Universidad de las Ciencias Informáticas

Facultad 7

Tema: Configuraciones históricas en el módulo Estadísticas

Título: Configuraciones históricas en el módulo Estadísticas del Sistema de Información Hospitalaria alas HIS

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

Autor: Danilo Marin Mesa

Tutores: Ing. Alain Ramos Medina

Ing. Omar García Bonelly

La Habana, junio de 2011
"Año del 53 de la Revolución"

Datos de contacto

Ing. Alain Ramos Medina

Graduado de Ingeniero en Ciencias Informáticas en el año 2007 en la Universidad de las Ciencias Informáticas. Actualmente se desempeña como profesor en la Universidad de las Ciencias Informáticas (UCI) donde ha impartido asignaturas como Introducción a la Programación, Programación I y Aplicaciones Informáticas del Sector de la Salud (2do perfil). Posee la categoría docente de Instructor. Se encuentra vinculado a actividades productivas en el área hospitalaria. Se desempeña como Jefe de desarrollo del módulo Estadísticas del Sistema de Información Hospitalaria alas HIS perteneciente al Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM).

Correo electrónico: aramos@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

En las instituciones hospitalarias se maneja mucha información referente a las estadísticas, lo cual es fundamental para poder generar reportes basados en estos datos. En el área de estadísticas de un centro hospitalario es necesario conocer datos históricos para poder generar dichos reportes. El objetivo del presente trabajo de diploma es desarrollar las configuraciones históricas en el módulo Estadísticas del Sistema de Información Hospitalaria alas HIS.

Con la implementación de estas nuevas funcionalidades en el módulo Estadísticas, se logrará una mayor veracidad de los reportes, además de facilitar la gestión y control de la información referente a las estadísticas del centro. Contribuyendo a elevar los beneficios que el sistema brinda tanto al personal médico como a los pacientes.

Las tecnologías utilizadas para el desarrollo son libres, garantizando así la independencia tecnológica de la solución.

Palabras claves:

Configuraciones históricas, estadísticas, reportes.

Tabla de contenido

Introducción

Por ser el activo fundamental para el crecimiento de una organización, la información es hoy, el recurso más importante para cualquier empresa. El flujo de la información y el conocimiento presenta un gran impacto en los resultados de las organizaciones, al ser estos apoyos fundamentales, para el aumento de la calidad en los servicios que prestan.

El ser humano en su desarrollo ha ido generando grandes volúmenes de información, y con ella la necesidad asociada de almacenarla y procesarla, por ende, con el paso del tiempo, ha perfeccionado los mecanismos para su manipulación. Son empleados diversos métodos para garantizar su integridad, disponibilidad y organización. Dependiendo de las necesidades y el desarrollo tecnológico, dichas prácticas han sido mantenidas o sustituidas por prácticas de mayor eficiencia.

A partir de la segunda mitad del siglo XX, el surgimiento de las computadoras y con ellas los sistemas informáticos, posibilitó una mejor gestión de la información. Además, propició el aumento de la calidad de los procesos llevados a cabo por el hombre en su vida cotidiana, generando mayores índices de productividad en cuanto a esfuerzo y resultados de trabajo.

La apropiación de estas tecnologías es parte de una nueva era; la era del conocimiento, en la que la mayoría de los trabajadores se consideran "trabajadores del conocimiento". La informática es usada para simplificar el trabajo en muchas instituciones que generan beneficios a la humanidad. El eliminar gran parte de los errores humanos derivados del uso de archivos, libros u otros mecanismos de control de la información permite agilizar los procesos productivos. Por las ventajas que ofrece para la sociedad en sectores fundamentales como: la cultura, el deporte, la recreación, los negocios, la educación, la política, entre muchos otros, es priorizado su desarrollo.

En la medicina uno de los renglones socio-económicos de mayor importancia, los conocimientos y las opiniones del personal especializado, así como los estudios realizados, son primordiales por la necesidad de utilizar la experiencia para lograr mejores resultados. En la calidad de los cuidados médicos intervienen diferentes entidades que necesitan comunicarse, compartir e intercambiar información. Un diagnóstico rápido y el tratamiento eficaz de un paciente dependen de la transmisión exacta de órdenes y resultados entre varios servicios hospitalarios.

