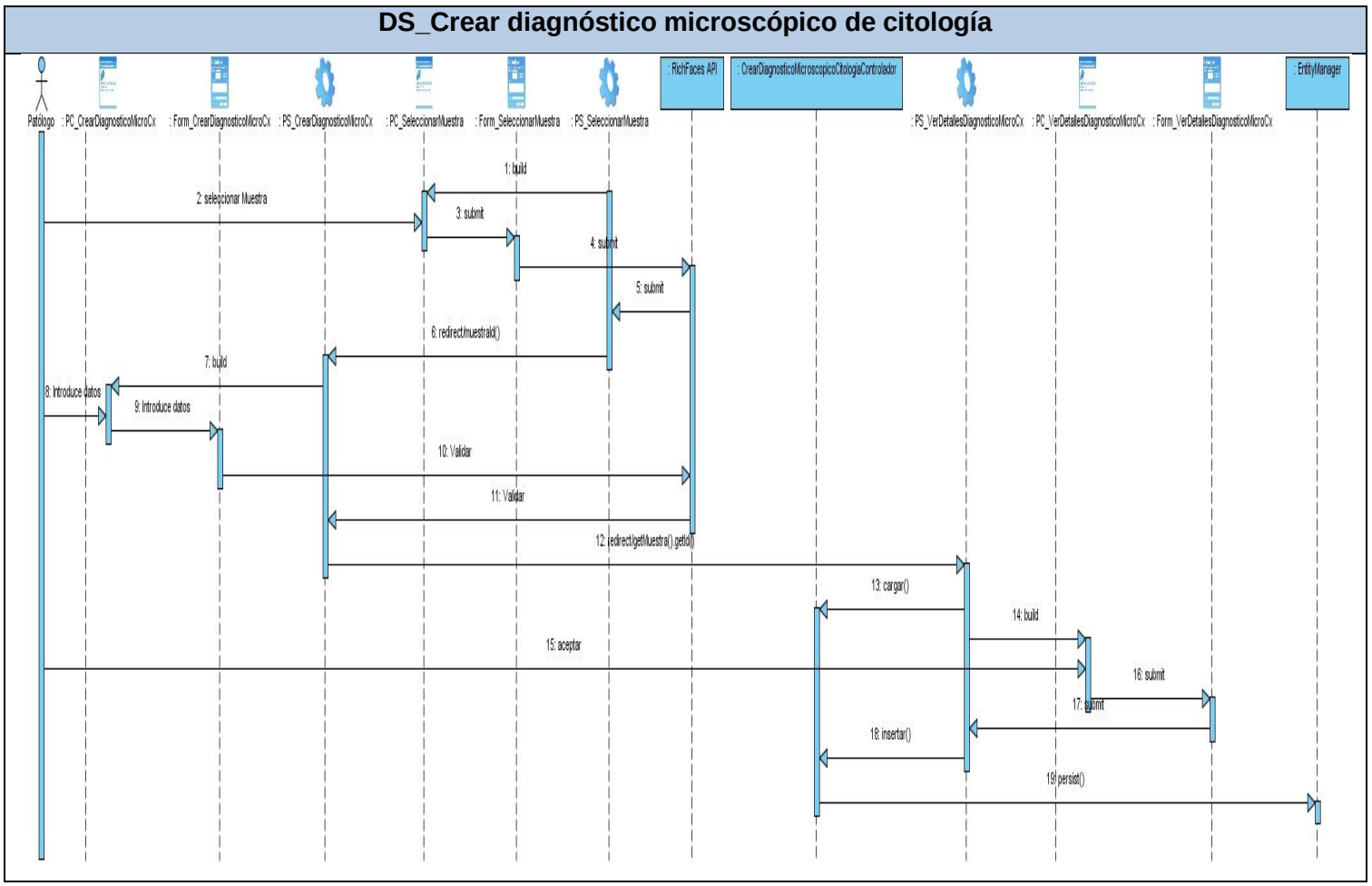

Figura 3.6. DS Crear diagnóstico microscópico de citología.

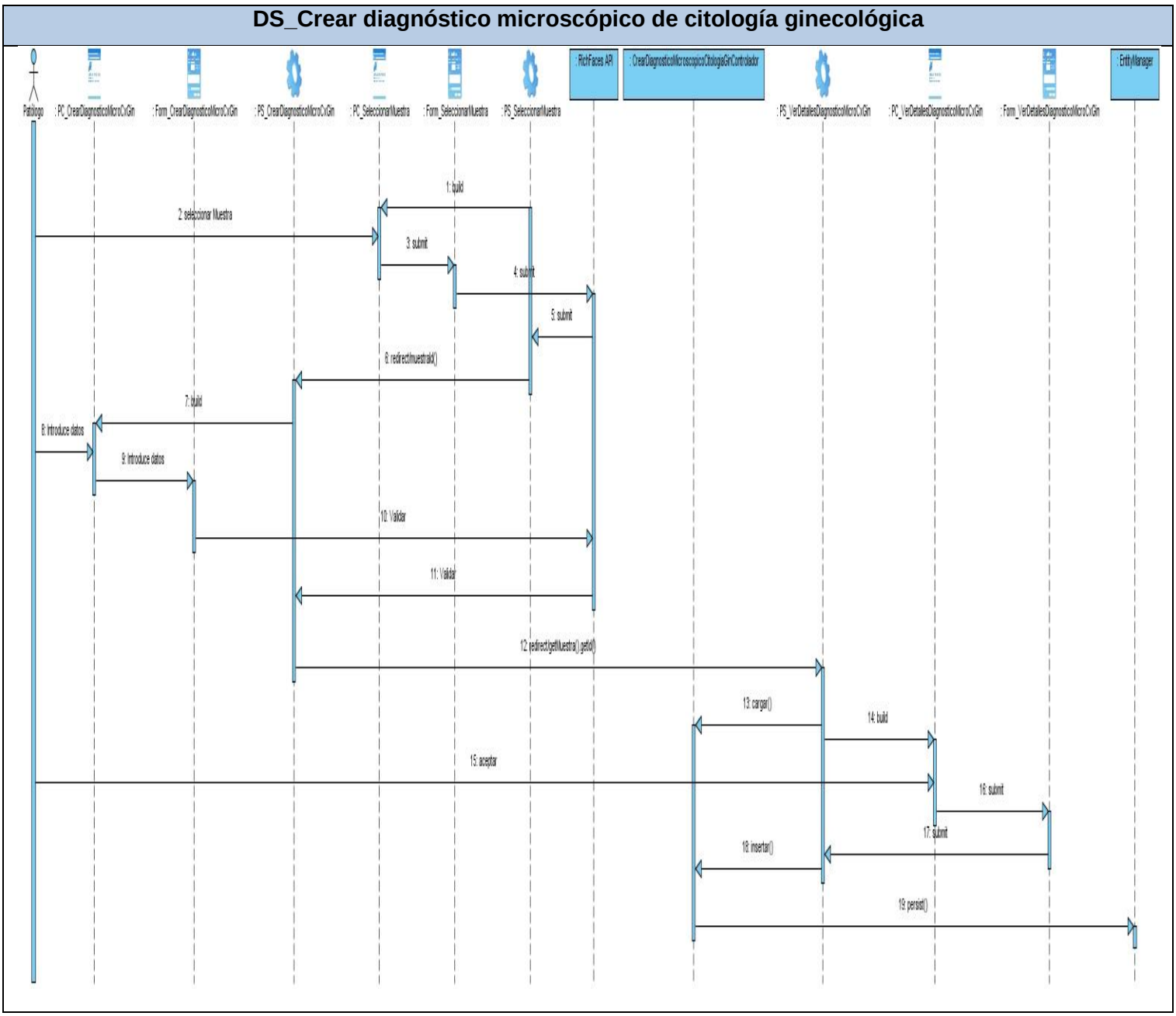

Figura 3.7. DS Crear diagnóstico microscópico de citología ginecológica.

