Universidad de las Ciencias Informáticas

Facultad 7

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

Título: Diseño e implementación de los procesos de Apoyo a la vigilancia epidemiológica del Sistema de Información Hospitalaria alas HIS

Autores: Arletis Rojas González

Rudny Rangel Marrero

Tutores: Ing. Omar García Bonelly

MSc. Maykell Sánchez Romero

Ciudad de La Habana, julio de 2010

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de julio del año 2010.

_____	_____
Arletis Rojas González	Rudny Rangel Marrero
Autora	Autor
_____	_____
MSc. Maykell Sánchez Romero	Ing. Omar García Bonelly
Tutor	Tutor

No pretendamos que las cosas cambien si siempre hacemos lo mismo. La crisis es la mejor bendición que puede sucederle a personas y países porque la crisis trae progresos. La creatividad nace de la angustia como el día nace de la noche oscura. Es en la crisis que nace la inventiva, los descubrimientos y las grandes estrategias. Quien supera la crisis se supera a sí mismo sin quedar "superado".

Quien atribuye a la crisis sus fracasos y penurias violenta su propio talento y respeta más a los problemas que a las soluciones. La verdadera crisis es la crisis de la incompetencia. El problema de las personas y los países es la pereza para encontrar las salidas y las soluciones. Sin crisis no hay desafíos, sin desafíos la vida es una rutina, una lenta agonía. Sin crisis no hay méritos. Es en la crisis donde aflora lo mejor de cada uno, porque sin crisis todo viento es caricia.

Hablar de crisis es promoverla, y callar en la crisis es exaltar el conformismo. En vez de esto trabajemos duro. Acabemos de una vez con la única crisis amenazadora que es la tragedia de no querer luchar por superarla.

Albert Einstein

DATOS DE CONTACTO

MSc. Maykell Sánchez Romero

Graduado de Ciencias de la Computación. Posee la categoría docente de Asistente. Tiene 1 año de experiencia en el tema y 5 de graduado.

Correo electrónico: maykell@uci.cu

Ing. Omar García Bonelly

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: obonelly@uci.cu

RESUMEN

La unidad de vigilancia epidemiológica de un hospital es el área que se encarga de desarrollar las actividades necesarias para poder notificar situaciones epidemiológicas como epidemias, casos de brotes u otras situaciones de emergencia. Existen instituciones hospitalarias que no cuentan con un sistema informático que cubra todas las necesidades de esta área.

En el presente trabajo se continúa el desarrollo del módulo de Epidemiología a través de la implementación de los procesos de apoyo a la vigilancia del Sistema de Información Hospitalaria alas HIS

Para la realización del mismo se utiliza el proceso unificado racional (RUP). El patrón arquitectónico que se aplica es el Modelo-Vista-Controlador, el cual proporciona una arquitectura bien definida aplicable al diseño e implementación de los procesos de apoyo a la vigilancia epidemiológica, que incluye la generación de reportes, las inspecciones sanitarias y las interconsultas.

Con la implementación de estas nuevas funcionalidades en el módulo Epidemiología del Sistema de Información Hospitalaria alas HIS, se logrará una mayor eficiencia al notificar alguna enfermedad de riesgo, además de facilitar la gestión y control de información de pacientes e instituciones con riesgos epidemiológicos.

PALABRAS CLAVES:

Vigilancia epidemiológica, epidemias, casos de brotes, epidemiología, inspección sanitaria, interconsulta.

TABLA DE CONTENIDO

INTRODUCCIÓN 6

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 11

1.1 Conceptos básicos relacionados con el problema planteado 11

1.2 Sistemas automatizados existentes 12

1.3 Herramientas y tecnologías de desarrollo 15

1.4 Lenguaje de programación 20

1.5 Metodologías de desarrollo 22

CAPÍTULO 2: DESCRIPCIÓN DE LA ARQUITECTURA 26

2.1 Requisitos no funcionales 26

2.2 Descripción de la arquitectura. 29

2.3 Posibles implementaciones de componentes o módulos que puedan ser reutiliza-dos. Estrategias de integración. 30

2.4 Seguridad 31

2.5 Vista de Despliegue 32

2.6 Estrategias de codificación. Estándares y estilos a utilizar. 33

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA 40

3.1 Valoración crítica del diseño propuesto por el analista. 40

3.2 Descripción de las nuevas clases u operaciones necesarias. 49

3.3 Modelo de datos 56

3.4 Breve valoración de las técnicas de validación (Integridad y Normalización de la Base de datos). 63

3.5 Vista de Implementación. (Diagrama de Componentes) 64

CAPÍTULO 4: MODELO DE PRUEBA 66

4.1 Prueba de caja negra 67

CONCLUSIONES 70

RECOMENDACIONES 71

REFERENCIAS BIBLIOGRÁFICAS 72

BIBLIOGRAFÍA 74

INTRODUCCIÓN

La humanidad siempre ha necesitado darle soluciones a problemas surgidos a lo largo de su evolución. Durante todos estos años el hombre ha creado diferentes metodologías, directivas y sistemas que lo han ayudado a resolver los problemas o al menos agilizar su solución, logrando un resultado final de mayor calidad.

Con el surgimiento de las computadoras y el uso masivo de las mismas a partir de la segunda mitad del siglo XX y la creación de sistemas capaces de almacenar, calcular y transmitir información, es que el hombre encuentra una forma ideal de administrar los grandes volúmenes de datos y de esta forma aumentar la calidad de los servicios que se pueden brindar en cualquier empresa o instalación laboral, resolviéndose digitalmente problemáticas planteadas cuya solución manual necesitaría mucho más tiempo y esfuerzo.

Un paso de avance a nivel mundial, es el surgimiento de las Tecnologías de la Información y las Comunicaciones (TIC), que constituyen un conjunto de técnicas y dispositivos avanzados que integran funcionalidades de gestión y control de datos que se han desarrollado durante los últimos años en las distintas ramas de la sociedad, para lograr una mayor eficiencia en los servicios.

Actualmente existen sistemas informáticos en diferentes sectores de la sociedad, como son: la Cultura, el Deporte, la Ciencia, la Recreación, entre otros. Uno de los sectores donde es más necesaria la informatización, es la Salud, ya que a diario el personal que labora en este sector debe manipular grandes volúmenes de datos, lo cual ocasiona un gasto innecesario de tiempo y medios para lograr una buena atención al paciente. Dichos sistemas brindan incontables beneficios ya que mediante su uso, disminuyen los gastos por adquisición de materiales y aumentan la eficiencia de la instalación donde sea aplicado.

A nivel mundial, en el sector de la Salud existen sistemas informáticos no estandarizados que muestran, imprimen y guardan toda la información con la que se trabaja en un hospital, pero estas funcionalidades no cubren todas sus necesidades. Con el paso del tiempo estos sistemas informáticos han evolucionado aumentando sus funcionalidades, hasta convertirse en los denominados Sistemas de Información Hospitalaria, comúnmente conocidos por sus siglas en inglés HIS, los cuales se utilizan con el fin de informatizar y optimizar el flujo de actividades hospitalarias, incrementándose así la calidad de la atención al paciente. Dichos sistemas de información y gestión hospitalaria permiten el almacenamiento,

procesamiento y el análisis de toda la información perteneciente a los pacientes y a los médicos que los atienden.

Estos sistemas tienen gran aceptación tanto por parte de los pacientes como de los trabajadores de esta esfera, pues se puede observar que su desarrollo en la gestión de información alivia grandemente el trabajo del personal médico, el cual de forma manual tarda demasiado en realizar todas las tareas pertinentes dentro de las instalaciones hospitalarias. La gestión y control de la información en las instalaciones hospitalarias se encuentra repartida en diferentes áreas, clasificadas según el tipo de información que manejan, para de esta forma intentar organizar todo el flujo de información dentro de la institución.

La unidad de vigilancia epidemiológica de un hospital es el área que se encarga de desarrollar y dar seguimiento a acciones pertinentes que permitan prevenir, detectar y actuar oportunamente ante la presencia de epidemias, casos de brotes y agentes epidemiológicos.

La vigilancia epidemiológica está constituida principalmente de cuatro elementos:

1. Un sistema de información, definido como el proceso sistemático de recolección, recuento y clasificación, presentación y análisis de los datos de enfermedades de riesgo en la población; que además cumpla con los requisitos básicos de calidad como: la integridad, la exactitud, la oportunidad y la comparabilidad.
2. Recursos humanos instruidos en epidemiología, capaces de estimar la magnitud que representa, detectar epidemias o subepidemias, documentar la distribución y propagación de los padecimientos, detectar y efectuar el seguimiento de casos y realizar el diagnóstico de laboratorio; con el objeto de ayudar a planificar, ejecutar y evaluar las acciones para la prevención y control del sida.
3. Laboratorios que garanticen la confirmación o descarte de diagnósticos de los padecimientos sujetos a vigilancia.
4. Un sistema de evaluación y supervisión con asesoría técnica en los distintos niveles técnico-administrativos.[1]

El personal que labora en dicha área de estas instituciones presenta problemas a la hora de emitir cierto número de reportes, algunos solicitados por la propia institución, por ejemplo el reporte estadístico sobre

la vigilancia de la mortalidad en mujeres en edad fértil, ya que para la realización del mismo se necesitan revisar los certificados de defunción, chequear que el paciente fuera de sexo femenino y estuviera en la edad fértil. Otro reporte como el de “Situaciones de Alerta y Epidemia”, solicitados por el propio sistema de salud nacional, es necesario para contabilizar situaciones de riesgo y/o alarma en la sociedad; requiere una búsqueda de datos de 7 años atrás referentes a la semana epidemiológica correspondiente con la del presente año.

Estas búsquedas se realizan en grandes estantes de información pertenecientes a diferentes áreas de la institución, según el período de tiempo solicitado del cual se quiera saber la situación estadística, lo cual conlleva a invertir mucho tiempo y esfuerzo en recopilar los datos necesarios para generar los mismos.

El proceso de Realizar Inspección Sanitaria, también perteneciente a esta área de Epidemiología, es el que se encarga de la vigilancia y control de todas las situaciones sanitarias dentro de la institución, ya sea enmarcado en que los productos de los almacenes o las farmacias no se encuentren en mal estado o además en notificar cualquier negligencia sanitaria dentro de algún local de la institución. El inspector sanitario se encarga de realizar una inspección en los diferentes locales basándose en: ambiente general, saneamiento ambiental y personal, dándole una calificación y generando un “Formato de Inspección” y en el caso que sea necesario, un “Formato de control de biológicos de farmacias” y/o una “Ficha de investigación y notificación de epizootias”.

Todas estas revisiones y evaluaciones son realizadas manualmente por los inspectores sanitarios de dicha área, por lo que en ocasiones no se notifican a tiempo las negligencias detectadas; así como no se cuentan con los datos puntuales de incidencias anteriores para verificar si fueron erradicadas.

En esta área también se reciben solicitudes para realizar una Interconsulta, estas se realizan por teléfono o personalmente, aumentando así el tiempo en que las mismas son notificadas. El epidemiólogo brinda el servicio solicitado para valorar al paciente y su situación actual. Luego se realiza un resumen médico tomando en cuenta el estado del paciente y los antecedentes personales y epidemiológicos de donde se generan los comentarios o sugerencias, según el caso. Estas sugerencias e interconsulta carecen en su mayoría de los antecedentes mencionados al no existir registros anteriores de los pacientes.

Debido a los grandes volúmenes de información a procesar de forma manual y la inconsistencia de la misma en algunos casos por su deterioro, al llevar a cabo las búsquedas de información para realizar dichas actividades, se demuestra que la velocidad de generación y entrega en tiempo de dichos reportes

y/o formatos de inspección e interconsultas dificulta la efectividad de las acciones que tienen lugar en el área de Epidemiología de las instituciones hospitalarias.

Hasta el momento no existe en estas instituciones un sistema organizado capaz de gestionar toda la información referente a los procesos de Generar Reportes, Realizar Inspección Sanitaria e Interconsultas en el área de Epidemiología.

A partir del análisis de la situación problémica, se define el siguiente **problema a resolver**: ¿Cómo mejorar los procesos de apoyo a la vigilancia que tienen lugar en el área de Epidemiología de las instituciones hospitalarias?

Este problema tiene como **objeto de estudio** los procesos de gestión de información en el área de Epidemiología de las instituciones hospitalarias. De manera que el **campo de acción** son los procesos de apoyo a la vigilancia que tienen lugar en el área de Epidemiología de las instituciones hospitalarias.

Basado en esta idea se define como **objetivo general** del presente trabajo: Realizar la implementación de los procesos de apoyo a la vigilancia del módulo de Epidemiología del Sistema de Información Hospitalaria alas HIS.

Para poder darle cumplimiento al objetivo general planteado, se trazaron las siguientes **tareas a desarrollar**:

1. Valorar las tendencias actuales en el mundo de los sistemas de información hospitalaria.
2. Asimilar la arquitectura definida por el Departamento de Sistema de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
3. Aplicar las pautas de desarrollo de los entregables y diseño definidas en el Departamento de Sistema de Gestión Hospitalaria.
4. Realizar el modelo de Diseño a partir del modelo de análisis existente.
5. Implementar los procesos de negocio de “Generar Reportes”, “Realizar Inspección Sanitaria” e” Interconsulta”.
6. Obtener los artefactos correspondientes a los flujos de trabajo “Diseño”, “Implementación” y “Prueba”.
7. Obtener el acta de liberación otorgada por Calidad.

Con la implementación del sistema se espera que los beneficios sean agilizar el procesamiento manual de la información, contribuyendo a la informatización de las instalaciones hospitalarias, obtener estadísticas en tiempo real que permitan detectar posibles amenazas epidemiológicas, aumentar la rapidez de notificaciones epidemiológicas en el hospital o fuera del mismo para lograr una mayor eficiencia y garantizar el control centralizado de casos epidemiológicos que posibilite centrar la gestión de la información.

El documento está estructurado en cuatro capítulos:

Capítulo 1 "Fundamentación Teórica": brinda un estado del arte de los Sistemas de Información Hospitalaria que gestionan los procesos asociados al área de Epidemiología.

Capítulo 2 "Descripción de la arquitectura": se definen los requisitos no funcionales, se describe la propuesta de arquitectura y se hace un análisis de la implementación del sistema.

Capítulo 3: "Descripción y análisis de la solución propuesta": se aborda el análisis y diseño de la aplicación. Se presentan los artefactos generados una vez analizado el modelo de datos y la vista de implementación.

Capítulo 4: "Modelo de Prueba": se define como método de prueba a utilizar las de caja negra y se diseñan los casos de pruebas correspondientes a los Casos de Usos del Sistema seleccionados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se brinda información detallada sobre el estado del arte de los Sistemas de Información Hospitalaria a nivel mundial, así como las tendencias actuales de los HIS que cuentan con la presencia y los servicios de un módulo de Epidemiología. Además se realiza un estudio crítico y valorativo de las metodologías, técnicas de programación, plataformas y librerías usadas para el desarrollo del módulo.

1.1 Conceptos básicos relacionados con el problema planteado

Al referirse al módulo de Epidemiología perteneciente a un HIS, se debe tener en cuenta algunos conceptos básicos para un mejor entendimiento del contenido del problema planteado.

En este trabajo, la investigación está enfocada en las unidades de vigilancia epidemiológicas pertenecientes a un hospital, área que se encarga de recopilar toda la información posible sobre epidemias, casos de brotes y agentes epidemiológicos, que en algún momento puedan afectar a la población. Es por ello que todo el análisis está dirigido específicamente a los procesos de vigilancia epidemiológica, los cuales intentan agrupar y reunir en un sistema lógico, completo, articulado y exacto los distintos elementos que permiten conocer los factores que condicionan y determinan el fenómeno y su dinámica en la sociedad, y mantener bajo observación todos los factores o causas incidentes.

Según Alexander Langmuir, doctor reconocido internacionalmente por sus contribuciones en el campo de la epidemiología, la Vigilancia Epidemiológica es la observación activa y permanente de la distribución y tendencias de la incidencia mediante la recolección sistemática, la consolidación y la evaluación de informes de morbilidad y mortalidad, así como de otros datos relevantes. Intrínseca al concepto es la distribución de los datos básicos y su interpretación, a todos los que han contribuido y a todos aquellos que necesitan conocerlos. [22]

La vigilancia epidemiológica tiene diferentes clasificaciones, como son las activas, las pasivas y las especializadas.