El manejo de información para respaldar la toma de decisiones en forma efectiva y oportuna es fundamental en la salud, y es la informatización del sector una posible respuesta a las necesidades emergentes del mismo. La formación de profesionales en el campo de los registros médicos, las ciencias de la información y la informática médica, es una práctica internacional; ellos, como parte de

la convergencia profesional y tecnológica requieren una alta calificación en el manejo de las tecnologías de la información para administrar las condiciones en que se comparte el conocimiento. Es por ello que en el sector de la salud toma gran auge el uso de la informática como apoyo para garantizar un buen servicio.

Los hospitales como actores principales del sistema sanitario, generan un importante volumen de datos. En muchos casos dicha información no está disponible en tiempo y forma necesarios, ya que las personas encargadas de su gestión utilizan los sistemas tradicionales imposibilitando que sea óptimo el manejo de la misma.

El funcionamiento de un centro hospitalario, al igual que en cualquier empresa, depende de lo bien que se utilicen sus recursos. La maquinaria, material diverso, el dinero, las personas y la información son todos recursos; pero la información es, quizás, el recurso más valioso que tiene, sin él todos los demás recursos quedan aislados e inmanejables y la información como recurso, es abundante en un hospital. Una entidad hospitalaria es el marco ideal para la tecnología de comunicaciones informatizada que se utiliza tan ampliamente en las empresas de gestión.

No existe un campo con una necesidad tan grande de proceso automático de datos como el de la asistencia sanitaria. Las mejoras en la eficiencia que se pueden realizar con un sistema que procesa datos, funcionando adecuadamente, se pueden medir en miles de horas/hombre, en mejores dividendos para la entidad y en un aumento considerable de la calidad del servicio que presta la entidad y por consiguiente, mayores índices de impacto en la salud de los pacientes que atiende. De hecho, la instalación de tecnología informática en los hospitales puede ser una herramienta muy importante en la reducción de costes y el aprovechamiento de los recursos, en aras de mejorar la eficacia de la atención hospitalaria.

Debido a la creciente necesidad de almacenar datos vitales, que contribuyan a aumentar el nivel de vida de la sociedad, se apuesta por la creación de sistemas que permitan la gestión de la información hospitalaria. Con el paso de los años, se ha ido estandarizando la creación de los denominados Sistemas de Información Hospitalarios (Hospitals Information Systems) llamados HIS por sus siglas en inglés, término más utilizado en la bibliografía sobre el tema.

Un HIS es un entorno integrado de datos comunes, permite que la información que se origina en diferentes unidades funcionales independientes sea compartida por todos; o bien un entorno distribuido en el cual cada unidad funcional administra y procesa los datos de interés local; así como los sistemas de uso común. Son sistemas de información para el beneficio de un hospital, para ello los datos son coherentemente almacenados en una base de datos, de donde son puestos a disposición de usuarios

autorizados en el lugar y momento en que los datos son requeridos, en un formato adecuado a las necesidades específicas del usuario.

Estos sistemas permiten mejorar la práctica clínica, apoyar las tareas de docencia e investigación, obtener estadísticas detalladas y confiables y armonizar la información científico-técnica con la administrativo-contable. Una adecuada capacitación y aplicación de los recursos en el área de la Informática Médica obliga a controlar la calidad, cantidad, oportunidad y accesibilidad de la información, así como a aumentar el conocimiento y sentido de pertenencia a la organización del personal, haciendo del hospital un centro que aproveche al máximo sus recursos a favor de la atención de los pacientes.

Dentro de un HIS, de vital importancia son las estadísticas, encargadas de recolectar, analizar, presentar e interpretar datos del centro. Resumiendo la información en tablas, gráficos e indicadores, que permiten la fácil compresión de las características concernientes a las distintas áreas. El hecho de poder organizar y determinar tendencias hace de esta capacidad una de las más importantes para el sector de la salud.