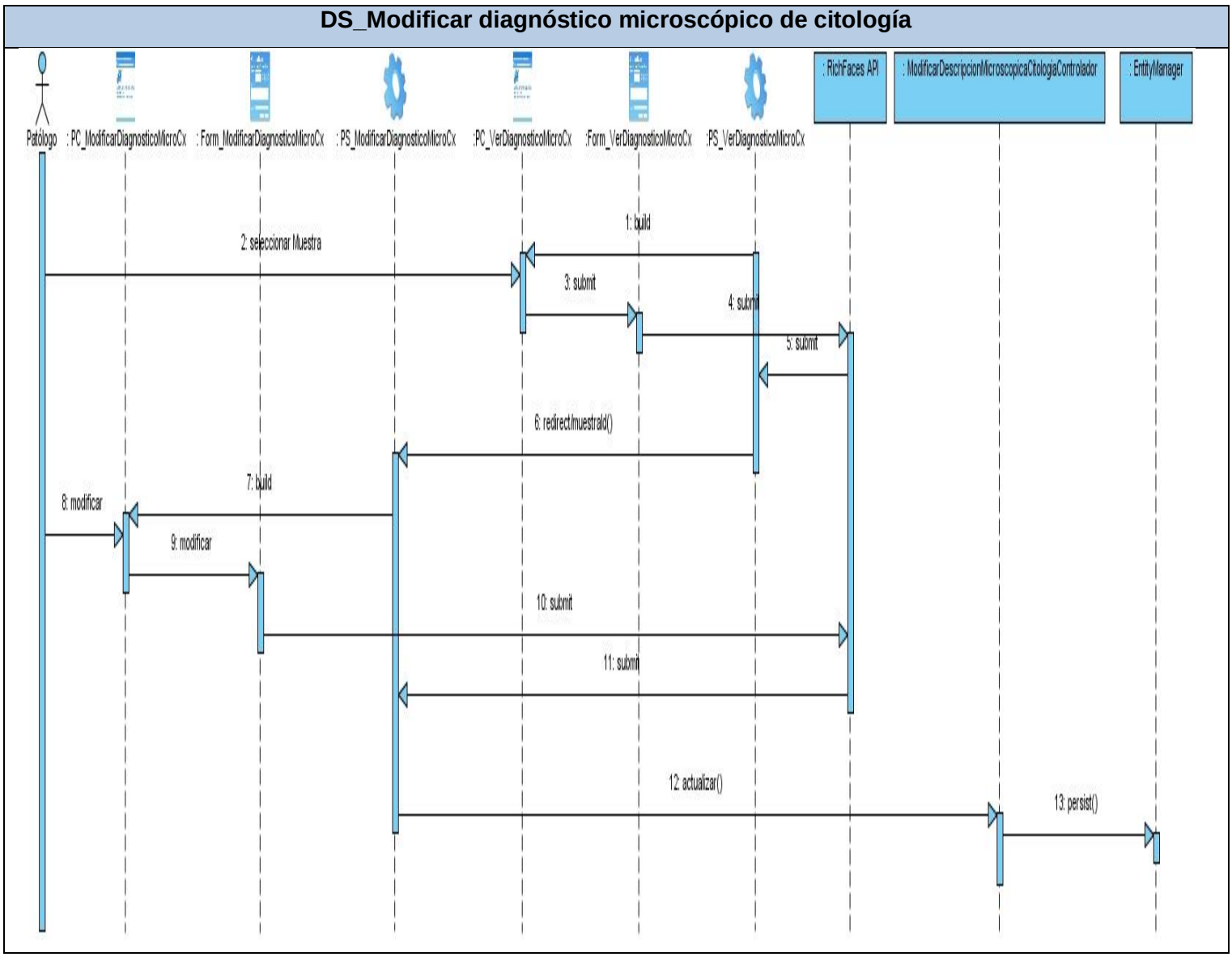

Figura 3.8. DS Modificar diagnóstico microscópico de citología.