La vigilancia pasiva es aquella en la que el especialista no ejecuta personalmente la acción para obtener la información, esta se obtiene directamente de los registros ya establecidos. Las fuentes más comunes donde se encuentran los datos son los anuarios estadísticos, los anuarios de estadísticas vitales, las historias clínicas, las hojas de cargo de las consultas externas, los cuerpos de guardia de hospitales y policlínicos, los registros de Enfermedades de Declaración Obligatoria, los Sistemas de Información

Directa, los certificados de defunción y los certificados de necropsias de Anatomía Patológica y Medicina Legal 2.0, es decir, que se limitan solamente a recoger la información de los sujetos que llegan a los centros de atención médica.

Otro caso es la vigilancia activa que es cuando el especialista ejecuta personalmente la búsqueda de la información específica objeto de la vigilancia, independientemente de que el enfermo o la persona acudan al servicio y se anote o registre el dato rutinariamente. Las fuentes de información de la vigilancia activa la constituyen las encuestas de morbilidad, investigaciones epidemiológicas de brotes epidémicos, controles de foco, las encuestas socioeconómicas, entomológicas y etnográficas. En general son los que recogen la información que se obtiene en el lugar mismo donde ocurren los hechos, sea que el sujeto acuda o no al servicio de salud.

Por último, se tiene la vigilancia especializada, es aquella que se realiza a un problema de salud en particular, debido a compromisos internacionales o prioridades nacionales, campañas de erradicación, enfermedades transmisibles de notificación individual, etc. Este tipo de vigilancia en ocasiones se caracteriza por utilizar elementos de la vigilancia activa y de la vigilancia pasiva, y además se reconoce por su rápida detección, inmediata acción y prevención específica.

1.2 Sistemas automatizados existentes

Es aconsejable fomentar el uso de nuevas tecnologías en la vigilancia epidemiológica y a la vez, realizar un proceso de capacitación permanente, que solidifique el desarrollo de los objetivos sanitarios que requieren obligatoriamente de estos procesos, con el objetivo de crear y producir un impacto positivo en el perfil epidemiológico tanto nacional como internacional.

Con el desarrollo de las tecnologías el sistema de salud ha sido beneficiado, ya que se han desarrollado un gran número de aplicaciones informáticas que se encargan de agilizar el trabajo en las instituciones hospitalarias, aunque no todos tienen desarrollado lo referente a las áreas de Epidemiología.

El grupo de Sistemas Especializados en salud de la Universidad de las Ciencias Informáticas, durante el 2008, implementó el sistema Control Sanitario Internacional (CSI). Uno de sus módulos es el que contiene la información referente al subsistema de Higiene y Epidemiología, que presenta dos partes fundamentales: el control de enfermedades tropicales, que esta primera versión solo se limitó al control de Dengue mediante la vigilancia a pacientes con síntomas febriles inespecíficos y el seguimiento a viajeros

con el objetivo de evitar la introducción y propagación de estas enfermedades en la región. También le permite al usuario recoger datos necesarios, además de estar actualizado en todo momento sobre la situación epidemiológica en el país.

Este software gestiona la información desde la base de aeropuertos, hospitales y unidades de salud, los que muestran la información más actualizada a las autoridades sanitarias en materia de control epidemiológico en los niveles municipales, provinciales y nacionales, según los permisos dentro del sistema. Este es un sistema centralizado, de esta forma, evita la duplicidad y pérdida total o parcial de información sensible sobre el estado epidemiológico del país y permite a los directivos de la especialidad, tomar decisiones precisas y asignar los recursos necesarios para dar respuesta a una situación determinada, así como para prevenirla.

Este módulo del sistema CSI, permite también la gestión de pruebas de laboratorio y seguimiento a pacientes con dengue o sospechosos a padecer esta enfermedad. El sistema CSI está implementado en el lenguaje de programación PHP, con su base de datos en MySQL. La combinación de ambos se ha popularizado en la actualidad dado el buen rendimiento de las aplicaciones web desarrolladas sobre ellas, así como la versatilidad para poder hacer diferentes tipos de software, el dinamismo y la velocidad de respuesta. Hay que tener en cuenta además que es software libre y gratuito, lo cual supone un ahorro por concepto de compra de licencias, soporte y mantenimiento.

Aunque este sistema fue realizado en Cuba y desarrollado con tecnología no propietaria; no se utiliza ya que no implementa una solución enfocada a los procesos de vigilancia epidemiológica de un hospital.

El sistema NOTI de la Oficina General de Epidemiología del Ministerio de Salud de Perú, es una aplicación que se encarga de la vigilancia epidemiológica. Su primera versión fue realizada en 1993, la misma aportó beneficios al sistema de vigilancia epidemiológica, durante sus primeros años fue utilizado principalmente por la Oficina General de Epidemiología. Ya en el año 1995, se le realizaron significativos cambios, necesarios en el software para que pudiera ser utilizado por las direcciones de salud con el objetivo de optimizar la vigilancia epidemiológica en estos ámbitos. El mismo está integrado a otros sistemas tales como HIS de la Oficina General de Estadística e Informática, sistemas de información VIH/SIDA, sistemas de información tuberculosis, sistemas de información malaria, entre otros.

La Organización Panamericana de la Salud cuenta con varias herramientas, las cuales se utilizan con el fin de gestionar información epidemiológica. El SIG-Epi es un Sistema de Información Geográfica (SIG)

diseñado para aplicaciones en Epidemiología y Salud Pública. Ofrece una compilación de técnicas, procedimientos y métodos para el análisis de datos epidemiológicos. Los mismos se presentan de manera simplificada, en un ambiente amigable y en múltiples idiomas. Esta herramienta computarizada apoya y facilita el análisis de situación de salud, el monitoreo y la evaluación de la efectividad de intervenciones, que son requeridas para la toma de decisiones y la planeación en salud. [2]

El sistema de vigilancia epidemiológica para el paciente diabético utiliza la tecnología computacional en la calidad de la atención médica y presentan un instrumento que permite llevar a cabo la vigilancia epidemiológica y evaluar la calidad de la atención a la diabetes *mellitus* en el primer nivel de atención. Este estudio se realizó del 1 de enero de 1998 al 30 de junio de 1999, en la Unidad de Investigación Epidemiológica y en Servicios de Salud, del Instituto Mexicano del Seguro Social (IMSS), en Hermosillo, Sonora. Se diseñó un formato único de reporte compuesto por los diferentes elementos que integran el sistema para la atención del paciente diabético en una Unidad de Medicina Familiar del IMSS.

Por lo que se desarrolló un paquete de computación (software) que permite la captura de los datos de dicho formato y la generación de reportes, de tipo individual y grupal, sobre el cumplimiento de citas, antecedentes personales patológicos y no patológicos, evolución de signos y síntomas, exámenes de laboratorio y el manejo terapéutico.

En Colombia se implantaron las redes de comunicación de Enlace Hispano Americano de Salud (EHAS), ya que existían grandes problemas a la hora de realizar la recolección de información epidemiológica en zonas rurales. Acompañado de esto se probó el Servicio de Información (SIVE) que fuese capaz de soportar estos procesos en las zonas de actuación del programa EHAS, tanto a nivel regional como a nivel local. El SIVE se implantó con el fin de mejorar los procesos de vigilancia epidemiológica, ya que el mismo soporta procesos de notificación y análisis de eventos epidemiológicos a nivel regional, definidos por el sistema SIVIGILA desde las unidades notificadoras de salud hacia hospitales de mayor nivel.

El SIVE fue y está siendo desarrollado en su totalidad con herramientas y programas de Software Libre, considerándose este como elemento primordial para llevar tecnología de punta a zonas rurales aisladas donde los recursos económicos son escasos. La herramienta Web se desarrolló bajo las especificaciones J2EE y un modelo de datos basado en XML que permite una fácil integración de la información y

presentación en diferentes formatos (HTML, PDF, Latex, Excel), compatibilidad con otros sistemas existentes y una mayor flexibilidad en la arquitectura del sistema.

1.3 Herramientas y tecnologías de desarrollo

Eclipse

Eclipse presenta código abierto es su entorno de desarrollo, se caracteriza por su portabilidad y también por ser multiplataforma. Este fue diseñado originalmente por la empresa IBM y actualmente es desarrollado por la Fundación Eclipse, una comunidad de código abierto. [3]

Eclipse trabaja principalmente a base de módulos o *plugins*, lo cual hace posible el trabajo en variados lenguajes de programación como son Java, C++, PHP, Perl y se le puede añadir otras funcionalidades. Permite la integración con la herramienta Visual Paradigm a través del Entorno de Desarrollo Inteligente *(Smart Development Environment),* más conocido por sus siglas en inglés: SDE Enterprise Edition, lo que proporciona un mejor entendimiento entre analistas y desarrolladores. Eclipse a través de SubVersion posibilita el trabajo en equipo mediante el Sistema de Control de Versiones (CVS), los cuales son una combinación de vistas y editores que muestran los diversos aspectos de los recursos del proyecto organizados por el rol o la tarea del desarrollador.

JBoss Seam

Es un *framework* de código abierto que enlaza diferentes tecnologías y estándares java adicionando algunas funcionalidades no contempladas en *frameworks* anteriores. Esto garantiza la plena comunicación de los elementos de la capa de presentación, de negocio y de acceso a datos. Con *Seam* basta agregar anotaciones propias de éste a los objetos Entidad y *Session* de EJB, lo que permite escribir menos código Java y XML. Otra característica importante es que se puede hacer validaciones en los POJOs *(Plain Object Java)* y además manejar directamente la lógica de la aplicación y de negocios desde las *sessions bean* [4].

Algunos de los *framework*s con los que se integra son:

- *Java Server Faces (JSF)*: Para el desarrollo de la interfaz de usuario, en este sentido *Seam* contiene la librería *RichFaces/Ajax4JSf*, que reduce enormemente el esfuerzo de los programadores con los componentes web.
- *Hibernate*: Para el mapeo y el acceso con la base de datos.

Una de las principales ventajas del uso de *Seam* es que permite el control de sus componentes mediante anotaciones, lo cual reduce enormemente la cantidad de archivos XML de configuración.

JBoss Server

JBoss Application Server es el servidor de aplicaciones J2EE, presenta código abierto implementado completamente en Java, lo que implica que puede ser utilizado en cualquier sistema operativo que lo soporte. Por ser una plataforma certificada J2EE, soporta todas las funcionalidades de J2EE 1.4, incluyendo servicios adicionales como *clustering, caching* y persistencia. JBoss es ideal para aplicaciones web en Java. También soporta *Enterprise Java Beans* (EJB) 3.0, y esto hace que el desarrollo de las aplicaciones empresariales sea mucho más simple. Posee la ventaja de ser compatible con la versión 2.0 del *framework* Seam.

Las características destacadas de JBoss 4.2 incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Soporta los estándares:
  - ✓ Portlet Specification and API 1.0 (JSR-168)
  - ✓ Content Repository for Java Technology API (JSR-170)
  - ✓ Java Server Faces 1.2 (JSR-252)
  - ✓ Java Management Extensión (JMX) 1.2
  - ✓ Compatibilidad 100% con J2EE 1.4
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.

- Flexibilidad consistente.
- Servicios del *middleware* para cualquier objeto de Java.

JBoss Tools

Jboss Tools que es un conjunto de herramientas para Eclipse. Entre estas herramientas, Jboss Tools dispone de *plugins* que proporcionan soporte en Eclipse para Hibernate, JBoss AS, Drools, jBPM, JSF, (X)HTML, Seam, Smooks, JBoss ESB o JBoss Portal, entre otros [5]

Algunos de estos *plugins* son:

- RichFaces VE: Editor visual para componentes HTML, JSF y RichFaces
- Seam tools: Incluye soporte para la integración de los componentes del *framework* Seam.
- Hibernate tools: Sirve de apoyo para la utilización de los componentes del *framework* Hibernate y para el mapeo con la base de datos.

RichFaces

RichFaces es una rica biblioteca de componentes para JSF y un avanzado marco para integrar fácilmente capacidades Ajax en el desarrollo de aplicaciones de negocios. Los componentes de la interfaz de usuario de RichFaces vienen preparados para su uso fuera del paquete, así los desarrolladores ahorrarán tiempo para la creación de aplicaciones Web. [6]

RichFaces permite definir (por medio de etiquetas de JSF) diferentes partes de una página JSF que se desee actualizar con una solicitud Ajax, proporcionando así varias opciones para enviar peticiones Ajax al servidor.

PosgtreSQL

PostgreSQL es un potente motor de bases de datos de código abierto que posee prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Además de presentar una alta concurrencia que evita tener que bloquear una tabla en el momento que se está escribiendo sobre ella. También permite realizar copias de seguridad en línea, replicación asíncrona, transacciones anidadas y optimizador de consultas.

PostgreSQL permite la creación de procedimientos almacenados, restricciones de integridad y vistas, destacándose por su robustez, escalabilidad y cumplimiento de los estándares SQL. Cuenta con diversas versiones para sistemas operativos tales como: Linux, Windows, Mac OS X, Solaris, BSD, Tru64 entre otros, e incluye la mayor parte de los tipos de datos especificados en los estándares SQL92 y SQL99, como: entero, numérico, booleano, char, varchar, fecha, interval o timestamp.

Algunas de sus funciones incluyen replicación, respaldos pesados, optimizador avanzado, codificado de caracteres de *multibyt*e, y un tamaño de base de datos ilimitado. Del mismo modo es capaz de soportar Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID acrónimo de Atomicity, Consistency, Isolation and Durability), lo que proporciona la realización de transacciones seguras, además de vistas, uniones, claves foráneas, procedimientos almacenados, *triggers*, entre otras cosas.

La versión 8.3 de PostgreSQL trae consigo nuevas características con el afán de mejorar el trabajo de los usuarios, tales como: soporte SQL/XML de acuerdo con el estándar ANSI, incluyendo exportación en formato XML; herramienta avanzada de búsqueda en texto, TSearch2, incorporada en la distribución central, con mejor manejo y nuevos diccionarios e idiomas; soporte de autentificación GSSAPI y SSPI; y nuevos tipos de datos: UUIDs, ENUMs y arreglos de tipos compuestos. Nuevas opciones de registro (*logging*) han sido agregadas y el sobrecosto del recolector de estadísticas ha sido disminuido para hacer más fácil el monitoreo de sus servidores.

PostgreSQL Maestro

PostgreSQL Maestro es una herramienta de administración GUI de alta calidad para el sistema operativo de Windows cuyo objetivo es la administración, control y desarrollo en servidores PostgreSQL. [7]

Este permite crear, editar, copiar, extraer y bajar todo objeto de las bases de datos tales como esquemas, tablas, vistas, funciones, dominios, reglas, secuencias, idiomas, operadores, etc. Además, brinda la posibilidad de construir consultas visualmente, ejecutar consultas y Scripts SQL, así como visualizar y editar datos incluyendo BLOBs.

Por otra parte, esta herramienta brinda la posibilidad de representar datos como diagramas, exportar e importar datos desde y hacia los formatos de archivos de uso más popular, administrar roles PostgreSQL, usuarios, grupos y sus privilegios; sin contar con que usa una serie de herramientas adicionales diseñadas para la más fácil y eficiente operación con el Servidor PostgreSQL. También brinda soporte a todas las

versiones de PostgreSQL de la 7.3 a la 8.2, permite una fácil administración de base de datos y gestión de objetos particulares de la base de datos.

PostgreSQL Maestro ofrece gran facilidad para la creación de tablas, obteniendo una descarga de su *metadata* y dato. Permite obtener reportes sobre sus bases de datos, los cuales se pueden imprimir en formatos de alta calidad, ejecutando cualquier Script de SQL. También posibilita realizar operaciones de arrastrar y soltar, o presionando el juego de teclas Ctrl+C/Ctrl+V para copiar una tabla de una base de datos a otra y una serie adicional de características.

Framework Hibernate

Hibernate es una capa de persistencia objeto/relacional. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. [8]

De una manera muy rápida y optimizada se podrá generar Bases de Datos en cualquiera de los entornos soportados: Oracle, DB2, MySQL, entre otras. Además, presenta código abierto.

Uno de los posibles procesos de desarrollo consiste en, una vez que se tenga el diseño de datos realizado, *mapear* este a ficheros XML a través del mapeo de Hibernate. Desde estos podremos generar el código de nuestros objetos persistentes en clases Java y también crear Bases de Datos, independientemente del entorno escogido. Es capaz de integrarse a cualquier tipo de aplicación.

jBPM

jBPM es un sistema flexible para administración de procesos de negocios, también se considera un motor de flujo de trabajo escrito en Java que puede ejecutar procesos descritos en BPEL(Lenguaje de Ejecución de Procesos de Negocio), o en su propio lenguaje de definición de proceso JPDL. Además es liberado bajo la licencia LGPL. Ofrecen características de BPM y de flujo de trabajo, de manera que les sean útiles, tanto los usuarios de negocio como los desarrolladores.