La elaboración de las estadísticas en gran parte de las instituciones hospitalarias no está informatizada. Normalmente son hechas manualmente o cuentan con aplicaciones no estandarizadas, que no forman parte de un sistema de gestión hospitalario que les permita a las diferentes áreas, nutrirse de datos generados por otras secciones del hospital. Las herramientas y recursos de que disponen los especialistas para realizarlas, son limitados en cuanto a capacidad de procesamiento, almacenamiento y presentación de todo el sistema estadístico empleado.

La situación antes mencionada, provoca que sea engorroso realizar las estadísticas en un centro, puesto que para ello, se deben consultar muchos datos de diferentes fuentes, con posibles inconsistencias en los mismos. Dificulta el tratamiento de la información y la presentación de los resultados, al no poseer métodos estandarizados para estas tareas. Además, demora la gestión del personal encargado de generar los reportes necesarios para apoyar la toma de decisiones.

Los reportes estadísticos son elementos necesarios en cualquier empresa de gestión, particularmente en un hospital, son de fundamental apoyo para optimizar el funcionamiento del mismo, debido a que le confieren una mayor utilidad a los datos. Su función principal es la de emitir consolidados de información en disímiles formatos, para una mejor comprensión, presentación y análisis. Áreas como: Anatomía patológica, Admisión, Consulta Externa, Hospitalización entre otras, se benefician del uso de estas herramientas para facilitar su trabajo.

Por ejemplo, el módulo Estadísticas genera el reporte de Consolidado de departamentos clínicos y servicios, dicho reporte resume, para un año seleccionado de acuerdo a un departamento o todos los departamentos de la entidad y un servicio o todos los servicios por departamento, el valor que determina esta asociación. La incorporación de información anterior a la puesta en funcionamiento de la aplicación, posibilitaría contar desde el inicio de la puesta en funcionamiento del sistema, los datos de los servicios por departamentos desde la misma fecha de apertura del hospital si han sido ingresados al mismo. Lo que posibilitaría tener el conocimiento de cuáles servicios han sido los más brindados, lo que permitiría poder administrar con mayor calidad los recursos destinados a cada uno.

Como se evidencia, muchos reportes estadísticos se basan en información generada, en ocasiones, con bastante tiempo de antelación a la emisión del mismo. Los datos necesarios para la conformación de informes con el mayor beneficio, pueden ser incluso de antes de la implantación de un sistema o mecanismo para generarlos. Por ende, es imprescindible incorporar al procedimiento los datos históricos en formatos estandarizados, para ser utilizados también en la elaboración de los consolidados que así lo requieran.

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones; y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría la sociedad acercarse más hacia el objetivo de un desarrollo sostenible. (1)

Para trabajar a favor de este proceso en la UCI, específicamente en la Facultad 7, el Centro de Informática Medica (CESIM) desarrolla el sistema alas HIS, que propone una solución informática para la gestión de los procesos hospitalarios. El sistema es una aplicación web de gestión concebida en aras de facilitar el trabajo en las instituciones hospitalarias. En este sistema es de gran importancia el módulo Estadísticas, así como los reportes que del mismo se generan, por la necesidad de poder hacer predicciones basadas en datos históricos.

El sistema empieza a operar con datos registrados a partir del momento de su instalación, sin embargo, muchos de los reportes que se necesitan generar se basan en información de momentos anteriores. El sistema no cuenta con los mecanismos efectivos para garantizar que se pueda ingresar al mismo, datos históricos que tributen a la generación de los reportes. Esto permitiría generar reportes con mayor veracidad que resuman aspectos de vital importancia para pacientes y trabajadores del sector de la salud.

A partir del análisis de la situación anterior, se define el siguiente ***problema a resolver***: ¿Cómo gestionar en el módulo Estadísticas del sistema de información hospitalario alas HIS los datos generados históricamente en las instituciones hospitalarias?

El ***Objeto de estudio*** lo constituye el proceso de gestión de datos históricos en el área de estadísticas de las instituciones hospitalarias, enmarcado en el ***Campo de acción***, gestión de datos históricos en el módulo Estadísticas del Sistema de Gestión Hospitalaria alas HIS.