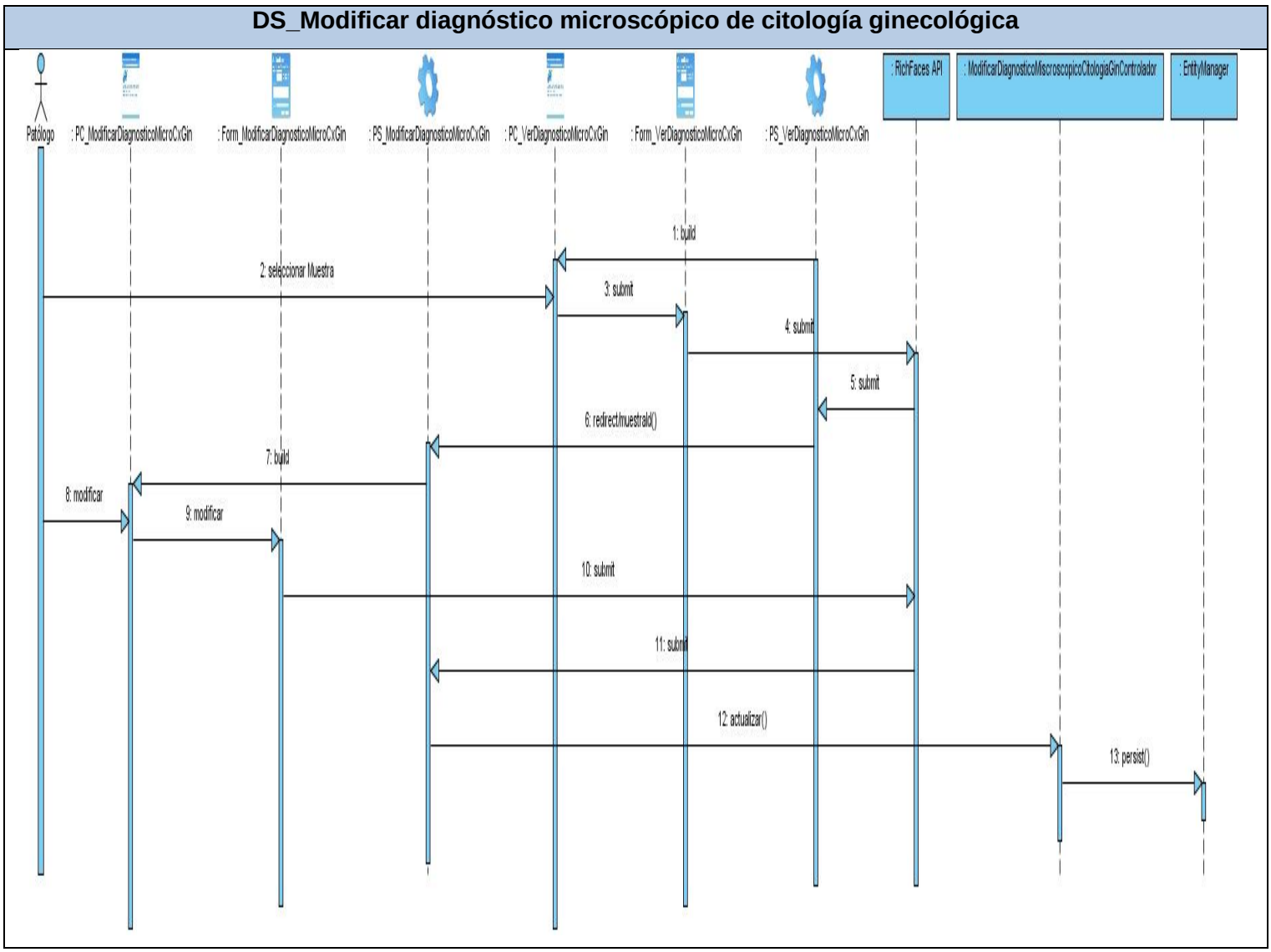

Figura 3.9. DS Modificar diagnóstico microscópico de citología ginecológica.

3.2 *Descripción de las nuevas clases u operaciones necesarias*

Nombre: CieCustomList	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	cambiarBusqueda():void
Descripción:	Cambia el tipo de búsqueda de los diagnósticos del CIE
Nombre:	obtenerTiposCie(): List<String>
Descripción:	Obtiene en una lista de la estructura que presentan los diagnósticos del CIE

Tabla 3.1 Inicializa toda la estructura de los diagnósticos del CIE

Nombre: SeleccionarCieControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	seleccionarEnfermedad():void
Descripción:	Permite seleccionar los diagnósticos del CIE
Nombre:	eliminarEnfermedad(): void
Descripción:	Permite eliminar los diagnósticos del CIE que fueron seleccionados
Nombre:	cargarDiagnosticoAnterior(): void
Descripción:	Carga los diagnósticos del CIE del diagnóstico microscópico de biopsia

Tabla 3.2 Cargar diagnósticos del CIE (Diagnóstico microscópico de biopsia)

Nombre: CodigoBethesdaList	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	CodigoBethesdaList()
Descripción:	Inicializa los datos del Código de Bethesda

Tabla 3.3 Inicializa el listado de los diagnósticos del Sistema de Bethesda

Nombre: SeleccionarBethesdaControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	seleccionarEnfermedad():void
Descripción:	Permite seleccionar los diagnósticos del Código de Bethesda
Nombre:	eliminarEnfermedad(): void
Descripción:	Permite eliminar los diagnósticos del Código de Bethesda que fueron seleccionados
Nombre:	cargarDiagnosticoAnteriorMicroCitologia(): void
Descripción:	Carga los diagnósticos del Código de Bethesda del diagnóstico microscópico de citología

Tabla 3.4 Cargar diagnóstico de Bethesda (Diagnóstico microscópico de citología)

3.3 *Modelo de datos*

Los modelos de datos son la base para las técnicas y herramientas empleadas en el uso y desarrollo de los sistemas de información. Es la representación física de las tablas de la base de datos, los mismos, contienen un conjunto de operaciones básicas para la realización de consultas y actualizaciones de los datos. Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo.

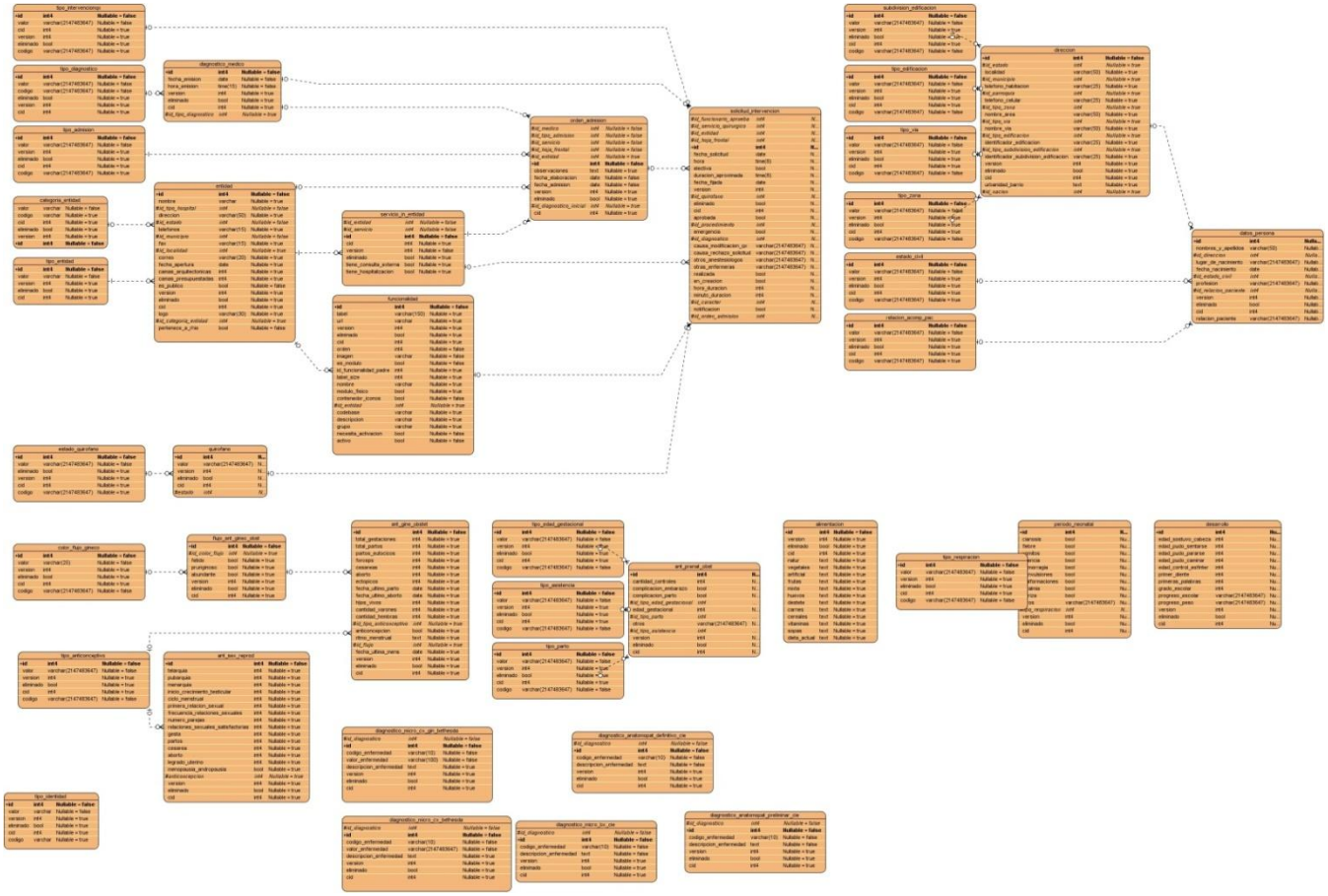

Figura 3.10. Modelo de datos

3.4 *Valoración de las técnicas de validación*

En el desarrollo de aplicaciones es necesario realizar validaciones de las entradas del usuario. El sistema propuesto utiliza para resolver este problema el framework JSF, el cual permite definir las validaciones en la vista, aunque tiene como inconveniente de que se repiten las validaciones una y otra vez. Para solucionar este problema se utiliza Hibernate Validator, al definirse las validaciones en los objetos del negocio y de esta forma, las validaciones pueden ser llamadas en el momento necesario.