Visual Paradigm

Visual Paradigm es una herramienta CASE (Computer Aided Software Engineering) que da soporte al modelado visual con UML y con la notación para la gestión de procesos de negocio (BPMN por sus siglas en inglés). Esta permite recorrer el ciclo de vida del desarrollo de software, desde el modelado de negocio hasta el despliegue del producto, generando artefactos en toda su trayectoria. Dentro de las ventajas de

esta herramienta se encuentra la rápida construcción de aplicaciones con gran calidad y menor costo. Permite generar diagramas de clases, código desde diagramas y documentación. Proporciona además abundantes tutoriales de UML, así como demostraciones interactivas de dicho lenguaje.

Esta herramienta CASE se integra a la Plataforma Java, la cual funciona en sistemas operativos Windows, Linux y Mac OS X. La integración es vista con ambientes inteligentes de desarrollo (SDE) para Eclipse, NetBeans, Oracle JDeveloper, JBuilder, entre otros.

iReport

La herramienta iReport es un constructor / diseñador de informes visual, poderoso, intuitivo y fácil de usar para JasperReports escrito en Java. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. iReport está además integrado con JFreeChart, una de la biblioteca gráficas OpenSource más difundida para Java. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, TableModels, JavaBeans, XML, etc.

JasperReport

Jasper Report se conoce por ser una biblioteca de clases de generación de informes, la cual es libre y está descrita en Java. Se encarga de generar archivos XML donde se recogen las particularidades del informa. La salida de este archivo puede ser un PDF, XML, HTML, CSV, XLS, RTF, TXT una vez tratada por las clases Jasper. Una de las ventajas del JasperReport es que se integra fácilmente con la librería de JFreeChart para todo tipo de gráficos.

1.4 Lenguaje de programación

Un lenguaje de programación no es más que un idioma artificial o una secuencia de símbolos diseñado por el hombre para expresar operaciones que puedan ser llevadas a cabo por una máquina. Puede utilizarse con el fin de crear reglas sintácticas o semánticas, métodos, algoritmos con precisión o como modo de comunicación humana para controlar el comportamiento lógico y físico de una máquina.

Java

Dentro de los lenguajes de programación se encuentra Java, el cual es utilizado para crear aplicaciones informáticas. Es un lenguaje orientado a objeto, de una plataforma independiente. La programación en Java permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar. [10]

El lenguaje hereda mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel. Debido a esto ocurren menos errores relacionados con la manipulación directa de punteros o memoria, por lo que se conoce como un lenguaje potente.

Entre sus principales características se destacan:

- Una misma aplicación puede funcionar en diversos tipos de ordenadores y sistemas operativos: Windows, Linux, Solaris, MacOS-X, así como en otros dispositivos inteligentes, de ahí que es multiplataforma.
- Los programas Java pueden ser aplicaciones de escritorio, se ejecutan sin necesidad de un navegador. *Applets*: Se pueden descargar de Internet y se observan en un navegador. *JavaBeans*: Componentes software Java, que se puedan incorporar gráficamente a otros componentes. *JavaScript*: Conjunto del lenguaje Java que puede codificarse directamente sobre cualquier documento HTML *Servlets:* Módulos que permiten sustituir o utilizar el lenguaje Java en lugar de programas CGI (Common Gateway Interface) a la hora de dotar de interactividad a las páginas Web.
- Su desarrollo está impulsado por un amplio colectivo de empresas, organizaciones y conecta con la filosofía de software abierto y entorno colaborativo.
- Elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el *garbage collector* (en español, recolector de basura). No es necesario preocuparse de liberar memoria, el recolector se encarga de ello y como es un *hilo* de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.
- Reduce en un 50% los errores más comunes de programación con lenguajes como C y C++

1.5 Metodologías de desarrollo

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software, en Inglés software development methodology (SDM) o system development life cycle (SDLC).

La finalidad de una metodología de desarrollo es garantizar la eficacia (p. ej. cumplir los requisitos iniciales) y la eficiencia (p. ej. minimizar las pérdidas de tiempo) en el proceso de generación de software.[11]

En la actualidad no se debe hablar de una única metodología universal de desarrollo ya que a la hora de crear un proyecto, en dependencia de las características y de la envergadura del mismo es que tiene sentido basarnos en la metodología más adecuada para utilizar en ese proceso. El Proceso Unificado Racional (RUP) está basado en UML y es una metodología Orientada a Objetos (OO).

RUP

RUP es un proceso para el desarrollo de un proyecto de un software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto. [12]

La versión de RUP que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo. Dicho proceso tiene tres características fundamentales. La primera de ella es que está dirigido por casos de uso, es decir, que en el proyecto se orientan a la importancia que tiene para el usuario lo que el producto debe hacer. También es un proceso centrado en la arquitectura ya que relaciona la toma de decisiones que indican cómo tiene que ser constituido el sistema y en qué orden se debe hacer. Es iterativo e incremental, divide el proyecto en miniproyectos donde los casos de usos y la arquitectura cumplen sus objetivos de manera más depurada.

El RUP se encarga de unificar todo el equipo de desarrollo de software, además de optimizar su comunicación proveyendo a cada miembro de una aproximación al desarrollo de software con una base de conocimiento de acuerdo con las necesidades específicas del proyecto. Él no es simplemente un

proceso, sino que es un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. Generalmente es aplicado a grandes proyectos de desarrollo de software.

Dentro de sus disciplinas gestiona el control de cambios, que permite mantener al equipo trabajando en los mismos artefactos, en cualquier momento del desarrollo del producto. En su modelación RUP define como sus principales elementos a los trabajadores, las actividades, los artefactos y los flujos de actividades. Los trabajadores son los propietarios de elementos o artefactos y se encargan de realizar las actividades, las cuales describen cómo una tarea es realizada por un trabajador y a su vez, manipulan elementos. Los artefactos constituyen los productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos. El flujo de actividades describe cuando estas son realizadas por trabajadores y produce un resultado de valor observable.

El ciclo de vida RUP es una implementación del desarrollo en espiral, creado ensamblando los elementos en secuencias semiordenadas, organizando las tareas en fases e iteraciones. Las cuatro fases de RUP son:

- ***Conceptualización (Concepción o Inicio):*** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- ***Elaboración:*** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- ***Construcción:*** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios *release* o versiones del producto que han pasado las pruebas.
- ***Transición:*** El producto ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

RUP además de estar compuesto por las fases anteriormente mencionadas, también cuenta con 9 flujos de trabajo de los cuales los primeros seis son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

Dentro de las disciplinas de ingeniería se encuentran las siguientes:

El ***modelado del negocio*****,** el cual describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

La ***captura de requerimientos*** define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

El ***análisis y diseño*** describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

La disciplina de ***implementación*** define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

El flujo de trabajo de **p*ruebas*** se encarga de buscar los defectos a lo largo del ciclo de vida.

El ***despliegue*** produce el *release* del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

Las disciplinas de apoyo que define RUP son:

Administración del proyecto, la cual involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

Administración de configuración y cambios que describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

Ambiente el cual contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. [13]

Aunque no se considera un estándar oficial, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

Este se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software como es el caso de RUP, pero no especifica en sí mismo qué metodología o proceso usar. UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Estos tienen como objetivo fundamental presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

Una vez culminado este capítulo, se realizó el estudio de los conceptos básicos relacionados con el problema planteado, además de las investigaciones sobre los sistemas automatizados existentes en el área de Epidemiología. Se puede plantear que los procesos de vigilancia epidemiológica, así como la generación de reportes no cuenta con sistemas informáticos estandarizados, ya que la información referente a esta área se gestiona de forma particular en las diferentes instalaciones hospitalarias a nivel mundial. Lo cual se debe en su mayoría, a que la elección de la arquitectura de software para la construcción de sistemas que informaticen los distintos servicios de un hospital, difieran de la seleccionada por los otros o simplemente posee dificultades de integración.

Para el desarrollo de la aplicación, debido al manejo eficiente y racional de los recursos de memoria, por estar bajo licencias de código abierto y facilitar el trabajo de los desarrolladores, se utiliza *RichFaces* como biblioteca de componentes web, *Hibernate* como mapeador objeto-relacional, *Jboss 4.2 GA* como servidor de aplicaciones web, Eclipse como entorno de desarrollo integrado y Seam como *framework* de integración.

CAPÍTULO 2: DESCRIPCIÓN DE LA ARQUITECTURA

En el contexto de este capítulo se plasma lo referente a los requisitos no funcionales de la propuesta a solución técnica a partir del análisis teórico anterior. También se da una breve descripción de la arquitectura del sistema a implementar y el análisis de posibles implementaciones, componentes o módulos ya existentes que puedan ser rehusados para viabilizar el trabajo final. Además, se explica los mecanismos de seguridad a utilizar en la creación del Software.

2.1 Requisitos no funcionales

Los requisitos no funcionales de un software son propiedades o cualidades que el producto debe tener. Estos muestran las características que hacen al producto atractivo, usable, rápido o confiable. A diferencia de los requisitos funcionales, estos no modifican o alteran la funcionalidad del producto. Los requisitos deben ser especificados por escrito y posibles de verificar o probar, además sus especificaciones deben ser lo más abstracto y conciso posible y descritos como una característica del sistema. Los requisitos no funcionales se dividen en varias categorías, a continuación se listan algunas de los más usados en el sistema a desarrollar:

Usabilidad

El sistema estará diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo, en un tiempo reducido, de forma tal que se conviertan en usuarios normales en 20 días y en avanzado en 30 días.

Seguridad

Este es uno de los más difíciles, ya que provocará los mayores riesgos si no se maneja correctamente. La seguridad de un sistema no solo tiene en cuenta la seguridad del sistema propiamente dicho sino, además, el ambiente en el que se usará el sistema. Por lo que se tiene que contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso al sistema y las regulaciones legales que afecta o determina el uso del sistema y que serán tenidas en cuenta si se incumple.

La seguridad puede ser tratada en tres aspectos diferentes: confidencialidad, integridad y disponibilidad, por lo que se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo con la función que realizan.

Las contraseñas podrán cambiarse sólo por el propio usuario o por el administrador del sistema. Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento. El sistema proporcionará un registro de actividades (log) de cada usuario en el sistema. Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD. Y por último, 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 además optimizará el uso de recursos críticos como la memoria y a la vez respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos.

Soporte

Este requisito abarca todas las acciones a tomar una vez que se ha terminado el desarrollo del software con motivos de asistir a los clientes de éste así como lograr su mejoramiento progresivo y evolución en el tiempo.

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

Hardware

- ***Estaciones de trabajo***

En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS, las que necesitan capacidad de hardware que soporte un sistema operativo que

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

- ***Servidores***

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

Software

El servidor debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA y los clientes deberán disponer de un navegador web, estos pueden ser IE 7 o superior, Opera 9 superior, Google Chrome 1 o superior y Firefox 2 o superior.

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

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

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

Interfaz

- ***Interfaces de usuario***

Las ventanas del sistema contendrán claramente y bien estructurados los datos, además de permitir la interpretación correcta de la información. La interfaz contará con accesos directos y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario y 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 estándar HL7 (Health Level Seven). El sistema usará el formato estándar WSDL 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.

Portabilidad

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

2.2 Descripción de la arquitectura.

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Estas cuestiones estructurales se vinculan con el diseño, pues la arquitectura de software es una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema.

En la creación del HIS se utiliza la arquitectura basada en el patrón Modelo Vista Controlador, el mismo fue descrito por primera vez en 1979 por Trygve Reenskaug y está diseñado en función de reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Este patrón es unos de los más usados a la hora de crear aplicaciones web, ya que aquí la vista contiene las paginas HTML y la parte del código provee de datos dinámicos a la página. El modelo es el Sistema

Gestor de Base de Datos y el controlador es el responsable de la Lógica de negocio al recibir los eventos de entrada desde la vista. Todo esto permite que un elemento de una capa pueda ser sustituido causando el mínimo de alteraciones en otro elemento que lo utilice.

La capa de presentación está desarrollada básicamente con JSF, usando la librería de componentes Richfaces, esta integra fácilmente con el *framework* de integración escogido Seam y permite generar vistas no necesariamente basadas en HTML (PDF, etc.). Incluye conversión y validación de campos, establecimiento de reglas de navegación declarativas, la internacionalización y accesibilidad de la interfaz de usuario, un modelo orientado a eventos y combinado con Facelets, además de que nos da la capacidad añadida de la tecnología de plantillas de Facelets. Por su parte los controles para UI de Seam adicionan varias mejoras a JSF, desde validación, expresiones EL extendidas, integración de la navegación en la UI basada en *pageflows* o procesos del negocio, etc.

La capa de negocio está integrada por clases controladoras, que encierran la lógica del negocio del módulo. A las cuales, mediante anotaciones que provee el framework Seam, se les especifica el contexto en que se encuentran (conversacional, evento, página, etc.), los cuales definen el estado de los datos y las entidades que manejan.

Para el acceso a datos se usa la implementación de JPA de Hibernate, minimizando por un lado las configuraciones en XML sin chequeo de tipos. Y por otro lado usando los servicios del contenedor de EJB3 y/o los contextos de persistencias administrados por Seam, se elimina gran parte del código “infraestructural” en cuanto a transacciones, la transmisión del contexto de persistencia, etc. Además se pueden establecer validaciones mediante los Hibernate Validators.

La comunicación entre los elementos de estas capas está regida por el *framework* Seam. Este permite, mediante anotaciones, que las páginas de la interfaz de usuario referencien las funcionalidades definidas en las clases controladoras, y que estas puedan usar los componentes de acceso a datos y otros de la capa del negocio.

2.3 Posibles implementaciones de componentes o módulos que puedan ser reutilizados. Estrategias de integración

El proceso de interconsulta de un hospital es el procedimiento mediante el cual, a petición de un médico, otro médico de otra área revisa la historia clínica de un paciente, así como su situación actual y en base a

lo observado realiza recomendaciones sobre la asistencia y el tratamiento. El módulo de Hospitalización del sistema alas HIS contiene todo lo referente a la gestión de solicitudes de interconsultas y la creación de hojas de interconsultas.

En este módulo se crean solicitudes de interconsultas para ser enviadas a las demás áreas del hospital, las cuales se encargan de aceptarlas y en el caso que sea necesario, crear una hoja de interconsulta. Por otra parte el área de Hospitalización puede recibir solicitudes de interconsultas y ser ellos los que la acepten y generen las hojas de interconsultas.

El área de epidemiología se encarga de recibir interconsultas generadas por otras áreas del hospital, las cuales llegan al médico epidemiólogo y éste se encarga de aceptarlas y crear las hojas de interconsultas. Por lo que el módulo de Epidemiología tiene funcionalidades comunes con el proceso de interconsulta que se desarrolla en el módulo de Hospitalización, incluyéndose solamente las funcionalidades referentes a" Crear Hoja de interconsulta" y "Ver Solicitudes de Interconsultas".

Al médico epidemiólogo sólo llegan las solicitudes de interconsultas que contengan el campo "epidemiología" con valor "*true*". Cuando estas son aceptadas en esta área, el proceso es el mismo que en módulo de Hospitalización, pero con la diferencia que al crear una hoja de interconsulta en el módulo no se trabaja con el campo diagnóstico que tiene incluido la hoja de solicitud. En este caso al reutilizar las funcionalidades del módulo de Hospitalización en Epidemiología se debe modificar el código de la página para requerir las recomendaciones y es necesario, además, hacer corresponder la página con el objeto controlador *sesión beans* <CrearHojaInterconsultaEpidemiologiaControlador> del módulo de Epidemiología y referenciar al atributo recomendaciones para que persistan en el campo correspondiente de la base de datos.

2.4 Seguridad

Para fomentar la seguridad en el sistema se propone un control de acceso a nivel de usuarios y contraseñas, así como su diferenciación atendiendo al rol de cada uno para asegurar que cada usuario puede acceder solamente a los lugares que su rol se lo permite. Se propone que las contraseñas de los usuarios solo podrán ser cambiadas por los usuarios cada cierto tiempo, o por los administradores del sistema ante cualquier situación anormal. El sistema permitirá a través de una bitácora de sucesos, llevar

una traza de todas las operaciones realizadas por cada uno de los usuarios mediante un registro de actividades en todo momento.

La fidelidad de los datos es otro de los aspectos a tener en cuenta. Esta se logra mediante el cifrado de la información proveniente de la comunicación entre los componentes internos del sistema y otros sistemas que soliciten información desde otra Institución Hospitalaria. Esto impide la lectura o modificación de los datos confidenciales que se manejan y de esta forma aumenta la seguridad del sistema.