Basado en esa idea se define el ***Objetivo general:*** Desarrollar las funcionalidades en el módulo Estadísticas del Sistema de Gestión Hospitalaria alas HIS que permitan gestionar datos históricos, y modificar los reportes existentes, para que incorporen al proceso de generación de los mismos los datos históricos.

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

- Valorar las tendencias actuales en el mundo de los sistemas de incorporación de datos particularmente los asociados a sistemas estadísticos.
- Aplicar la arquitectura definida por el Departamento de Sistema de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
- Desarrollar las funcionalidades necesarias.
- Obtener los artefactos correspondientes a los flujos de trabajo “Implementación” y “Pruebas”.
- Obtener el acta de liberación otorgada por Calidad.

El presente documento se encuentra estructurado en cuatro capítulos, el primero de ellos, **“FUNDAMENTACIÓN TEÓRICA”,** donde se realiza un estudio del estado del arte en cuanto a sistemas de información hospitalarios, tecnologías, metodologías y herramientas existentes en el mundo.

El siguiente capítulo, **“CARACTERÍSTICAS DEL SISTEMA”,** contiene el marco conceptual asociado a la información que será manipulada por el sistema, arribando a un acuerdo sobre las funcionalidades, requerimientos deseados y el objeto de automatización.

El tercer capítulo **“DISEÑO DEL SISTEMA”** se centra en la modelación detallada y la construcción de la estructura de la aplicación.

En el cuarto y último, **“IMPLEMENTACIÓN”,** se implementan las clases y subsistemas en términos de componentes. Se presenta la propuesta de solución para lograr una gestión más eficiente de los requerimientos de seguridad de los proyectos que pertenecen al Departamento de Gestión Hospitalaria del Centro de Informática Médica (CESIM).

Capitulo 1: Fundamentación teórica.

En este capítulo se realiza un estudio de los sistemas vinculados al campo de acción, valorando sus capacidades y estudiando una posible propuesta de solución a la situación problémica. Así como que se enuncian las herramientas, metodologías y el lenguaje de programación a utilizar para desarrollar la solución.

1.1 Sistemas automatizados existentes

Existen en el mundo empresas que se especializan en el desarrollo de sistemas informáticos vinculados a la incorporación de datos históricos. Con la experiencia adquirida se han desarrollado soluciones de software que utilizan distintos métodos para agilizar el proceso y minimizar la pérdida de información sensible. El proceso es básicamente la agregación de datos a los sistemas, provenientes de diferentes fuentes como materiales tangibles, por ejemplo documentos clínicos en papel con disímiles formatos, así como materiales electrónicos, digitalizados y provenientes de diferentes bases de datos.

El software **Kofax Capture**, antiguamente Ascent Capture, es una herramienta que posibilita agilizar los procesos empresariales recopilando documentos y formularios, y transformándolos en datos fácilmente recuperables. Permite convertir documentos físicos a digitales y permite importar documentos en formato electrónico. Para ello extrae la información importante, como texto impreso, códigos de barras, texto manuscrito y casillas marcadas. Permite en el módulo de administración configurar, por ejemplo, los tipos de documentos que se procesarán y cómo, y en qué orden se procesarán. Además se pueden definir perfiles de separación de documentos, identificación de formularios, limpieza de imágenes y perfiles de reconocimientos.

Utiliza el Reconocimiento de Caracteres Ópticos (OCR), Reconocimientos de Caracteres Inteligentes (ICR) y el Reconocimiento Óptico de marcas (OMR).Es una aplicación modular que puede utilizarse directamente para satisfacer las necesidades de captura de información de un departamento específico o de toda una empresa.

Kofax Capture puede exportar a cualquier base de datos compatible con ODBC o archivos ASCII delimitados. Kofax Capture en su edición para empresas ofrece opciones para sistemas de gestión empresarial de bases de datos, como IBM DB2, Microsoft SQL Server Enterprise, y Oracle Database. Tiene la limitante de ser un software privativo que solo tiene soporte para las versiones de Windows y sus funcionalidades están basadas en la plataforma .Net y es una aplicación de escritorio. (2)

ABBYY FlexiCapture 9.0, es una solución para captura de datos y procesamiento de documentos. En ABBYY FlexiCapture el proceso de captura de datos está dividido en varias etapas. La distinción básica está en los dos modos de usuario: modo administrador que configura todo el proceso de captura, prepara las plantillas documentales y prueba la preparación de resultados; y el modo operador que realiza operaciones de captura y procesamiento solamente.