Descripción de las tablas

Nombre: diagnosticoMicroBxCie		
Descripción: Tabla para registrar los datos pertenecientes a los diagnósticos del CIE asociados a un diagnóstico microscópico de biopsia		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
diagnosticoMicroBx	integer	Identificador del diagnóstico microscópico de biopsia
codigoEnfermedad	varchar	Código de el diagnóstico del CIE
descripcionEnfermedad	varchar	Descripción de el diagnóstico del CIE
version	integer	Versión
eliminado	boolean	Si fue eliminado o no
cid	integer	Identificador de la bitácora de sucesos

Tabla 3.5 Crear los diagnósticos del CIE a un diagnóstico microscópico de biopsia

Nombre: diagnosticoAnatomopatDefinitivoCie		
Descripción: Tabla para registrar los datos pertenecientes a los diagnósticos del CIE asociados a un diagnóstico anatomopatológico definitivo de autopsia		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
diagnosticoAnatomopatDefinitivo	integer	Identificador del diagnóstico anatomopatológicos definitivo de autopsia

codigoEnfermedad	varchar	Código de el diagnóstico del CIE
descripcionEnfermedad	varchar	Descripción de el diagnóstico del CIE
version	integer	Versión
eliminado	boolean	Si fue eliminado o no
cid	integer	Identificador de la bitácora de sucesos

Tabla 3.6 Registrar los diagnósticos del CIE vinculados a un diagnóstico de autopsia

Nombre: diagnosticoMicroCxBethesda		
Descripción: Tabla para registrar los datos pertenecientes a los diagnósticos del código de Bethesda asociados a un diagnóstico microscópico de citología		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
diagnosticoMicroCx	integer	Identificador del diagnóstico microscópico de citología
codigoEnfermedad	varchar	Código de el diagnóstico del Código de Bethesda
descripcionEnfermedad	varchar	Descripción de el diagnóstico del Código de Bethesda
valorEnfermedad	varchar	Valor de el diagnóstico del Código de Bethesda
version	integer	Versión

eliminado	boolean	Si fue eliminado o no
cid	integer	Identificador de la bitácora de sucesos

Tabla 3.7 Crear los diagnósticos de Bethesda a un diagnóstico microscópico de citología

3.5 *Vista de implementación*

En esta vista es donde se realiza una descripción del sistema en capas y subsistemas de implementación, y donde se describe la forma de realizar la asignación de paquetes y clases que se encuentran en la vista lógica a los paquetes y módulos que pertenecen a la implementación. Uno de sus artefactos es el diagrama de componentes.

- ✓ Diagrama de componentes: Muestran un conjunto de componentes y sus relaciones, modelan los aspectos físicos de un sistema y los elementos físicos que residen en un nodo tales como: librerías, ejecutables, tablas, archivos y documentos. Estos diagramas están compuestos por: componentes; interfaces y relaciones de dependencia, generalización, asociación, realización.

A continuación se muestra la distribución de los componentes en el sistema propuesto.

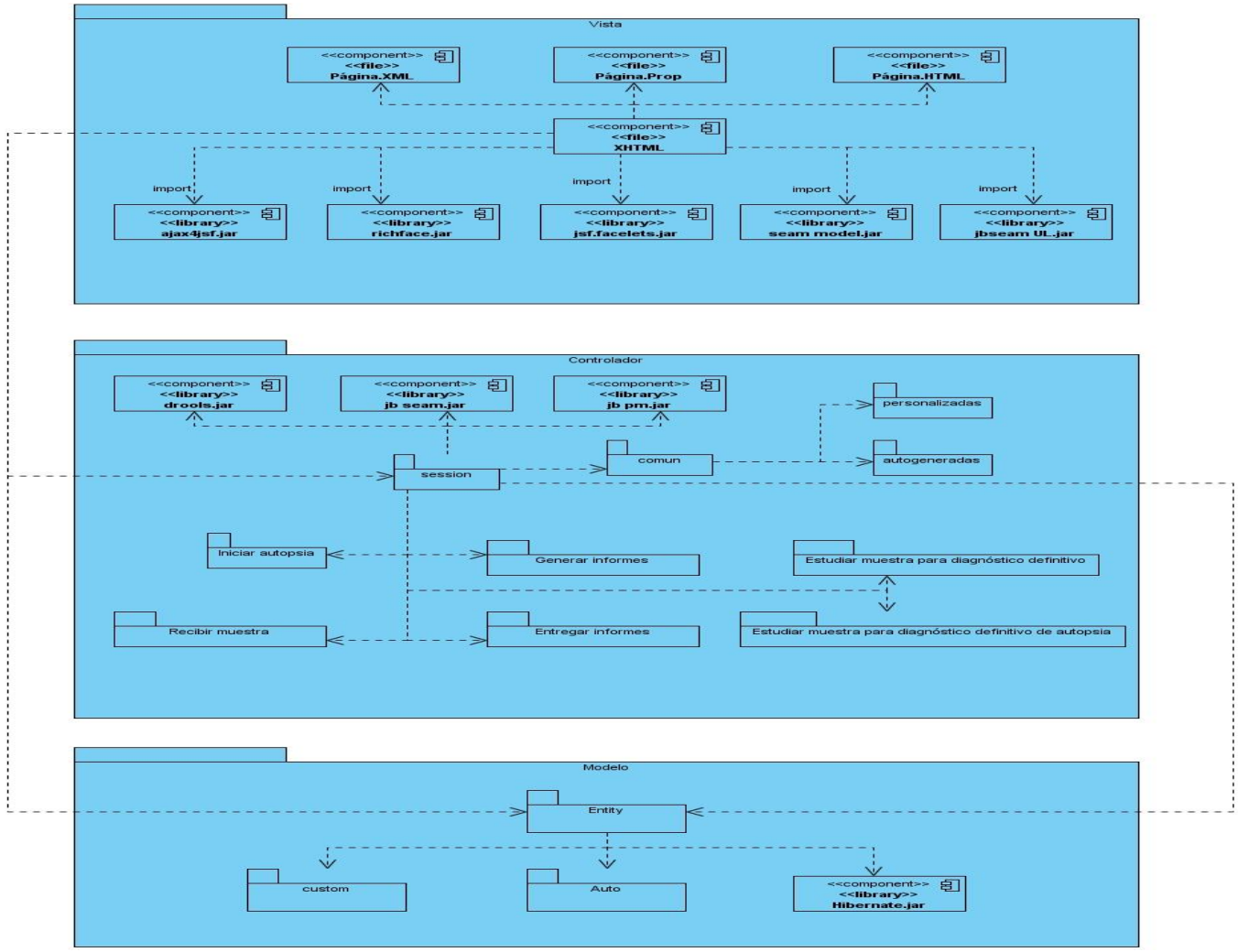

Figura 3.11. Diagrama de componentes

En el presente capítulo se realizó una valoración del diseño y la estructura de esta propuesta por el analista del sistema, describiéndose los casos de usos y sus correspondientes clases. Se realizó además una breve descripción de los atributos y métodos de cada clase y se obtuvo el modelo de datos del sistema, la descripción de las tablas con las que interactúa el módulo y la vista de Implementación.