2.5 Vista de Despliegue

El objetivo de un diagrama de despliegue es mostrar las relaciones físicas entre nodos del sistema final, hardware y software. Este diagrama se considera un grafo de nodos unidos que logran su comunicación a través de conexiones; donde un nodo es un objeto físico con capacidad operacional. Generalmente en tiempo de ejecución los nodos se consideran recursos computacionales, con capacidad de procesamiento y memoria. A continuación se muestra el Diagrama de Despliegue correspondiente al módulo de Epidemiología.

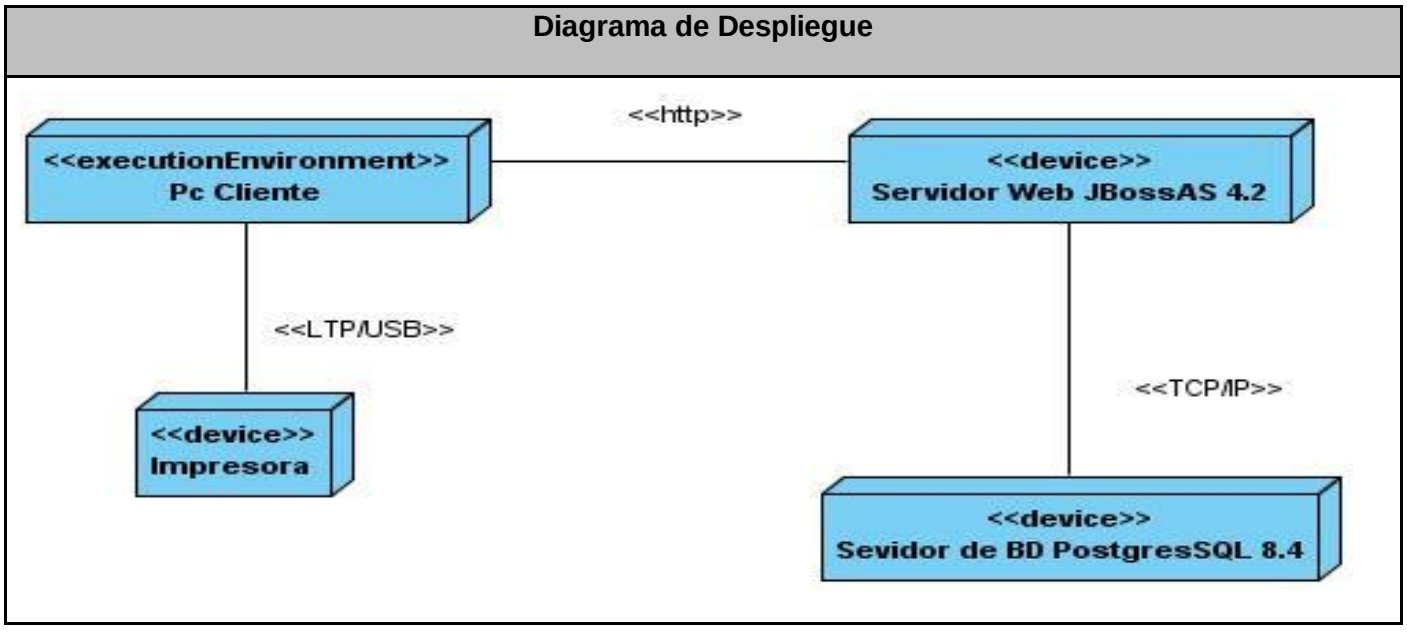

Figura 2.1 Diagrama de Despliegue.

2.6 Estrategias de codificación. Estándares y estilos a utilizar

El mejor método para lograr la legibilidad del código implementado por más de una persona, es la aplicación de un estándar de codificación. Un estándar de codificación completo debe comprender todos los aspectos de la generación de código. Estos estándares se utilizan con el fin de lograr organización, uniformidad y homogeneidad como si un único programador hubiese escrito todo el código. La aplicación del mismo también ayuda a evitar las confusiones y el no entendimiento del código entre todos los desarrolladores de un proyecto y el resto del personal que debe revisarlo.

Al aplicar correctamente y de forma sistemática un estándar de codificación bien definido, desde el principio hasta el final del proyecto de software se asegura que el producto se convierta en un sistema de software fácil de comprender y de mantener, generando un código con calidad y organización, minimizando los errores a la hora de entenderlo; ya sea para su revisión o para realizarle modificaciones, al agregarle más funcionalidades o características al sistema.

Se incluyen en el estándar el uso de las notaciones CamellCasing, para las variables y métodos con nombres compuestos por múltiples palabras juntas, el cual sugiere iniciar cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula, y PascalCasing para las clases con nombres compuestos por múltiples palabras juntas, el cual sugiere iniciar cada palabra con letra mayúscula. A continuación se muestran algunas restricciones de la nomenclatura del código haciendo uso de los estándares de codificación mencionados anteriormente:

Variables y constantes		
Apariencia de constantes	Todas sus letras en mayúscula	Se deben declarar las constantes con todas sus letras en mayúscula.
Aspectos generales	Nombres de las variables y constantes	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Tabla 2.1 Estándares de codificación - Variables y constantes.

Identación	
Inicio y fin de bloque	Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.
Aspectos generales	El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la computadora o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar { en la línea de un código cualquiera, esto requiere una línea propia.

Tabla 2.2 Estándares de codificación – Identación.

Comentarios, separadores, líneas, espacios en blanco y márgenes		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.
Líneas en blanco	Se emplean antes y después de métodos, clases y estructuras.	Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función
Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nomproducto

Aspectos generales	Sobre el comentario	Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción
	Sobre los espacios en blanco	No se debe usar espacio en blanco: Después del corchete abierto y antes del cerrado de un arreglo. Después del paréntesis abierto y antes del cerrado. Antes de un punto y coma.

Tabla 2.3 Estándares de codificación - Comentarios, separadores, líneas, espacios en blanco y márgenes.

Clases y Objetos		
Apariencia de clases y objetos	Primera letra en mayúscula	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Ejemplo: MiClase(). Para el caso de las instancias se comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación Camello.
Declaración de parámetro en	Agrupados por tipos Poner los string 1	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén

funciones	numéricos 2, además, agrupar según valores por defecto.	agrupados por el tipo de dato que contienen, especificando el tipo de datos (tabla 51.2).
Aspectos generales.	Sobre las clases, los objetos, los atributos y las funciones.	El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

Tabla 2.4 Estándares de codificación - Clases y Objetos.

Bases de datos, tablas, esquemas y campos		
Apariencia de la base de datos.	Las 2 primeras letras representan el tipo.	Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación underscoard y luego el nombre completamente en minúscula, en caso de que sea un nombre compuesto se empleará notación. Los nombres serán cortos y descriptivos.
Apariencia de las vistas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula	El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.
Apariencia de las tablas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizara underscoard para separarlo.

Tablas que representen Relaciones	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre de será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula.
Tablas que representen nomencladores.	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre de será corto y descriptivo todo en minúscula.
Apariencia de los procedimientos almacenados.	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	En los procedimientos el nombre debe comenzar con el prefijo pa, underscoard y luego todas las letras en minúscula en caso de que sea un nombre compuesto se utilizara underscoard para separarlo.
Apariencia de los campos	Todas las letras en minúscula.	El nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.
Nombre de los campos	En caso de identificadores.	Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo.
Sentencias SQL	Todas las letras en mayúscula.	Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.
Aspectos generales	Sobre las BD, vistas, tablas atributos y procedimientos.	El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con sólo leerlos se conozca el propósito de los mismos.

Tabla 2.5 Estándares de codificación - Bases de datos, tablas, esquemas y campos.

Controles		
Apariencia de los controles.	Los controles tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula, las cuales identificaran el tipo de datos al que se refiere (ver tabla 1.2), en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Tabla 2.6 Estándares de codificación – Controles.

Tipo Datos	**Prefijo**	**Ejemplo**
Int	i	iCantPacientes
double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	e	eSexo
sbyte	sb	sbEdadPaciente
short	sh	shVariableShort
uint	ui	uiVariableUint
long	l	lVariableLong

Objetos	o	oPacienteHistorico

Tabla 2.7 Estándares de codificación - Tipo Datos.

Control	**Prefijo**	**Ejemplo**
Botón	btn	btnAceptar
Etiqueta	lbl	lblNombre
Lista/Menú	mn	mnPrincipal
Campo de Texto	txt	txtFecha
Botón de Opción	bpt	optSexo
Casilla de Verificación	chx	chxBorrar
Casilla de Selección	cbx	cbxSexo

Tabla 2.8 Estándares de co0064ificación – Control.

En el contenido del presente capítulo se han analizado los requisitos no funcionales, especificando cada uno de ellos; así como la descripción y la fundamentación de la arquitectura propuesta para el desarrollo de la aplicación. Además, se realizó el análisis de posibles implementaciones, llegando a la conclusión de que existen componentes ya existentes, los cuales pueden ser reutilizables. También se explica lo referente a la seguridad del sistema, basándose principalmente en el nivel de acceso de los usuarios. Al mismo tiempo, se describe la vista de despliegue y la estrategia de codificación a ser usadas en todo el desarrollo del sistema propuesto.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo se aborda el análisis y diseño de la aplicación. Además, se presentan los artefactos generados una vez analizado el modelo de datos, así como la vista de implementación.

3.1 Valoración crítica del diseño propuesto por el analista

Una vez realizado el Análisis, el Diseño se encarga de darle un refinamiento al mismo. El Diseño se intenta desarrollar sin ambigüedades, para así traducir los requisitos a una especificación que describe como implementar el sistema, pretendiendo crear un modelo de implementación. Se definen ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física.

En esta fase se realiza el diseño de los datos, donde se transforma el modelo de dominio de la información creado en el análisis, en las estructuras de datos necesarios para implementar el software. Al mismo tiempo, se realiza el diseño arquitectónico, por lo que se definen las relaciones entre cada uno de los elementos estructurales del programa. Conjuntamente se realiza el diseño de la interfaz, se describen como se comunica el software consigo mismo y con los sistemas que operan junto con él, con los operadores y usuarios que lo emplean. También se diseñan los procedimientos, donde se transforman elementos estructurales de la arquitectura del programa.

El Diseño es la única manera de materializar con precisión los requerimientos del cliente y a la vez, se fomenta la calidad del proyecto. El proceso de Diseño es un conjunto de pasos repetitivos que permiten al diseñador, describir todos los aspectos del sistema a construir. Además, es esta fase la que debe proporcionar una completa idea de lo que es el software, enfocando los dominios de datos y el comportamiento desde el punto de vista de la Implementación.

Existen varios tipos de patrones de diseño para aplicar según el trabajo especifico que se esté realizando, encaminados a lograr mejores soluciones y a la reutilización del conocimiento. Los patrones GRASP brindan grandes ventajas ya que se encargan de asignar responsabilidades a las diferentes clases que se definen en el Diseño. Dentro de este grupo de patrones se pueden mencionar: experto, creador, alta cohesión, bajo acoplamiento y controlador.

El Diseño también se encarga de modelar el sistema para que soporte todos los requisitos, funcionales y no funcionales que se le suponen. En este modelo, los casos de uso son realizados por las clases del diseño y sus objetos, a partir de los cuales se forma el diagrama de clases del diseño.

Una clase de diseño es aquella muy bien detallada que se utiliza como base para generar código fuente, sus especificaciones están descritas a un nivel que puedan ser implementadas. Los métodos de las mismas deben de estar enfocados al cumplimiento de un determinado servicio para la clase.

Los diagramas de clases de diseño son los diagramas principales del diseño, exponen un conjunto de interfaces, colaboraciones y sus relaciones. Especifican la estructura de clases del sistema y las relaciones entre clases y estructuras de herencia. Forman parte de las realizaciones de casos de usos y se realizan para lograr un mejor entendimiento del sistema.

Los diagramas de interacción también se consideran artefactos que muestran gráficamente cómo los objetos se comunican entre ellos, a fin de cumplir con los requerimientos. Estos diagramas de interacción desempeñan un papel importante dentro de la metodología RUP, ya que sirven para modelar los aspectos dinámicos de un sistema.

Los diagramas de interacción se dividen en dos tipos: los Diagramas de Colaboración y los Diagramas de Secuencia. Por lo general se utiliza el Diagrama de Secuencia para la etapa de Diseño, ya que el de Colaboración habrá sido utilizado en el Análisis; este se caracteriza por el orden en que se muestran los mensajes, dando la sucesión de los pasos a realizarse en el sistema.

Para facilitar el trabajo se permite dividir el sistema en partes manejables para su futura implementación, definiéndose una estructura de paquetes. Estos pueden ser separados por clases o por procesos. A partir de los procesos definidos en el sistema, se grafican los paquetes correspondientes, los cuales utilizan el paquete repositorio de clases para su funcionamiento.

A continuación se muestra el Diagrama de Paquetes del sistema propuesto:

Diagrama de paquetes

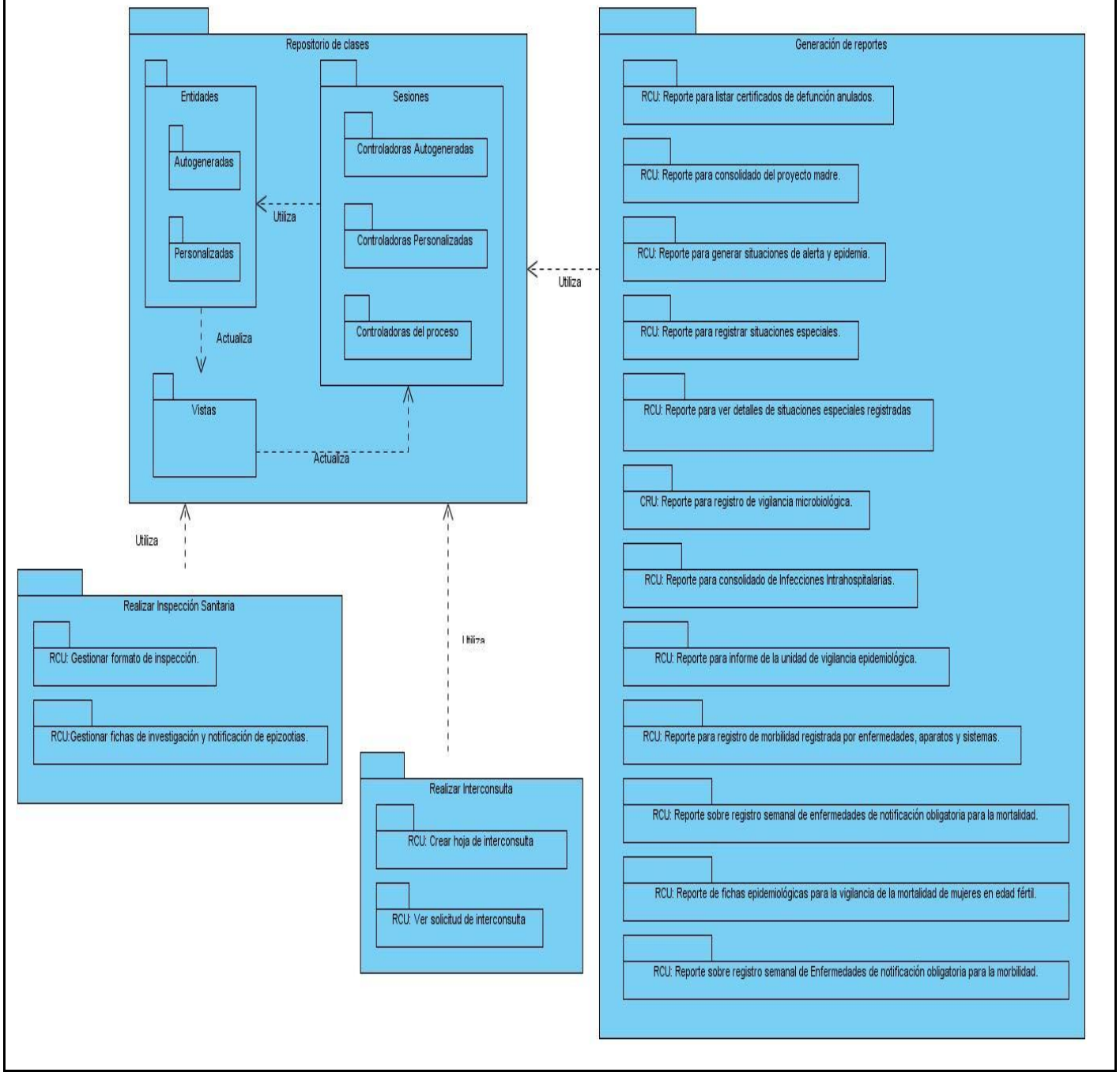

Figura 3.1 Diagrama de Paquetes.