Provee un conjunto de reglas de validación predefinidas para garantizar la corrección de datos. Mediante la aplicación de estas reglas es posible comprobar los datos automáticamente contra una base de datos, seleccionarlos para satisfacer al formato, comprobar si la suma total está correcta (por ejemplo, para las tareas de captura de facturas), llevando los datos a un estándar (normalizado). También es posible añadir cualquier regla personalizada usando el idioma de scripting. Está diseñada para control manual de la exactitud de los datos reconocidos y estructura de documentos así como para la entrada manual de alguna información, por ejemplo, texto irreconocible.

Los datos extraídos e imágenes de documento pueden ser exportados a destinos diferentes:

- Archivos.
- Bases de datos externas (a través de ODBC).
- Microsoft SharePoint 2003/2007.
- Cualquier aplicación de negocios, sistemas ERP o ECM/DMS.
- Los documentos pueden ser guardados como ficheros PDF/PDF-A con posibilidad de búsqueda para poder archivarlos después.

Tiene un API de servicios Web que provee acceso SOAP al servidor de procesamiento de FlexiCapture desde una aplicación externa. Al utilizar el API, FlexiCapture puede ser integrado dentro de varias aplicaciones de negocio y flujos de trabajo como un clasificador automático de documentos y servicio de captura de datos. El API de servicios Web permite cargar imágenes desde la carpeta a un proyecto pre-configurado y recibir los resultados de reconocimiento en formato XML avanzado. (3)

Basado en estas tecnologías, la empresa provee el ABBYY FlexiCapture Engine 9.0 que es un kit de desarrollo de software para extracción de datos, clasificación de documentos, indexado y conversión a PDF, que permite a los desarrolladores crear aplicaciones que apoyen la gestión empresarial básicamente en el proceso de captura de datos. (4)

EpiData basado en la versión 6 de EpiInfo, cuyas funcionalidades se basan en la idea de que el usuario escribe líneas de texto simple que el programa traducirá en un formulario de entrada de datos. Una vez que el formulario está listo, es sencillo definir qué datos pueden almacenarse en cada campo.Durante la entrada de datos se pueden realizar cálculos o controlar los valores que pueden introducirse. Se puede seleccionar un valor de una lista y guardar su código numérico (1 = No 2= Si), la lista de textos se exportará con etiquetas para su uso en paquetes estadísticos.

Las fechas son manipuladas fácilmente, por ejemplo, 2301 será reconvertida como 23/01/2001 si se graba en el año 2001 en un campo "dd/mm/yyyy". Es útil para manejar conjuntos de datos simples, como cuando se tienen datos de una fuente única por ejemplo, un cuestionario o un formulario de un laboratorio, o conjuntos más complejos con varios formularios o cuestionarios relacionados. Un punto fuerte de EpiData es la posibilidad de especificar reglas y realizar cálculos durante la entrada de datos.

Dentro de las funcionalidades del sistema se encuentra, realizar cálculos durante el proceso de entrada de datos, por ejemplo, calcular la edad, el día de una visita, a partir de la fecha de nacimiento y la fecha de la visita. Después de crear el archivo de datos, se puede documentar la base de datos. Es posible visualizar los registros de información filtrando las variables del reporte a mostrar. EpiData incluye otros aspectos como la posibilidad de comparar dos archivos de datos indicando las diferencias entre ellos a nivel de registro. Es una aplicación libre con interfaces puramente de escritorio. (5)

Como conclusión se evidencia que en un software de entrada automatizada de datos, para minimizar la pérdida de información es necesario utilizar dispositivos de gran calidad para el escaneo de documentos, lo que impone una gran inversión en este sentido. Así como la necesaria capacitación de los usuarios al utilizar tecnología de avanzada. Además, los resultados de la captura de datos dependen de la calidad de la imagen de los documentos digitalizados. Los documentos que tienen colores o patrones de fondo, que han sido marcados con rotulador o que se han doblado pueden inducir a errores en el sistema.