Capítulo4 Modelo de pruebas

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

4.1 *Prueba de caja negra*

La prueba de caja negra, conocida también como pruebas de entrada-salida, pruebas de caja opaca, pruebas inducidas por los datos o pruebas funcionales se utiliza para representar a los sistemas cuando no se conoce que elementos componen al sistema o proceso, pero se conoce a qué determinadas entradas corresponden determinadas salidas y con ello poder inducir, presumiendo que a determinados estímulos, las variables funcionarán en cierto sentido.

Las pruebas de caja negra se centran en lo que se espera del módulo, es decir, buscan encontrar casos en que el módulo no se atiende a su especificación. Es por esto que son denominadas pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo internamente el módulo.

Esta prueba está centrada principalmente en los requisitos funcionales del software, esto permite la obtención de una serie de condiciones de entrada que ejerciten estos requisitos y se ignora la estructura de control. No son más que las pruebas que se realizan sobre la interfaz del software.

Para el desarrollo de dicha prueba se utilizan varias técnicas, entre ellas se encuentran:

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

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

Un caso de prueba es un conjunto de entradas de pruebas donde se describen escenarios y se identifican variables, las cuales representan a un conjunto de estados válidos o inválidos para las condiciones de entrada para demostrar que las funcionalidades del software son operativas.

4.2 *Descripción de los casos de pruebas*

Caso de prueba: Crear diagnóstico microscópico de biopsia

Escenarios del Crear diagnóstico microscópico de biopsia	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Crear diagnóstico microscópico de biopsia	Crear el diagnóstico microscópico de biopsia satisfactoriamente.	Se selecciona la opción Crear diagnóstico microscópico de biopsia. Se selecciona la muestra correspondiente. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se regresa a la vista anterior listar muestra.
EC 2: Cancelar operación	Cancelar la opción de Crear diagnóstico microscópico de biopsia.	Se selecciona la opción Crear diagnóstico microscópico de biopsia. Se selecciona la muestra correspondiente. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior listar muestra.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de existir incompletos muestra un indicador sobre los campos incompletos.	Se selecciona la opción Crear diagnóstico microscópico de biopsia. Se introducen los datos incompletos. Se selecciona la opción Aceptar.

		Muestra un indicador sobre los campos incompletos.

Tabla 4.1 CP Crear diagnóstico microscópico de biopsia.

Id del escenario	EC 1	EC 2	EC 3
Escenario	Crear diagnóstico microscópico de biopsia	Cancelar operación	Existen datos incompletos
Variable 1 (Diagnósticos del CIE)	V		I
Variable 2 (Observaciones)	V		V
Botón 1 (Aceptar)	NA		NA
Botón 2 (Cancelar)		NA	
Respuesta del Sistema	Crear el diagnóstico microscópico de biopsia y regresa a la vista anterior listar muestra.	Regresa a la vista anterior de listar muestra.	Muestra un indicador sobre los campos incompletos.
Resultado de la Prueba			

Tabla 4.2 SC Crear diagnóstico microscópico de biopsia.

Caso de prueba: Crear diagnóstico microscópico de citología

Escenarios del Crear diagnóstico microscópico de citología	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Crear diagnóstico microscópico de citología	Crear diagnóstico microscópico de citología satisfactoriamente.	Se selecciona la opción Crear diagnóstico microscópico de citología. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se regresa a la vista anterior listar muestra.
EC 2: Cancelar operación	Cancelar la opción de Crear diagnóstico microscópico de citología.	Se selecciona la opción Crear diagnóstico microscópico de citología. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior listar muestra.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de existir incompletos muestra un indicador sobre los campos incompletos.	Se selecciona la opción Crear diagnóstico microscópico de citología. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.

Tabla 4.3 CP Crear diagnóstico microscópico de citología.

Id del escenario	EC 1	EC 2	EC 3
Escenario	Crear diagnóstico microscópico de citología	Cancelar operación	Existen datos incompletos
Variable 1 (Responsable screening)	V		I
Variable 2 (Patólogo supervisor)	V		I
Variable 3 (Diagnóstico código de Bethesda)	V		I
Variable 4 (Observaciones)			
Variable 5 (Descripción)	V		I
Variable 6 (Fecha Screening)			
Botón 1 (Aceptar)	NA		NA
Botón 2 (Cancelar)		NA	
Respuesta del Sistema	Crear diagnóstico microscópico de citología y regresa a la vista anterior listar muestra.	Regresa a la vista anterior de listar muestra.	Muestra un indicador sobre los campos incompletos.
Resultado de la Prueba			

Tabla 4.4 SC Crear diagnóstico microscópico de citología.

Caso de prueba: Crear diagnóstico anatomopatológicos preliminar

Escenarios del Crear diagnóstico anatomopatológicos preliminar	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Crear diagnóstico anatomopatológicos preliminar.	Crear el diagnóstico anatomopatológicos preliminar satisfactoriamente.	Se selecciona la opción Crear diagnóstico anatomopatológicos preliminar. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se regresa a la vista anterior listar muestra.
EC 2: Cancelar operación	Cancelar la opción de Crear anatomopatológicos preliminar satisfactoriamente.	Se selecciona la opción Crear anatomopatológicos preliminar satisfactoriamente. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior listar muestra.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de existir incompletos muestra un indicador sobre los campos incompletos.	Se selecciona la opción Crear diagnóstico anatomopatológicos preliminar. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.

Tabla 4.5 CP Crear diagnóstico anatomopatológicos preliminar.

Id del escenario	EC 1	EC 2	EC 3
Escenario	Crear diagnóstico anatomopatológicos preliminar	Cancelar operación	Existen datos incompletos
Variable 1 (Diagnóstico CIE)	V		I
Botón 1 (Aceptar)	NA		NA
Botón 2 (Cancelar)		NA	
Respuesta del Sistema	Crear diagnóstico anatomopatológicos preliminar y regresa a la vista anterior listar muestra.	Regresa a la vista anterior de listar muestra.	Muestra un indicador sobre los campos incompletos.
Resultado de la Prueba			

Tabla 4.6 SC Crear diagnóstico anatomopatológicos preliminar.