DCD_ Gestionar fichas de investigación y notificación de epizootias

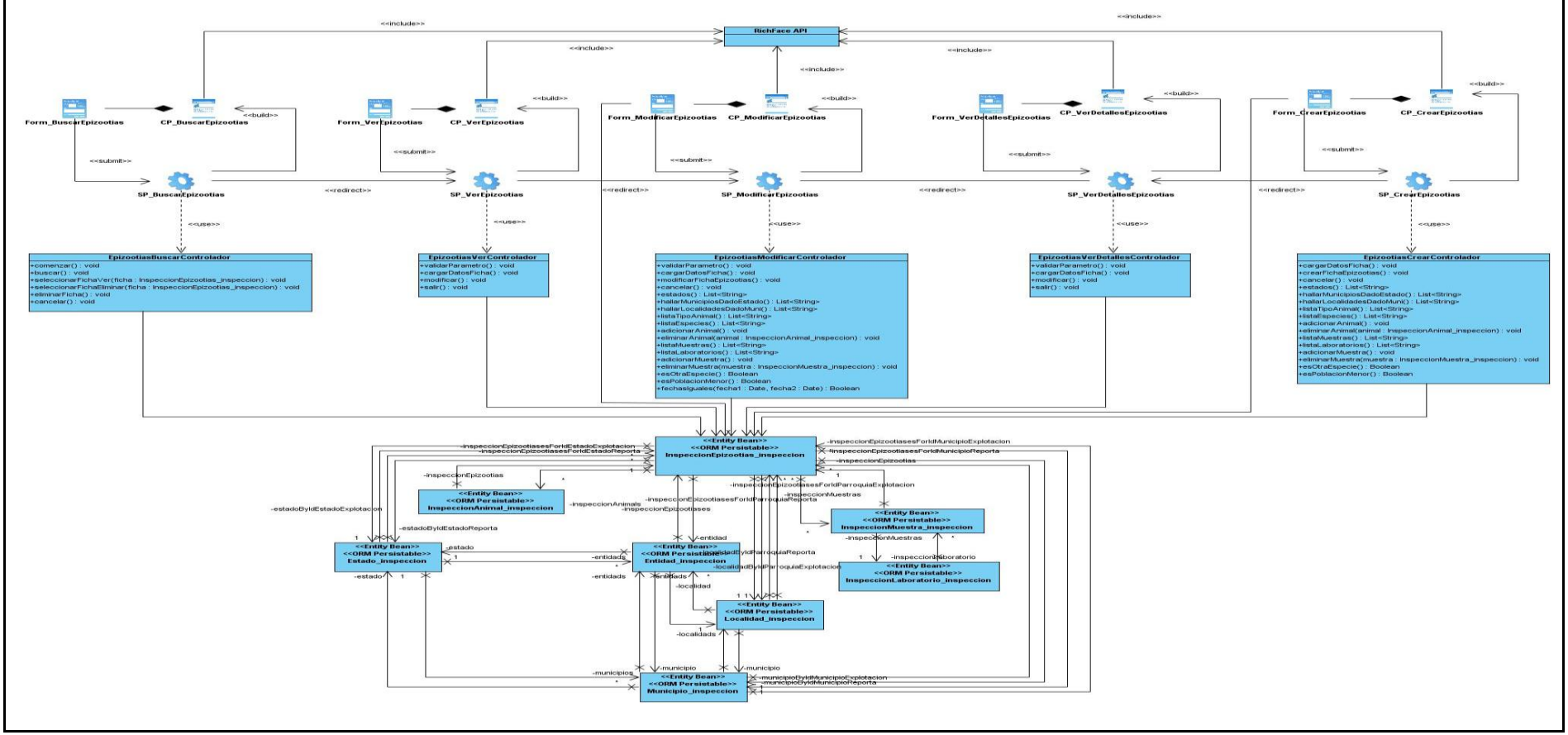

Figura 3.2 Diagrama de Clases del Diseño – Gestionar fichas de investigación y notificación de epizootias.

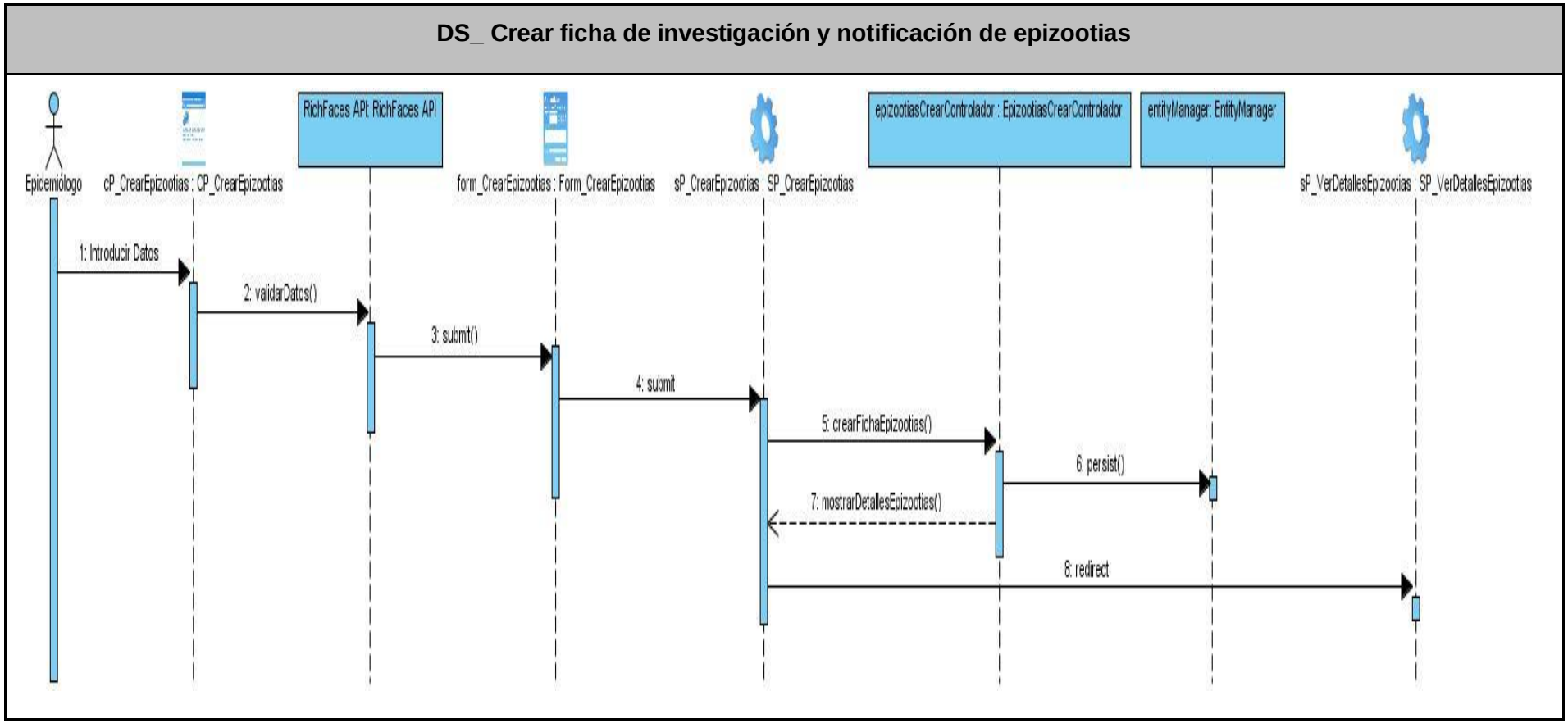

Figura 3.3 Diagrama de Secuencia – Crear ficha de investigación y notificación de epizootias.

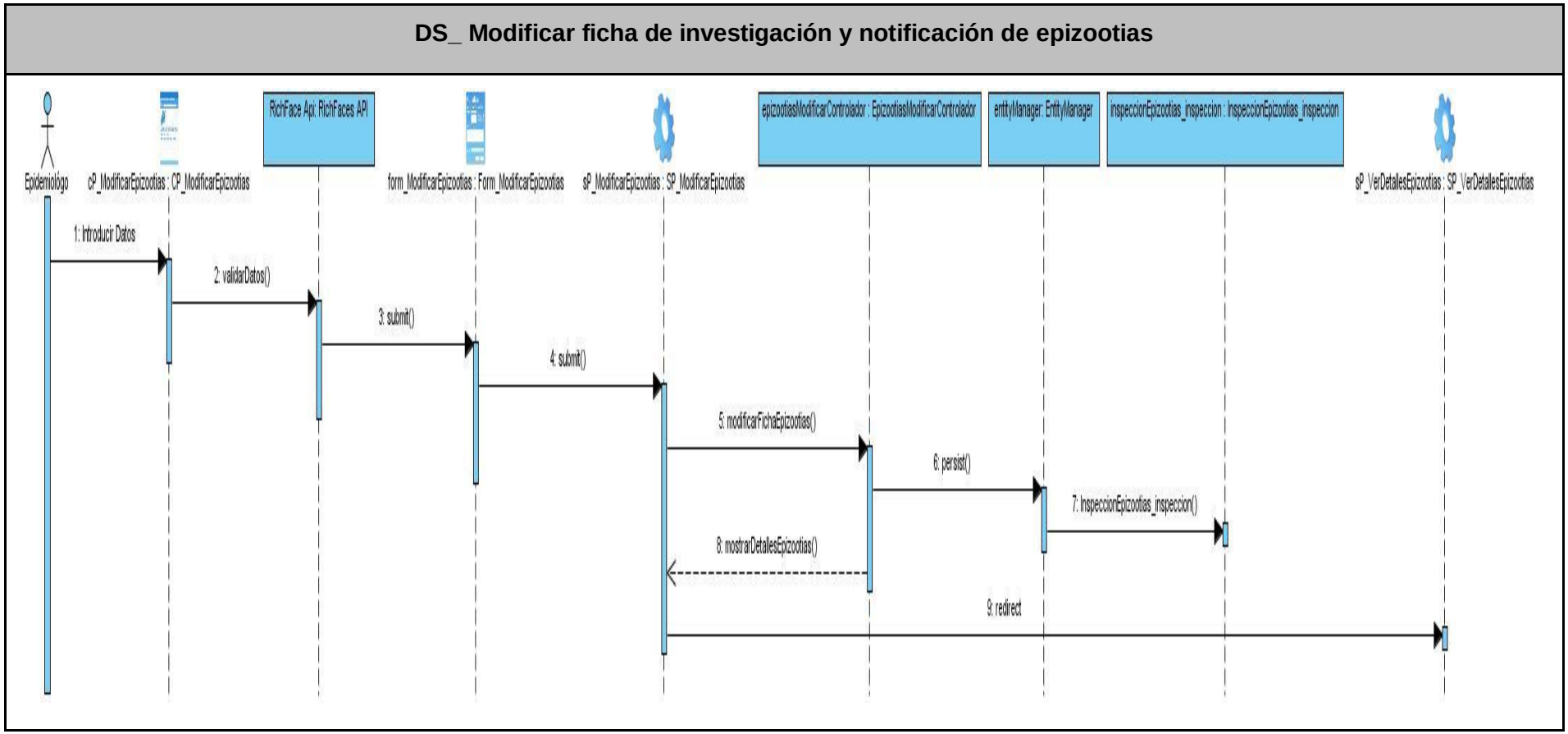

Figura 3.4 Diagrama de Secuencia – Modificar ficha de investigación y notificación de epizootias.

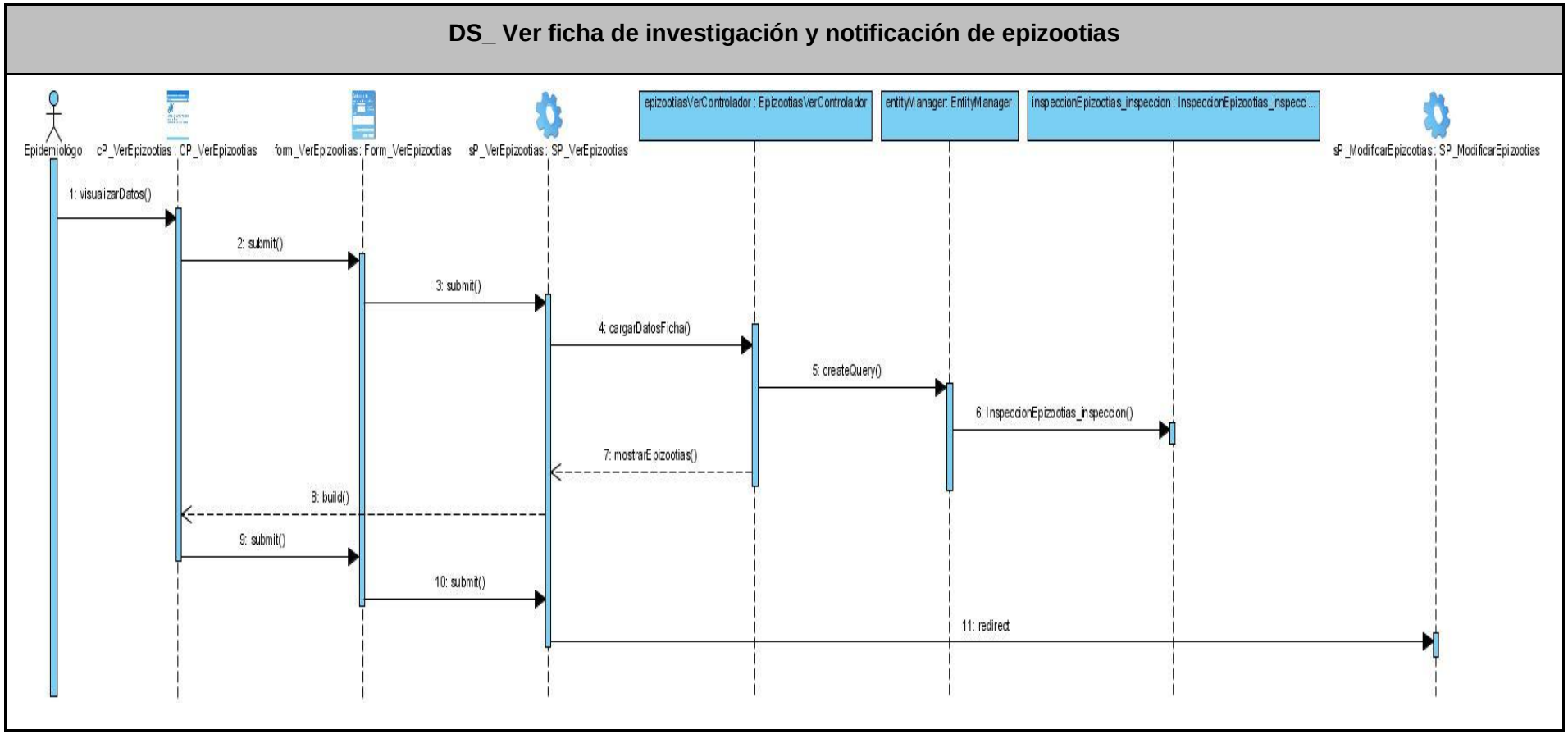

Figura 3.5 Diagrama de Secuencia – Ver ficha de investigación y notificación de epizootias.