Basado en el estudio realizado se propone una solución orientada a la entrada manual de datos históricos al sistema alas HIS debido a que esta opción resulta mucho más económica y de mayor compatibilidad con las características del sistema y para un entorno en el que no se puede suponer que cuenta con documentos clínicos digitalizados y en buen estado. Así como la no existencia de bases de datos con la información necesaria para el funcionamiento del sistema.

1.2 Herramientas y tecnologías de desarrollo

Eclipse

Eclipse es un entorno de desarrollo integrado multiplataforma de código abierto. Actualmente es desarrollado por la Fundación Eclipse organización independiente sin ánimo de lucro, pero fue creado originalmente por IBM. 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. Es totalmente extensible con módulos que aumentan su funcionalidad. Solo requiere tener instalado Java Runtime Environment 1.4.2 o superior. Permite la integración con la herramienta Visual Paradigm herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software lo que proporciona un mejor entendimiento entre analistas y desarrolladores. (6)

PostgresSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD, que se destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL. Este cuenta con diversas versiones para sistemas operativos tales como: Linux, Windows, Unix, Mac OS X, Solaris, BSD, Tru64 y otros.

Tiene como algunas características fundamentales:

- Soporte nativo para los lenguajes más populares: PHP, C, C++, Perl, Python, etc.
- Soporte de todas las características de una base de datos profesional (triggers, procedimientos almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.
- Soporte de tipos de datos de SQL92 y SQL99.Soporte de protocolo de comunicación encriptado por SSL. Extensiones para alta disponibilidad, datos espaciales, minería de datos, etc.

PostgresSQL usa una estrategia de almacenamiento de filas llamada MVCC (Acceso concurrente multiversión, por sus siglas en inglés) para conseguir una mejor respuesta en ambientes de grandes volúmenes. (7)

Jboss Seam

Jboss Seam es un moderno framework desarrollado por Jboss, el mismo unifica e integra los distintos estándares de la plataforma Java EE 5.0. Es la combinación de dos frameworks el Enterprise Java Beans y el JavaServerFaces. Seam provee una mayor granularidad de contextos de estado a diferencia de los frameworks tradicionales donde todo el estado es administrado en la sesión http. Una

característica importante es que se pueden hacer validaciones en los POJOs (Plain Object Java). Además desde de la sesión Beans se maneja la lógica de la aplicación y el negocio.

Seam proporciona soporte completo para dos de las arquitecturas de persistencia más populares: hibernate 3 y el Java Persistence API introducido con EJB 3.0. JBoss Seam hace uso extensivo de AOP para proporcionar funcionalidades de caché, seguridad, inyección de dependencias, interceptores, pageflo. El diseño de JBoss Seam ha sido concebido con Ajax en mente. Es capaz de manejar peticiones simultáneas de distintos usuarios, preservando las condiciones de aislamiento e integridad de los datos. (8)

JBoss Server

JBoss es un Servidor de Aplicaciones Java EE de Software Libre implementado en Java. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc., adquirida por Red Hat en Abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El servidor de JBoss está dividido en módulos y está implementado usando plug-ins basados en componente. Esto es posible gracias a que su microkernel está basado en JMX, que es un API de Java, Java Management Extension.

Características

• Producto con licencia de código abierto: esto significa que JBoss puede ser descargado, instalado, utilizado y distribuido sin restricciones por la licencia.

• Cumple con los estándares J2EE: JBoss está implementado en Java puro. Como una de las principales características al estar basado en Java, JBoss AS puede ser utilizado en cualquier sistema operativo que lo soporte. Este cumplimiento de los estándares lo convierte en un framework capaz de soportar aplicaciones empresariales.

• Confiable a nivel de empresa: Este proyecto está apoyado por una red mundial de colaboradores, lo que inspira confianza a las empresas que lo utilizan.