En este capítulo se realizó el modelo de prueba. Utilizando el método de 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 guías para las pruebas efectuadas.

Conclusiones

En correspondencia con los objetivos trazados en la investigación de este trabajo, se concluye:

- ✓ El análisis de los sistemas estudiados, evidenció que no comprenden todas las funcionalidades requeridas para la obtención de los diagnósticos finales de los estudios del módulo Anatomía Patológica del Sistema de Información Hospitalaria alas HIS.
- ✓ Las herramientas, tecnologías y estándares utilizados, hicieron posible la construcción de un sistema robusto y flexible.
- ✓ La utilización de las pautas de diseño aplicadas permitieron lograr uniformidad y homogeneidad en las interfaces visuales obtenidas.
- ✓ La implementación de los estándares definidos posibilitó la obtención estructurada de los informes finales, lo cual facilitará su gestión en esta área.

Referencias bibliográficas

1. Dr. Antonio Cerritos. Sistema de Información Hospitalaria. 2003 [Citado el: 20 de Octubre del 2010] http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf.

2. Infomed. IV Congreso virtual hispanoamericano de Anatomía Patológica. 2004 [Citado el: 20 de Octubre del 2010] http://conganat.uninet.edu/6CVHAP/autores/trabajos/T067/index.html.

3. Infomed. Biblioteca virtual en salud. 2010 [Citado el: 20 de Octubre del 2010] http://bvs.sld.cu/revistas/mil/vol24_2_95/mil10295.htm.

4. Pat-Win. 2010 [Citado el: 25 de Octubre del 2010] http://www.ittrade.com.mx/pdf/PATwin_080108.pdf.

5. NovoPath. 2010 [Citado el: 25 de Octubre del 2010] http://www.vitroweb.com/documents/VSF/VSF-NP0010_OTR.pdf.

6. CIE. 2010 [Citado el: 26 de Octubre del 2010] http://sssalud.gov.ar/hospitales/archivos/cie_10_revi.pdf.

7. National Library of Medicine. SNOMED Clinical Terms® To Be Added To UMLS® Metathesaurus®. 2003. http://www.nlm.nih.gov/research/umls/Snomed/snomed_announcement.html.

8. Bethesda. 2010 [Citado el: 26 de Octubre del 2010] http://www.medicalcriteria.com/es/criterios/gin_bethesda_es.htm.

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

10. UI Develoment with JaveServer Faces.

11. Ajax4jsf Developer Guide. 2007.

12. JBoss RichFaces. [En línea] 2009. [Citado el: 20 de Octubre del 2010] http://labs.jboss.com/portal/jbossrichfaces/.

13. Allen, Dan. Seam in Action. s.l.: Manning Publication, 2008.

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

15. Cristian Bauer, Gavin King. Java Persistence with Hibernate. 2005. 1-932394-88-5.

16. Pressman, Roger S. Ingeniería de software, un enfoque práctico. 2005.

17. Larman, Craig. UML y Patrones. Introdución al análisis y diseño. 2004.

18. Ingeniería de software, un enfoque práctico. Pressman, Roger S.2005

.

Bibliografía

1. Dr. Antonio Cerritos. Sistema de Información Hospitalaria. 2003 [Citado el: 20 de Octubre del 2010] http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf.

2. Infomed. IV Congreso virtual hispanoamericano de Anatomía Patológica. 2004 [Citado el: 20 de Octubre del 2010] http://conganat.uninet.edu/6CVHAP/autores/trabajos/T067/index.html.

3. Infomed. Biblioteca virtual en salud. 2010 [Citado el: 20 de Octubre del 2010] http://bvs.sld.cu/revistas/mil/vol24_2_95/mil10295.htm.

4. Pat-Win. 2010 [Citado el: 25 de Octubre del 2010] http://www.ittrade.com.mx/pdf/PATwin_080108.pdf.

5. NovoPath. 2010 [Citado el: 25 de Octubre del 2010] http://www.vitroweb.com/documents/VSF/VSF-NP0010_OTR.pdf.

6. CIE. 2010 [Citado el: 26 de Octubre del 2010] http://sssalud.gov.ar/hospitales/archivos/cie_10_revi.pdf.

7. National Library of Medicine. SNOMED Clinical Terms® To Be Added To UMLS® Metathesaurus®. 2003. http://www.nlm.nih.gov/research/umls/Snomed/snomed_announcement.html.

8. Bethesda. 2010 [Citado el: 26 de Octubre del 2010] http://www.medicalcriteria.com/es/criterios/gin_bethesda_es.htm.

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

10. UI Develoment with JaveServer Faces.

11. Ajax4jsf Developer Guide. 2007.

12. JBoss RichFaces. [En línea] 2009. [Citado el: 20 de Octubre del 2010] http://labs.jboss.com/portal/jbossrichfaces/.

13. Allen, Dan. Seam in Action. s.l.: Manning Publication, 2008.

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

15. Cristian Bauer, Gavin King. Java Persistence with Hibernate. 2005. 1-932394-88-5.

16. Pressman, Roger S. Ingeniería de software, un enfoque práctico. 2005.

17. Larman, Craig. UML y Patrones. Introdución al análisis y diseño. 2004.

18. Ingeniería de software, un enfoque práctico. Pressman, Roger S.2005

19. Milán Cristo, Nadiezka. Resumen por procesos del módulo Anatomía Patológica. UCI: s.n., 2008.

20. Red Hat. RichFaces developer Guide. 2007.

21. The postgreSQL Global Development Group. PostgreSQL 8.3 Documentation. 2008.

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

23. Milán Cristo, Nadiezka. Resumen por procesos del módulo Anatomía Patológica. UCI: s.n., 2008.

24. The postgreSQL Global Development Group. PostgreSQL 8.3 Documentation. 2008.

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

Glosario de términos

AJAX: Técnica de desarrollo web para crear aplicaciones interactivas.

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

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

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

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

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

DDL: Scripts de creación de la base de datos.

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

ICEFaces: Biblioteca de componentes JSF Ajax.