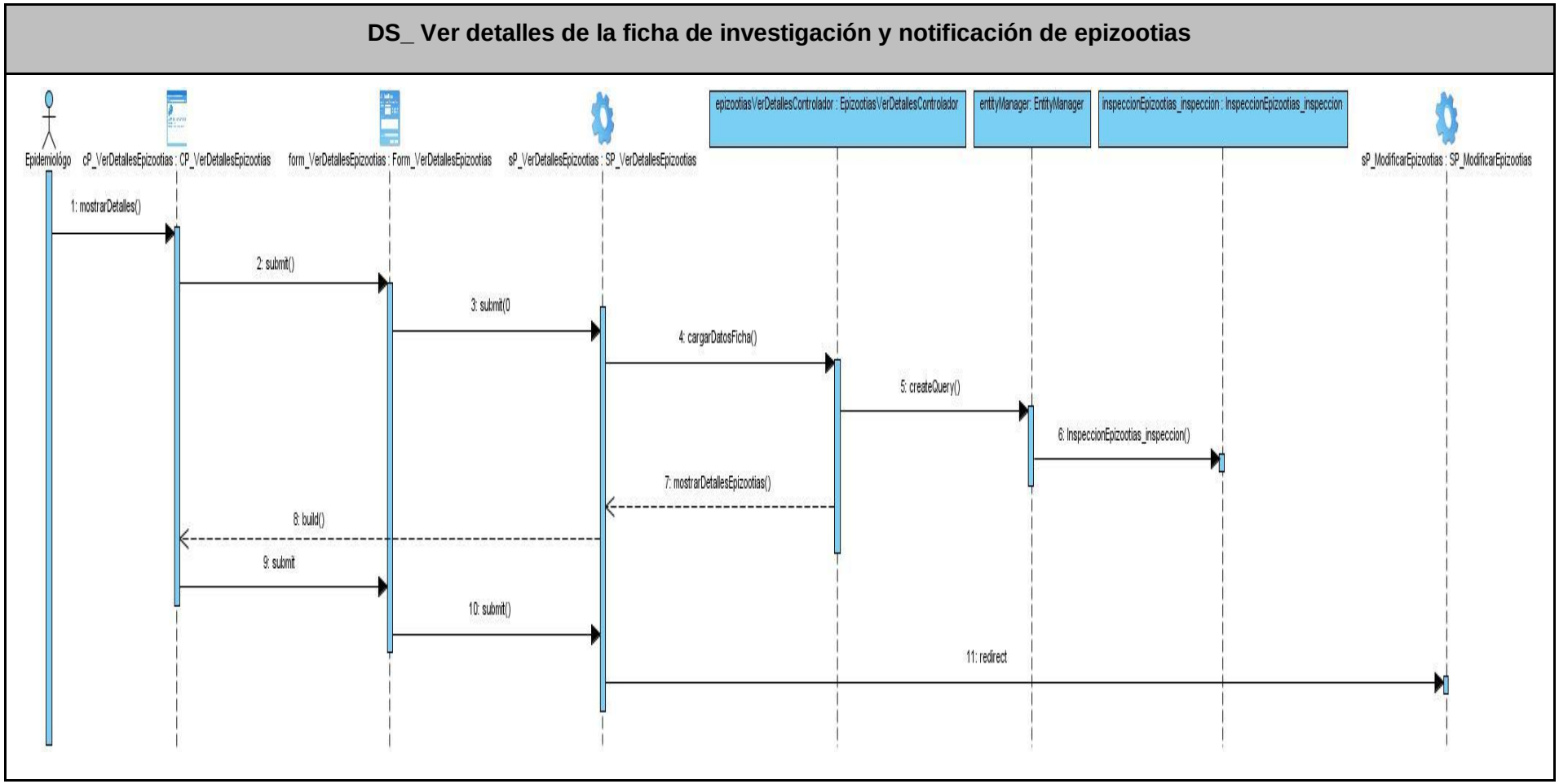

Figura 3.6 Diagrama de Secuencia – Ver detalles de la ficha de investigación y notificación de epizootias.

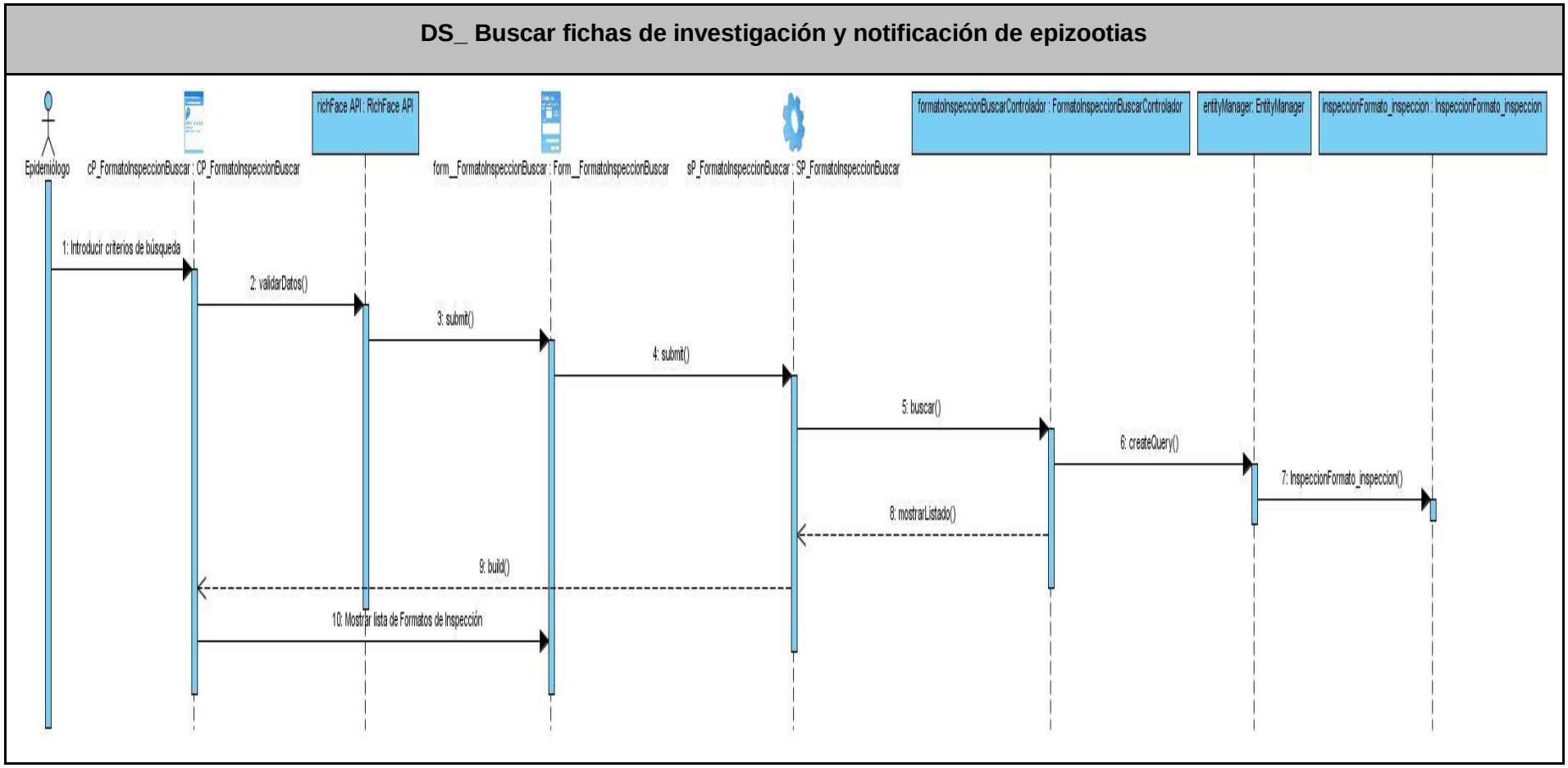

Figura 3.7 Diagrama de Secuencia – Buscar ficha de investigación y notificación de epizootias.

3.2 Descripción de las nuevas clases u operaciones necesarias.

Nombre: EpizootiasCrearControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	cargarDatosFicha() : void
Descripción:	Carga los nomencladores y otros datos necesarios para crear la ficha
Nombre:	crearFichaEpizootias() : void
Descripción:	Crea la ficha y persiste todos sus datos
Nombre:	cancelar() : void
Descripción:	Cancela accion
Nombre:	estados() : List<java.lang.String>
Descripción:	Devuelve un listado de los estados de venezuela
Nombre:	hallarMunicipiosDadoEstado() : List<java.lang.String>
Descripción:	Devuelve un listado de los municipios dado un estado
Nombre:	hallarLocalidadesDadoMuni() : List<java.lang.String>
Descripción:	Devuelve un listado de las localidades dado un municipio
Nombre:	listaTipoAnimal() : List<java.lang.String>

Descripción:	Devuelve un listado de tipos de animales
Nombre:	listaEspecies() : List<java.lang.String>
Descripción:	Devuelve un listado de tipos de especies
Nombre:	adicionarAnimal() : void
Descripción:	Adiciona animal a la lista de seleccionados
Nombre:	eliminarAnimal(animal : gehos.epidemiologia.entity.inspeccionSanitaria.InspeccionAnimal_inspeccion) : void
Descripción:	Elimina animal de la lista de seleccionados
Nombre:	listaMuestras() : List<java.lang.String>
Descripción:	Devuelve un listado de las muestras posibles
Nombre:	listaLaboratorios() : List<java.lang.String>
Descripción:	Devuelve un listado de los laboratorios
Nombre:	adicionarMuestra() : void
Descripción:	Adiciona muestra a la lista de las seleccionadas
Nombre:	eliminarMuestra(muestra : gehos.epidemiologia.entity.inspeccionSanitaria.InspeccionMuestra_inspeccion) : void
Descripción:	Elimina muestra a la lista de las seleccionadas

Nombre:	esOtraEspecie() : java.lang.Boolean
Descripción:	Devuelve si es de otra especie
Nombre:	esPoblacionMenor() : java.lang.Boolean
Descripción:	Devuelve si es de una poblacion menor

Tabla 3.1 Descripción de la clase EpizootiasCrearControlador.

Nombre: EpizootiasModificarControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	validarParametro() : void
Descripción:	Valida los parámetros que le llegan a la página
Nombre:	cargarDatosFicha() : void
Descripción:	Carga los nomencladores y los datosde la ficha necesarios para modificarla
Nombre:	modificarFichaEpizootias() : void
Descripción:	Modifica la ficha y persistiendo los cambios
Nombre:	cancelar() : void
Descripción:	Cancela la acción
Nombre:	estados() : List<java.lang.String>

Descripción:	Devuelve un listado de los estados de venezuela
Nombre:	hallarMunicipiosDadoEstado() : List<java.lang.String>
Descripción:	Devuelve un listado de los municipios dado un estado
Nombre:	hallarLocalidadesDadoMuni() : List<java.lang.String>
Descripción:	Devuelve un listado de las localidades dado un municipio
Nombre:	listaTipoAnimal() : List<java.lang.String>
Descripción:	Devuelve un listado de los tipos de animales
Nombre:	listaEspecies() : List<java.lang.String>
Descripción:	Devuelve un listado de los tipos de especies
Nombre:	adicionarAnimal() : void
Descripción:	Adiciona animal a la lista de seleccionados
Nombre:	eliminarAnimal(animal : gehos.epidemiologia.entity.inspeccionSanitaria.InspeccionAnimal_inspeccion) : void
Descripción:	Elimina animal de la lista de seleccionados
Nombre:	listaMuestras() : List<java.lang.String>
Descripción:	Devuelve un listado de las muestras posibles
Nombre:	listaLaboratorios() : List<java.lang.String>
Descripción:	Devuelve un listado de los laboratorios

Nombre:	adicionarMuestra() : void
Descripción:	Adiciona muestra a la lista de las seleccionadas
Nombre:	eliminarMuestra(muestra : gehos.epidemiologia.entity.inspeccionSanitaria.InspeccionMuestra_inspeccion) : void
Descripción:	Elimina muestra de la lista de las seleccionadas
Nombre:	esOtraEspecie() : java.lang.Boolean
Descripción:	Devuelve si es de otra especie
Nombre:	esPoblacionMenor() : java.lang.Boolean
Descripción:	Devuelve si es de una poblacion menor
Nombre:	fechasIguales(fecha1 : java.util.Date, fecha2 : java.util.Date) : java.lang.Boolean
Descripción:	Devuele si las fechas son iguales

Tabla 3.2 Descripción de la clase EpizootiasModificarControlador

Nombre: EpizootiasVerControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	validarParametro() : void

Descripción:	Valida los parámetros que le llegan a la página
Nombre:	cargarDatosFicha() : void
Descripción:	Carga los datos de la ficha
Nombre:	modificar() : void
Descripción:	Redirecciona a la página de modificar
Nombre:	salir() : void
Descripción:	Sale de esta acción

Tabla 3.3 Descripción de la clase EpizootiasVerControlador.

Nombre: EpizootiasVerDetallesControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	validarParametro() : void
Descripción:	Valida los parámetros que le llegan a la página
Nombre:	cargarDatosFicha() : void
Descripción:	Carga los datos de la ficha
Nombre:	modificar() : void
Descripción:	Redirecciona a la página de modificar

Nombre:	salir() : void
Descripción:	Sale de esta acción

Tabla 3.4 Descripción de la clase EpizootiasVerDetallesControlador.

Nombre: EpizootiasBuscarControlador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	comenzar() : void
Descripción:	Da inicio a la conversación
Nombre:	buscar() : void
Descripción:	Busca las fichas según los criterios de búsquedad
Nombre:	seleccionarFichaVer(ficha : gehos.epidemiologia.entity.inspeccionSanitaria.InspeccionEpizootias_inspeccion) : void
Descripción:	Selecciona la ficha
Nombre:	seleccionarFichaEliminar(ficha : gehos.epidemiologia.entity.inspeccionSanitaria.InspeccionEpizootias_inspeccion) : void
Descripción:	Selecciona la ficha

Nombre:	eliminarFicha() : void
Descripción:	Elimina la entidad seleccionada
Nombre:	cancelar() : void
Descripción:	Cancela esta acción

Tabla 3.5 Descripción de la clase EpizootiasBuscarControlador.

3.3 Modelo de datos

Un Modelo de datos es un conjunto de conceptos, reglas y convenciones para describir distintos aspectos del negocio del sistema. Requiere conocimiento, normas y estándares que aseguren la correcta descripción e interpretación de dichas técnicas. Su objetivo es producir cierta descripción estructurada de la organización y el negocio del cliente para permitir construir un sistema. Existen varias clasificaciones de los modelos de datos, dentro de los cuales podemos mencionar los modelos de datos conceptuales, los de datos lógicos y los de datos físicos.

Al modelar un sistema, la técnica del modelado Entidad - Relación mediante diagramas o modelos Entidad - Relación (denominado por sus siglas, E-R “Entity relationship”, o, "DER" Diagrama de Entidad Relación) brindan comodidades, ya que estos constituyen una representación conceptual de la información, mediante la cual se pretende “visualizar” los objetos que pertenecen a la Base de Datos como entidades (se corresponde al concepto de objeto de la Programación Orientada a Objetos) las cuales tienen atributos y se vinculan mediante relaciones. Además, estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades. Seguidamente se muestran los DER pertenecientes a los procesos Realizar Inspección Sanitaria e Interconsulta, así como la descripción de la taba de la base de datos:

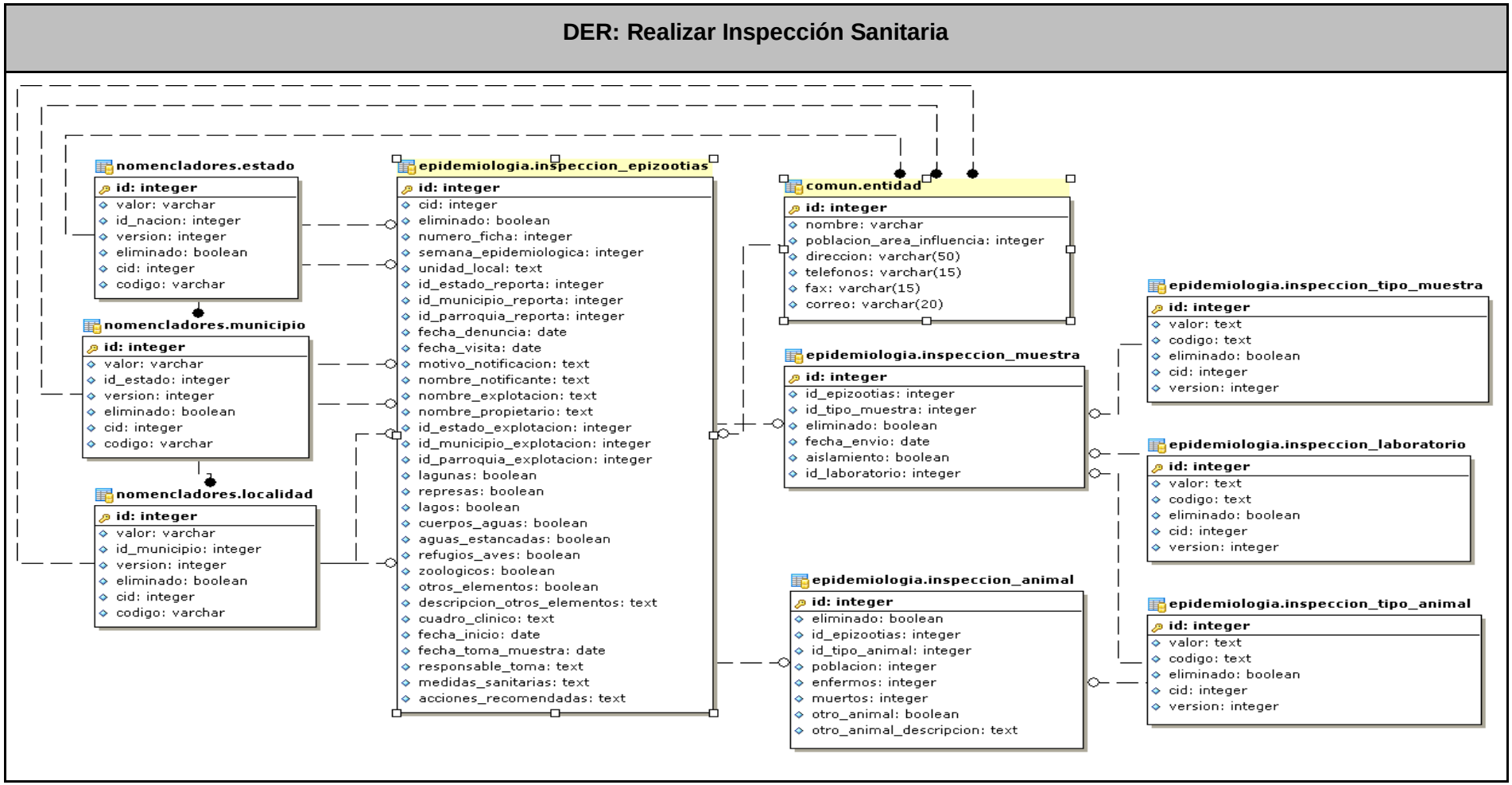

Figura 3.8 Diagrama Entidad – Relación - Realizar Inspección Sanitaria.