• Arquitectura orientada a servicios: Esto le permite a las empresas que implementen soluciones de software utilizando JBoss AS, configurar los servicios que provee el servidor y que mejor se adapten a sus necesidades empresariales.

• Servicios de middleware: Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas para cualquier objeto de Java. (9)

Hibernate

Es una herramienta de Mapeo Objeto-Relacional (ORM) para la plataforma Java. Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (*Hibernate Query Language*). Puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA, que es parte de esta plataforma.

Es un popular servicio de persistencia que proporciona una simple, pero poderosa, alternativa a los "beans" estándares. Fue diseñado para trabajar en un clúster de servidor de aplicaciones y proporcionar una arquitectura altamente escalable. Facilita la creación de clases de persistencia utilizando el lenguaje Java, incluyendo la asociación, herencia, polimorfismo y composición. (10)

IReport

La herramienta IReport es un constructor / diseñador de informes visuales, poderoso, intuitivo y fácil de usar para JasperReports escrito en Java y es de código abierto. Maneja el 98 % de las herramientas de JasperReports. Soporta Java Beans como orígenes de datos. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. IReport está además integrado con JFreeChart, una de las bibliotecas gráficas OpenSource más difundida para Java. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, TableModels, JavaBeans, XML, entre otras. (11)

JasperReport

El Jasper Report es una librería para la generación de informes. Está escrita en java y es libre, puede ser usada en gran variedad de aplicaciones de Java, incluyendo J2EE o aplicaciones web, para generar contenido dinámico. El funcionamiento consiste en escribir un xml donde se recogen las particularidades del informe. Este xml lo tratan las clases del Jasper para obtener una salida. Esta salida puede ser un PDF, XML, HTML, CSV, XLS, RTF, TXT. Otra ventaja de utilizar Jasper Report es que se integra perfectamente con el JFreeChart que es una librería libre para la generación de todo tipo de gráficas. JasperReports se usa comúnmente con IReport, un front-end gráfico de código abierto para la edición de informes. (12)

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. 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. (13)

1.3 **Lenguaje de programación**

Un lenguaje de programación no es más que un idioma artificial diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para controlar el comportamiento físico y lógico de la máquina. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Dentro de los lenguajes de programación más usados y exitosos en el mundo del desarrollo de aplicaciones informáticas se encuentra Java. Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principio de los años 90.

Es elogiado por tener un modelo más simple que lenguajes más antiguos como C y C++, y por eliminar herramientas de bajo nivel que estos sostienen como el uso de punteros que puede inducir a cometer errores con mayor facilidad. Es independiente de la plataforma, es decir, programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. En Java el problema de las fugas de memoria se evita en gran medida gracias a la recolección de basura (o *automatic garbage collector*).

El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. La sintaxis de Java se deriva en gran medida de C++. Pero a diferencia de éste, que combina la sintaxis para programación genérica, estructurada y orientada a objetos, Java fue construido desde el principio para ser completamente orientado a objetos.

Este lenguaje se caracteriza por ser:

Compilado: Genera ficheros de clases compiladas, pero estas son en realidad interpretadas por la máquina virtual de java. Dicha máquina virtual es quien mantiene el control sobre las clases que se ejecutan.

Multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual de java.

Seguro: La máquina virtual, al ejecutar el código, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

Distribuido: Fue construido con extensas capacidades de interconexión TCP/IP. Tiene librerías de rutinas para acceder e interactuar con protocolos como http y ftp, lo que permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales. (14)

1.4 **Metodologías de desarrollo**

Metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término "método" para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño.

Se puede definir a la metodología de desarrollo de software como un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. (15)

RUP

El *Proceso Unificado de Rational* (RUP), es un proceso de ingeniería de software cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecido. Cubre el ciclo de vida y desarrollo de software.

El Proceso Unificado está equilibrado por ser el producto final de tres décadas de desarrollo y uso práctico. Su desarrollo como producto sigue un camino desde el *Proceso Objectory* (primera publicación en 1987) pasando por el *Proceso Objectory de Rational* (publicado en 1997) hasta el *Proceso Unificado de Rational* (publicado en 1998). En este camino de desarrollo ha tenido la influencia mayoritaria de dos grandes métodos: el *Método de Ericsson* y el *Método de Rational*.