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

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

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

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

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

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

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

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

Anexos

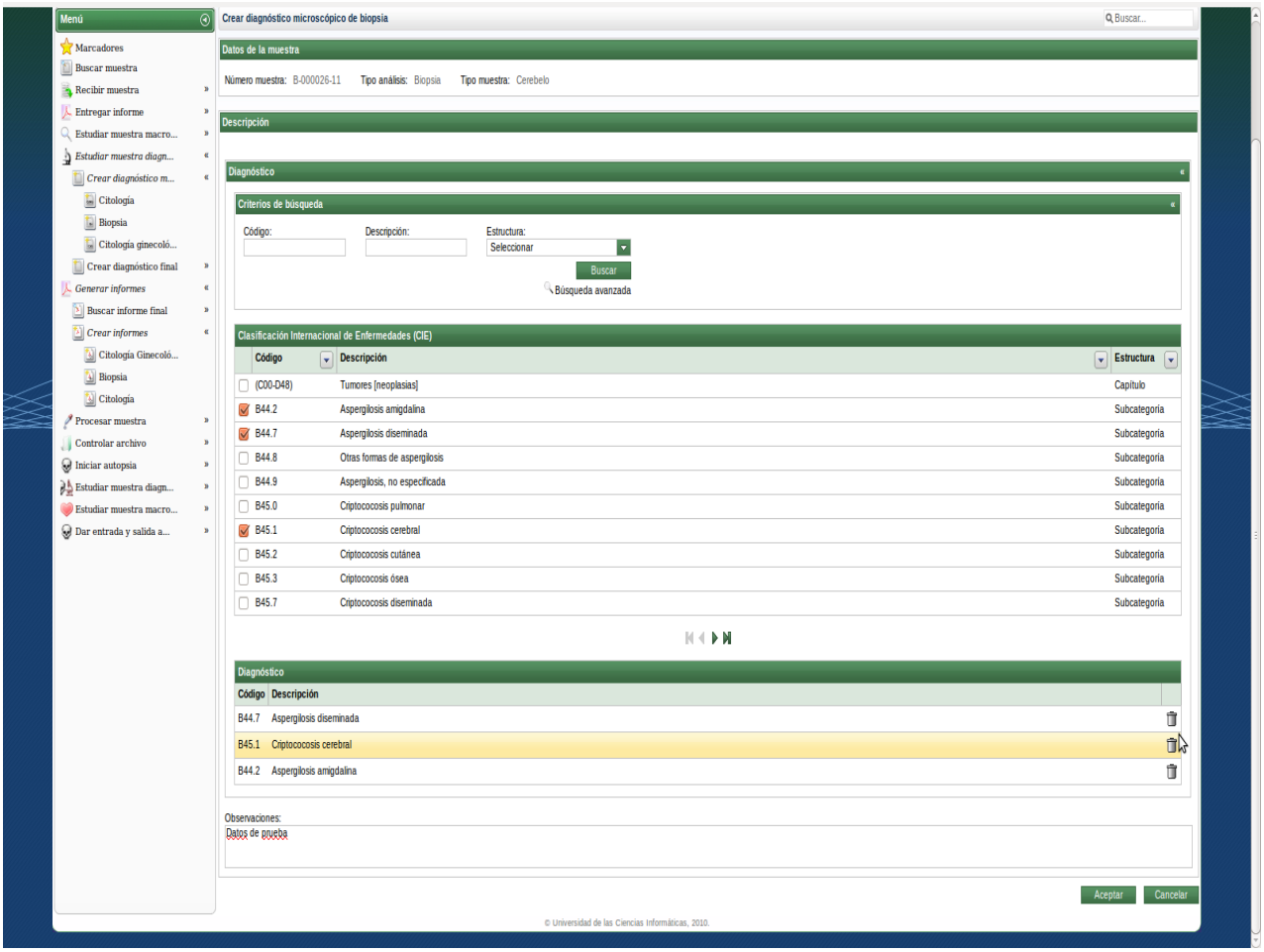

Figura 3.12. Figura: Crear diagnóstico microscópico de biopsia.