Nombre: inspeccion_epizootias		
Descripción:		
Atributo	**Tipo**	**Descripción**
id	SERIAL	Identificador
cid	INTEGER	Identificador de registro de la bitácora de sucesos
eliminado	BOOLEAN	Si fue eliminado del registro
numero_ficha	INTEGER	Número de la ficha
semana_epidemiologica	INTEGER	Número de la semana epidemiológica
unidad_local	TEXT	Nombre del establecimiento
id_estado_reporta	INTEGER	Identificador del estado que reporta
id_municipio_reporta	INTEGER	Identificador del municipio que reporta
id_parroquia_reporta	INTEGER	Identificador de la parroquia que reporta
fecha_denuncia	DATE	Fecha de la denuncia
fecha_visita	DATE	Fecha de la visita
motivo_notificacion	TEXT	Motivo de la notificación
nombre_notificante	TEXT	Nombre del notificante
nombre_explotacion	TEXT	Nombre de la explotación
nombre_propietario	TEXT	Nombre del propietario
id_estado_explotacion	INTEGER	Estado de la explotación

id_municipio_explotacion	INTEGER	Municipio de la explotación
id_parroquia_explotacion	INTEGER	Parroquia de la explotación
lagunas	BOOLEAN	Si existen lagunas
represas	BOOLEAN	Si existen represas
lagos	BOOLEAN	Si existen lagos
cuerpos_aguas	BOOLEAN	Si existen cuerpos de aguas
aguas_estancadas	BOOLEAN	Si existen aguas estancadas
refugios_aves	BOOLEAN	Si existen refugios de aves
zoologicos	BOOLEAN	Si existen zoológicos
otros_elementos	BOOLEAN	Si existen otros elementos
descripcion_otros_elementos	TEXT	Descripción de otros elementos
cuadro_clinico	TEXT	Cuadro clínico observado
fecha_inicio	DATE	Fecha de inicio
fecha_toma_muestra	DATE	Fecha de toma de la muestra
responsable_toma	TEXT	Nombre del responsable de la toma
medidas_sanitarias	TEXT	Medidas sanitarias aplicadas
acciones_recomendadas	TEXT	Acciones recomendadas

Tabla 3.6 Descripción de la tabla inspeccion_epizootias.

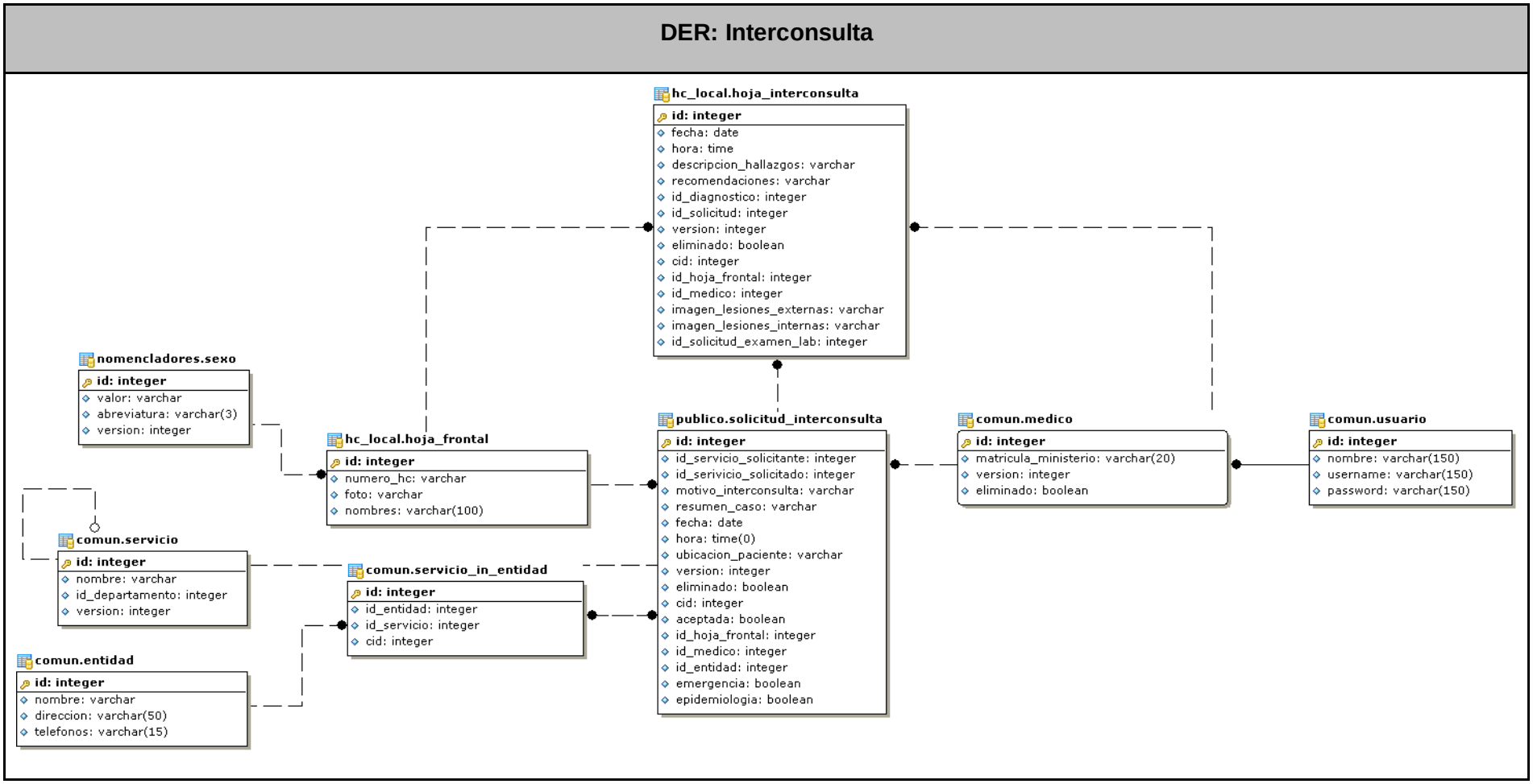

Figura 3.9 Diagrama Entidad – Relación - Interconsulta.

Nombre: hoja_interconsulta		
Descripción:		
Atributo	**Tipo**	**Descripción**
id	SERIAL	Identificador que identifica la hoja de consulta
id_indicaciones_medicas	INTEGER	Identificador de las indicaciones médicas
id_constancia	INTEGER	Identificador de la constancia
id_referencia	INTEGER	Identificador de la referencia
id_contrarreferencia	INTEGER	Identificador de la contrarreferencia
id_orden_admision	INTEGER	Identificador de la orden de admisión emitida
id_solicitud_analisis_lab	INTEGER	Identificador de la solicitud del análisis de laboratorio
id_solicitud_examen_tipiaje	INTEGER	Identificador de la solicitud del examen
id_solicitud_intervencion_q	INTEGER	Identificador de la solicitud de intervención
id_cita_consulta	INTEGER	Identificador de la cita de consulta
fecha	DATE	Fecha de creación
id_diagnostico	INTEGER	Identificador del diagnóstico médico
observaciones	TEXT	Observaciones emitidas
conclusiones	TEXT	Conclusiones del médico
id_diagnostico_princ_final	INTEGER	Identificador del diagnóstico final
id_hoja_dermatologia	INTEGER	Identificador de la hoja de dermatología

id_hoja_general	INTEGER	Identificeador de la hoja general
id_hoja_hem_onc_pediatrica	INTEGER	Identificador de de la hoja de pediatría
id_hoja_radioterapia	INTEGER	Identificador de la hoja de radio terapia
id_hoja_triaje	INTEGER	Identificador de la hoja de triaje
id_hoja_cirugia_colo_rectal	INTEGER	Identificador de la hoja de cirugía colo rectal
id_hoja_ginecologia	INTEGER	Identificador de la hoja de ginecología
id_hoja_neurologia	INTEGER	Identificador de la hoja de neurología
id_hoja_oftalmologia	INTEGER	Identificador de la hoja de oftalmología
id_hoja_uro_genital	INTEGER	Identificador de la hoja urogenital
id_hoja_gastroenterologia	INTEGER	Identificador de la hoja de gastroenterología
id_hoja_hem_oncologica	INTEGER	Identificador de la hoja oncología
id_hoja_obstetricia	INTEGER	Identificador de la hoja de obstetricia
id_hoja_orl	INTEGER	Identificador de la hoja de orl
id_hoja_traumatologia	INTEGER	Identificador de la hoja de traumatología
motivo	TEXT	Motivo de la interconsulta
enfermedad_actual	TEXT	Enfermedad actual
hora_fin	TIME	Hora en que termina
id_cita_origen	INTEGER	Identificador de la cita de origen
version	INTEGER	Versión del registro

eliminado	BOOLEAN	Si fue eliminado
cid	INTEGER	Identificador de registro de la bitácora de sucesos
id_cita_triaje_gral	INTEGER	Identificador de la cita
id_tipo_hoja	INTEGER	Identificador del tipo de hoja
id_especialidad	INTEGER	Identificador de la especialidad
hora_inicio	TIME	Hora de inicio
id_entidad	INTEGER	Identificador de la entidad donde se realiza

Tabla 3.7 Descripción de la tabla hoja_interconsulta.

3.4 Breve valoración de las técnicas de validación (Integridad y Normalización de la Base de datos)

Cuando se crea un sistema, se debe asegurar que tenga una respuesta ante cualquier circunstancia posible, para lograrlo se realizan técnicas de validación aunque es difícil disponer de datos de entrenamiento para simular todas las circunstancias posibles, se deben generalizar las situaciones a través de los datos disponibles. Existe un gran número de técnicas de validación, las cuales deben de estar dirigidas al supuesto comportamiento erróneo que tendrá el sistema. Para evitar sucesos imprevistos es que se ponen en práctica métodos y estrategias que facilitan todo este trabajo.

Uno de los puntos a tener en cuenta en las técnicas de validación son las excepciones y el tratamiento de errores, las cuales no son más que eventos en tiempo de ejecución que pueden causar que una funcionalidad fracase. Las excepciones aparecen ante un error u otro evento capaz de interrumpir la ejecución normal de un programa. Se encargan de transferir el control de un programa en el punto donde ocurre la excepción al manipulador adecuado, posibilitando que continúe la ejecución del mismo aún en presencia de un error.

En el sistema el control de las excepciones se lleva a cabo en todas las porciones de código donde pueda surgir alguna situación imprevista y además, donde se ejecutan sentencias que manipulan los datos que

viajan entre la aplicación y la base de datos. También se controlan la validación de algunos datos provenientes de la interfaz de usuario, puesto que encierran una lógica compleja.

El archivo XML denominado *page.xml*, engloba la configuración de todos los mensajes que se deben mostrar por cada tipo de excepción, así como la página a la que el sistema redirecciona en caso de la aparición de un error sorpresivo.

La base de datos del sistema presenta una gran cantidad de información, la cual está dividida en procesos y en entidades. Se aplicaron estrategias con el objetivo de lograr un mayor entendimiento y organización de los procesos: Realizar Inspección Sanitaria, Interconsultas y Reporte, así como en sus entidades correspondientes.

En el caso del proceso de Realizar Inspección Sanitaria, pertenecientes al módulo de Epidemiología, todas las entidades vinculadas a este proceso están precedidas de la palabra “inspección_”, seguidas del nombre de la entidad. En el caso de que exista alguna entidad que tenga un tipo asociado o alguna característica específica, se le pondrá la siguiente notación: ” inspección_”, seguido del nombre de la entidad y luego “_” y el tipo asociado.

3.5 Vista de Implementación. (Diagrama de Componentes)

En la vista de implementación se desarrollan diferentes artefactos, el diagrama de componentes es uno de ellos. Este ilustra las piezas del software, controladores y embebidos que conformarán un sistema. Los mismos están conformados por dependencias lógicas entre componentes del software. También muestra cómo un sistema de software es dividido en componentes y las dependencias entre estos componentes. Se utiliza para describir la vista de implementación estática de un determinado sistema ya que presenta un nivel más alto de abstracción que un diagrama de clase usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción y eventualmente un componente puede comprender una gran porción de un sistema.

En la creación de este diagrama se tienen en cuenta los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en la implementación.

Seguidamente se muestra el diagrama de componente perteneciente al módulo de Epidemiología.

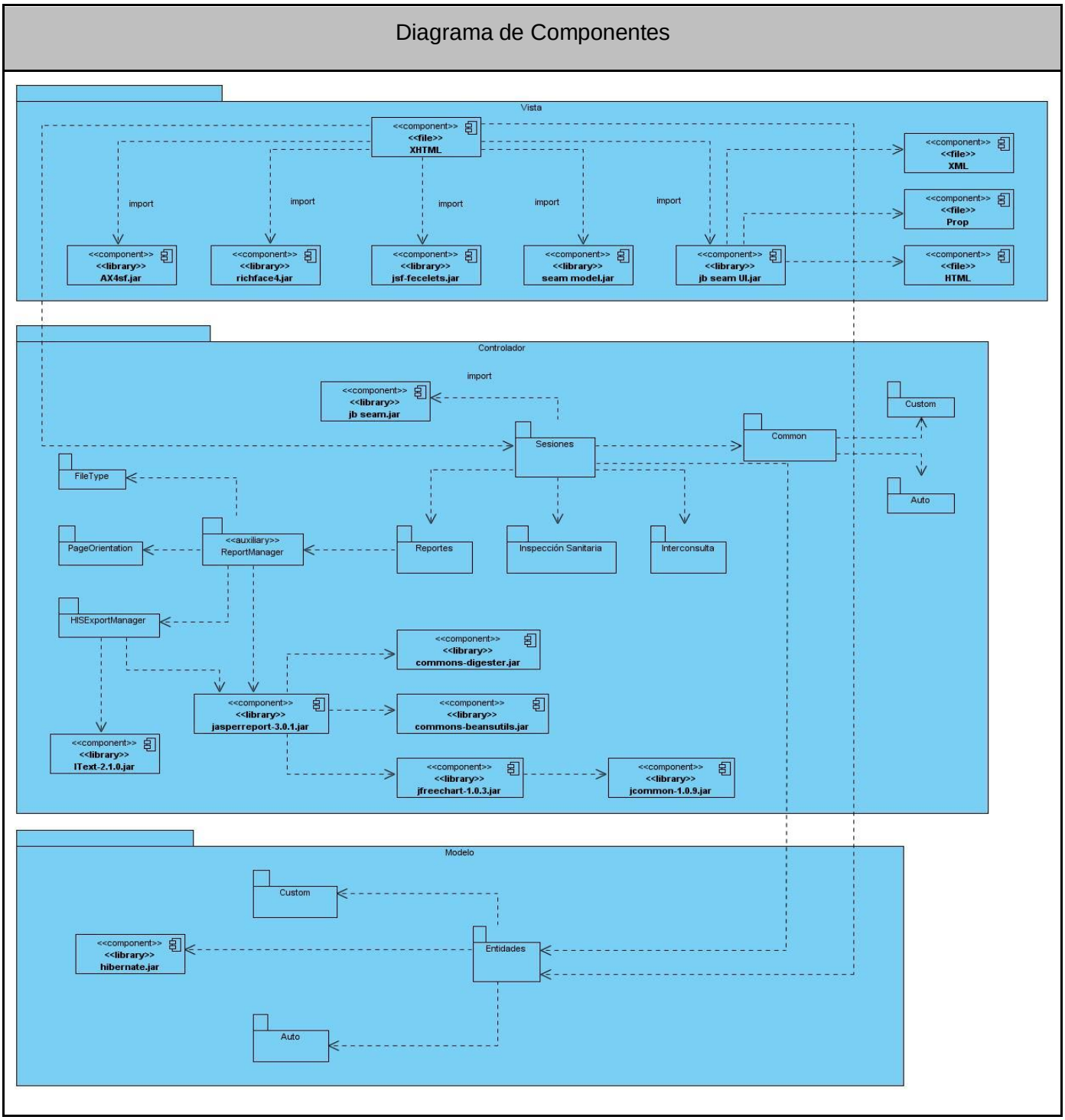

Figura 3.10 Diagrama de Componente.

En este capítulo, se obtuvo el modelo de datos correspondiente a los procesos que se desarrollan como parte de la investigación. Este constituyó uno de los puntos de entrada para la realización de las actividades concernientes al flujo de trabajo de implementación. De esta forma, se elaboraron los artefactos correspondientes, como son el modelo de implementación el cual engloba el diagrama de despliegue y el diagrama de componentes.

CAPÍTULO 4: MODELO DE PRUEBA

El desarrollo de un producto de software implica la realización de una serie de actividades encaminadas a encontrar errores. Es por ello que se deben incorporar acciones que evalúen la calidad del producto que se está desarrollando. Dentro del proceso de desarrollo de un software, el flujo de trabajo de Prueba, es mediante el que se puede validar que las suposiciones hechas en el diseño y los requerimientos se estén cumpliendo satisfactoriamente, por lo que se encarga de verificar que el producto funcione como se diseñó y que los requerimientos se cumplan cabalmente.

Este flujo de trabajo brinda soporte para encontrar, documentar y solucionar defectos en el sistema, por lo debe estar presente en todo el ciclo de vida del desarrollo del sistema para ir refinándolo paulatinamente y no al final del mismo.

Prueba es un proceso de ejecución de un programa con la intención de descubrir un error no detectado hasta entonces. Los casos de pruebas no pueden asegurar la ausencia de errores, sólo puede demostrar que existen defectos en el software.

Las pruebas se pueden realizar basándose en dos esquemas diferentes: demostrar a través de pruebas de caja blanca que las operaciones internas se ajustan a lo especificado y que los componentes internos andan bien, o mediante las pruebas de caja negra, conociendo la función del programa, intentar demostrar que las funciones están correctas.

4.1 Prueba de caja negra

Las técnicas o pruebas de caja negra se basan fundamentalmente en las entradas y salidas de datos que produce un sistema, sin importar el funcionamiento del código interno. Es decir, lo más importante será la forma en que el sistema interactúe con el medio, mostrando respuestas correctas en cada caso.

Estas pruebas se utilizan para demostrar que las funciones del software son operativas, que la entrada se acepta de forma correcta. Además se produce una salida correcta y la integridad de la información externa se mantiene.

Además, pretenden encontrar errores, tales como: funciones incorrectas o ausentes, errores en la interfaz, errores en estructuras de datos o en accesos a bases de datos, errores externos, errores de rendimiento y también errores de inicialización y de terminación.

Para la realización de las pruebas de caja negra existen varios métodos. Uno de los métodos es el del análisis de valores límite, el cual se basa en la combinación de diferentes valores de pruebas interesantes y su calidad depende de las habilidades del desarrollador. Además, se usa estrictamente para medir la calidad de un conjunto de casos de pruebas y los valores de dichas pruebas deben seleccionarse con anterioridad.

El método a utilizar en el presente trabajo es el de partición equivalente. Su funcionamiento se basa en que este divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada. En este método primeramente se identifican las clases de equivalencia y luego se identifican los casos de pruebas.

El propósito de un diseño de caso de prueba es definir una forma de probar el sistema en desarrollo, incluyendo las entradas de datos con las que se ha de probar, los resultados esperados y las condiciones bajo las que ha de probarse. Estos ayudan a validar y verificar las expectativas de los clientes. Seguidamente de muestran los diseños de casos de pruebas correspondientes a los Casos de Usos expuestos anteriormente:

Escenarios del ver Ficha de investigación y notificación de epizootias	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Ver Ficha de investigación y notificación de epizootias	Ver una Ficha de investigación y notificación de epizootias satisfactoriamente.	Se muestra la interfaz Ver Ficha de investigación y notificación de epizootias. Se muestran los datos de la Ficha de investigación y notificación de epizootias. Selecciona la opción Salir. Regresa a la vista de anterior.
EC 2:Modificar Ficha de investigación y notificación de epizootias	Permite acceder al caso de uso modificar Ficha de investigación y notificación de	Se muestra la interfaz Ver Ficha de investigación y notificación de epizootias.

	epizootias.	Se muestran los datos de la Ficha de investigación y notificación de epizootias. Se selecciona la opción Modificar. Ver DCP **Modificar Ficha de Ficha de investigación y notificación de epizootias**.

Tabla 4.1 Secciones a probar en el Caso de Uso: Ver Ficha de investigación y notificación de epizootias.

Id del escenario	EC 1	EC 2
Escenario	Ver Ficha de investigación y notificación de epizootias satisfactoriamente	Modificar Ficha de investigación y notificación de epizootias
Salir	NA	
Modificar		NA
Respuesta del Sistema	Regresa a la vista de anterior.	Muestra la interfaz para modificar.
Resultado de la Prueba		

Tabla 4.2 SC 1: Ver Ficha de investigación y notificación de epizootias.

En este capítulo, se estableció el método de prueba a utilizar, a partir del cual se crearon un conjunto de diseños de casos de prueba, lo que posibilitó evaluar las funcionalidades del sistema y detectar errores de las mismas.

CONCLUSIONES

Al lograr la implementación de los procesos de vigilancia epidemiológica referente al módulo de Epidemiología, se concluyó que:

- Los sistemas de estadísticas epidemiológicas a nivel mundial son propietarios y heterogéneos en su mayoría; por lo que su adquisición representaría un gran costo. Además, estos no cubren toda la gestión de los procesos del área de Epidemiología.
- La aplicación del patrón de diseño MVC resultó conveniente pues permitió la organización de las funcionalidades en los tres componentes de diseño según sus responsabilidades, disminuyendo la complejidad de implementación de los casos de uso.
- La adopción de las pautas de diseño de interfaz de usuario permitió que los nuevos casos de uso sean visualmente homogéneos a los existentes, obteniendo una aplicación visualmente uniforme.
- La implementación de los procesos: Realizar Inspección Sanitaria, Generar Reportes e Interconsulta contribuirá a un mejor control epidemiológico, aumentando la eficiencia de las acciones que tienen lugar en el área de Epidemiología de las instituciones hospitalarias.

RECOMENDACIONES

Se recomienda:

- Añadir al proceso Generar Reporte una funcionalidad que permita recoger antecedentes de morbilidad y mortalidad preexistentes, con lo cual el reporte "Generar situaciones de alertas y epidemias" pueda ser más exacto al considerar datos anteriores a la fecha de instalación del sistema.
- Especificar con un mayor grado de detalle los Formatos de Inspección Sanitaria de manera que queden completamente descritas las situaciones sanitarias de las instituciones hospitalarias.
- Utilizar estándares en la generación de alertas y notificaciones de modo que el sistema y sus clientes puedan beneficiarse de la reutilización de conocimiento médico existente en repositorios de estos estándares (Arden Syntax, por ejemplo).
- Rediseñar los casos de uso correspondientes a la generación de reportes para que permitan filtrar la información a través de varios criterios de búsqueda.
- Añadir funcionalidades que permitan configurar las unidades locales del área de Epidemiología específica para cada entidad.
- En la funcionalidad “Buscar formato de inspección sanitaria” mostrar solamente el “Ver formato de inspección sanitaria” ya que no se requieren las opciones de “Modificar" y "Eliminar formato de inspección sanitaria”.
- Al crear los Formatos de Inspección Sanitaria se debe brindar la posibilidad de visualizar los servicios no médicos solicitados por la entidad hospitalaria.

REFERENCIAS BIBLIOGRÁFICAS

1 Departamento de vigilancia epidemiológica. Servicios de Salud de Yucatán. Diciembre 2009.

Disponible en: *http://www.salud.yucatan.gob.mx/content/view/25/2/*

2 Organización Panamericana de la Salud. Febrero 2010.

Disponible en: *http://ais.paho.org/sigepi/index.asp*

3 Eclipse (Software). Mayo 2009.

Disponible en: *http://plataformaeclipse.com/*

4 JBoss Seam Framework. Febrero 2008

Disponible en: *http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/*

5 Instalación de JBoss Tools. Febrero 2010

Disponible en: *http://tratandodeentenderlo.blogspot.com/2009/07/instalacion-de-jboss-tools.html*

6 Junta de Andalucía. RichFaces. Febrero 2009

Disponible en: *http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces#caracteristicas%20Seam*

7 PostgreSQL Maestro 8.3. Marzo 2008.

Disponible en: *http://www.freedownloadscenter.com/es/Programacion/Base_de_Datos_y_Redes/PostgreSQL_Maestro.html* .

8 Red Hat Middleware Relational Persistence for Java and .NET.2009

Disponible en: *https://hibernate.bluemars.net/*

9 Informática Profesional. Febrero 2010

Disponible en: *http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=ireport*

10 Arnold, Ken; Gosling, James; Holmes, David. El lenguaje de Programación Java. Addison Wesley. 2009.

Disponible en: *http://www.lenguajes-de-programacion.com/programacion-java.shtml*

11 Metodologías de desarrollo. Febrero 2010

Disponible en: *http://www.marblestation.com/?p=644*

12 Gómez, Juan P. Fundamentos de la metodología RUP. Septiembre 2007.

Disponible en: *http://www.scribd.com/doc/297224/RUP*

13 Salinas, Patricio; Histchfeld, Nancy.Tutorial de UML. Octubre 2006.

Disponible en: *http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html*

BIBLIOGRAFÍA

1. ALTERNATIVA DE CÓDIGO ABIERTO. FEBRERO 2010

   DISPONIBLE EN: *http://www.osalt.com/es/jboss*

2. *Arnold, Ken; Gosling, James; Holmes, David. El lenguaje de Programación Java. Addison Wesley. 2009.*

   Disponible en: *http://www.lenguajes-de-programacion.com/programacion-java.shtml*

3. CASOS DE PRUEBA. MARZO 2010

   DISPONIBLE EN:*http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/prod_des/documento/casos_prueba_d.php*

4. CHACÓN, YUNAISY Y GARCÍA, OMAR. TESIS MÓDULO EPIDEMIOLOGÍA DEL SISTEMA DE INFORMACIÓN HOSPITALARIA ALAS HIS. UCI 2009

5. CURSO PRÁCTICO DE MODELADO DE NEGOCIOS CON UML Y BPMN. FEBRERO 2010

   DISPONIBLE EN: *http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm*

6. DEPARTAMENTO DE VIGILANCIA EPIDEMIOLÓGICA. SERVICIOS DE SALUD DE YUCATÁN. DICIEMBRE 2009.

   DISPONIBLE EN: *HTTP://WWW.SALUD.YUCATAN.GOB.MX/CONTENT/VIEW/25/2/*

7. DOSIDEAS. ABRIL 2010

   DISPONIBLE EN: *http://www.dosideas.com/noticias/java/592-primeros-pasos-con-drools.html*

8. Eclipse (Software). Mayo 2009.

   Disponible en: *http://plataformaeclipse.com/*

9. EPIDEMIOLOGÍA BÁSICA. ENERO 2010

   DISPONIBLE EN:*http://www.mailxmail.com/curso-epidemiologia-basica/sistema-vigilancia-epidemiologica*

10. Gómez, Juan P. Fundamentos de la metodología RUP. Septiembre 2007.

    Disponible en: *http://www.scribd.com/doc/297224/RUP*

11. GUÍA UBUNTU. FEBRERO 2010

    DISPONIBLE EN: *http://www.guia-ubuntu.org/index.php?title=PostgreSQL*

12. HISTORIA DE LA MEDICINA

DISPONIBLE EN HTTP://HISTORIADELAMEDICINA.ORG/BLOG/2006/11/22/ALEXANDER-DUNCAN-LANGMUIR-Y-LOS-CDC-1910-1993/

13. Informática Profesional. Febrero 2010

Disponible en: *http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=ireport*

14. Instalación de JBoss Tools. Febrero 2010

Disponible en: *http://tratandodeentenderlo.blogspot.com/2009/07/instalacion-de-jboss-tools.html*

15. JASPERREPORT INFORMACIÓN DEL PROYECTO. FEBRERO 2010

DISPONIBLE EN: *http://jasperforge.org/projects/jasperreports*

16. JBOSS COMMUNITY. ABRIL 2010

DISPONIBLE EN: *http://www.jboss.org/jbpm*

17. JBoss Seam Framework. Febrero 2008

Disponible en: *http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/*

18. Junta de Andalucía. RichFaces. Febrero 2009

Disponible en:
http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces#caracteristicas%20Seam

19. LA VIGILANCIA EN SALUD. ELEMENTOS BÁSICOS QUE DEBE CONOCER EL MÉDICO DE FAMILIA. FEBRERO 2010

DISPONIBLE EN: *http://bvs.sld.cu/revistas/mgi/vol18_1_02/mgi11102.htm*

20. LEYVA, YAREL. CONTROL SANITARIO INTERNACIONAL. 2009

21. MATEO, PEDRO REALES. CALIDAD DE CASOS DE PRUEBA

DISPONIBLE EN: *http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Pedro%20Reales%20Expo.pdf*

22. MERINDE. PRUEBA. MARZO 2010

DISPONIBLE EN:
http://MERINDE.RINDE.GOB.VE/INDEX.PHP?OPTION=COM_CONTENT&TASK=VIEW&ID=139&ITEMID=194

23. Metodologías de desarrollo. Febrero 2010

Disponible en: *http://www.marblestation.com/?p=644*

24. MONOGRAFIA.COM. MARZO 2010

DISPONIBLE EN: *http://www.monografias.com/trabajos5/andi/andi.shtml*

25. MUNDO GEEK. MARZO 2010

DISPONIBLE EN: *http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/*

26. ORGANIZACIÓN PANAMERICANA DE LA SALUD. FEBRERO 2010.

Disponible en: *http://ais.paho.org/sigepi/index.asp*

27. PESQUISA EN BASES DE DATOS. FEBRERO 2010.

DISPONIBLE EN: *http://bases.bireme.br/cgi-bin/wxislind.exe/iah/online/?IsisScript=iah/iah.xis&src=google&base=LILACS&lang=p&nextAction=lnk&exprSearch=477297&indexSearch=ID*

28. PostgreSQL Maestro 8.3. Marzo 2008.

Disponible en: *http://www.freedownloadscenter.com/es/Programacion/Base_de_Datos_y_Redes/PostgreSQL_Maestro.html* .

29. RATIONAL UNIFIED PROCESS. FEBRERO 2010

DISPONIBLE EN: *http://www.rational.com.ar/herramientas/rup.html*

30. Red Hat Middleware Relational Persistence for Java and .NET.2009

Disponible en: *https://hibernate.bluemars.net/*

31. REYNOSO, CARLOS B. INTRODUCCIÓN A LA ARQUITECTURA DE SOFTWARE. 2004

DISPONIBLE EN: *http://www.willydev.net/descargas/prev/IntroArq.pdf*

32. Salinas, Patricio; Histchfeld, Nancy.Tutorial de UML. Octubre 2006.

Disponible en: *http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html*

33. SECCIÓN DE PARADIGMA VISUAL UML. FEBRERO 2010

DISPONIBLE EN: *http://www.freedownloadmanager.org/es/downloads/paradigma_visual_uml_libre/*

34. SISTEMA PARA GESTIÓN DE INFORMACIÓN EPIDEMIOLÓGICA PARA ENTORNOS DE CONECTIVIDAD LIMITADA. 2008

DISPONIBLE EN: *http://www.revistaesalud.com/index.php/revistaesalud/article/viewArticle/251/571*

35. TUTORIAL DE JAVA-ARQUITECTURA MVC. FEBRERO 2010

DISPONIBLE EN: *http://sunsite.dcc.uchile.cl/java/docs/JavaTut/Apendice/arq_mvc.html*