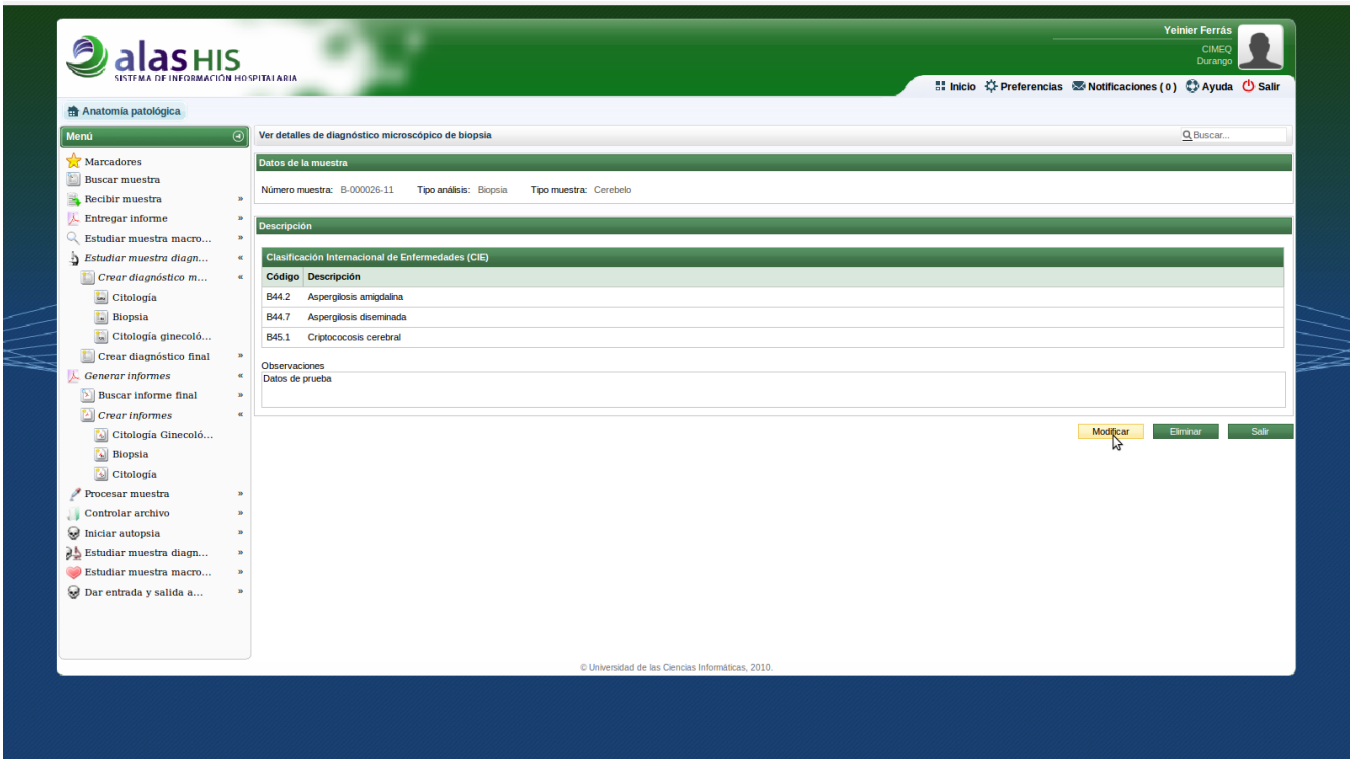

Figura 3.13. Figura: Ver detalles del diagnóstico microscópico de biopsia

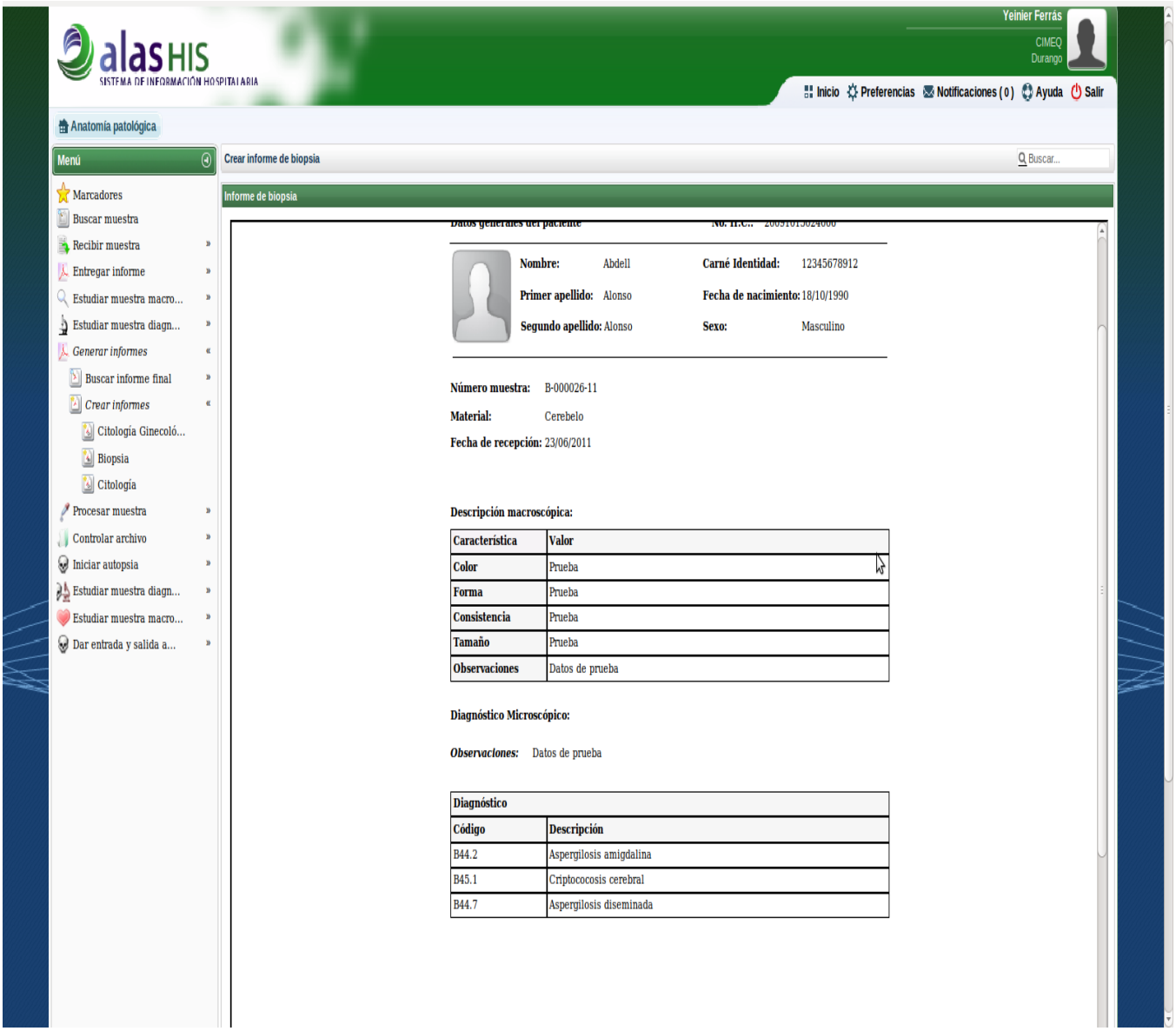

Figura 3.14. Figura: Crear informe de biopsia

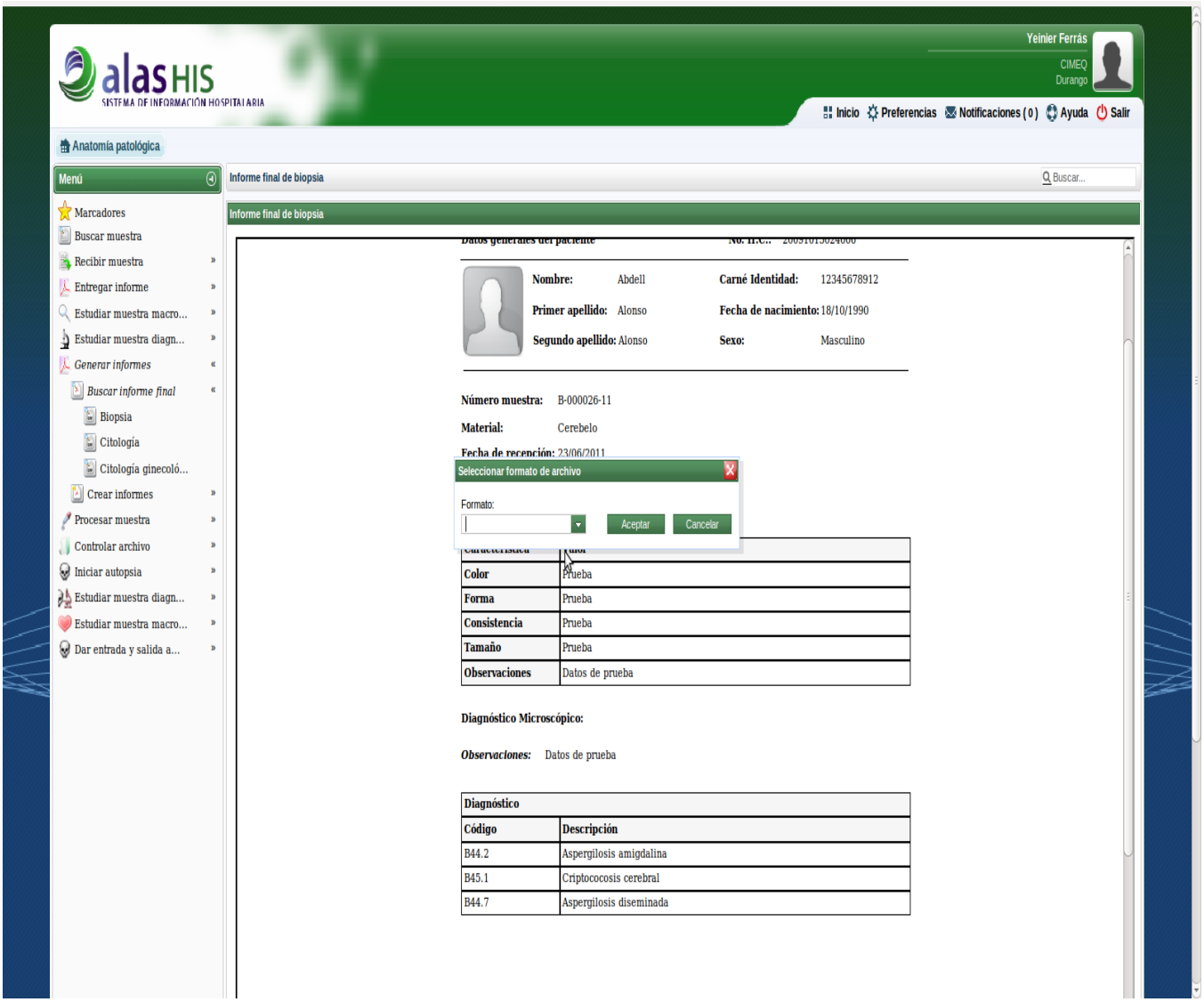

Figura 3.15. Figura: Informe final de biopsia