RUP toma en cuenta las mejores prácticas en el modelo de desarrollo de software, en particular las siguientes:

1. Desarrollo de software en forma iterativa.

2. Manejo de requerimientos.

3. Utiliza arquitectura basada en componentes.

4. Modela el software visualmente (modela con UML).

5. Verifica la calidad del software.

6. Controla los cambios.

El Proceso Unificado es un proceso de desarrollo de software cuyo ciclo de vida se caracteriza por:

- Dirigido por casos de Uso
- Centrado en arquitectura
- Iterativo e incremental

El Proceso Unificado de Racional consta de cuatro fases o etapas:

Comienzo 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 uno o varios release del producto que ha pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.

Transición: El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Flujos de trabajo de desarrollo:

Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

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.

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.

Prueba (Testeo): Busca los defectos a los largo del ciclo de vida.

Flujos de trabajo de soporte:

Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

Administración de configuración y cambios: 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: 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. (16)

UML

Lenguaje Unificado de Modelado (LUM O UML, por sus siglas en inglés) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, está respaldado por la OMG (Object Management Group).

UML es un lenguaje gráfico para construir, documentar, visualizar y especificar un sistema de software, en otras palabras UML se utiliza para definir un sistema de software. Ofrece un estándar para describir un “plano” del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un “lenguaje” para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. (17)

En este capitulo se realizó un estudio de los sistemas vinculados al campo de acción, así como un análisis de las herramientas, tecnologías, metodologías y el lenguaje de programación a utilizar para desarrollar la solución.

Capitulo 2: Descripción de la arquitectura.

En el presente capitulo se especifican los requerimientos no funcionales y se describe la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria. Se describe brevemente la definición de la arquitectura del sistema a implementar. También se analizan posibles implementaciones de componentes o módulos que puedan ser reutilizados. Además se especifican las estrategias de codificación y se explican los mecanismos de seguridad a utilizar en la creación del software.

2.1 Requisitos no funcionales.

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema que definen las limitantes del mismo; así como las del proceso de desarrollo. Son aquellos requerimientos que surgen de la necesidad del usuario y no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. Es por ello que una falla en un requerimiento no funcional del sistema lo inutiliza. (18)

Usabilidad

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

Usuarios normales: 30 días

Usuarios avanzados: 20 días

Fiabilidad

En los servidores de los hospitales se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de base de datos como de aplicación. Además se garantizarán políticas de respaldo de todos los datos, evitando que en caso de desastres ajenos ocurran pérdidas de información en el sistema. Se almacenarán localmente en los hospitales los estudios que por su tamaño no se puedan replicar hacia el Centro de Datos, quedando en el Centro de Datos solo la referencia a dicho estudio, de esta forma se accederá a dichos estudios mediante una transmisión directa entre los hospitales, sin que el Centro de Datos Nacional medie en la transmisión de la información.

Las informaciones médicas relacionadas con los pacientes, que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas. El personal autorizado será alertado de posibles irregularidades que den indicios sobre introducción de información falseada por los mecanismos de control y verificación de fraudes. Se mantendrá seguridad

y control a nivel de usuario, restringiendo el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el administrador del sistema o por los propios usuarios.

El sistema proporcionará un registro de actividades (log) de cada usuario en el sistema, al implementar un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios. El producto soportará el uso de firmas digitales para la transferencia de información cuya certificación sea imprescindible para validar el uso de la misma. Además, implementará un control de cambios a determinados campos de información (seleccionados por su importancia), de forma tal que sea posible determinar cuáles han sido las actualizaciones que se le han realizado.

Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos, independientemente de que para el sistema, este elemento ya no exista. Se recuperará la información de la base de datos a partir de los respaldos o salvas realizadas.

Eficiencia

Sin afectar los sistemas, el Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento, garantizando así expansiones motivadas por futuros requerimientos. Además de, minimizar el volumen de datos en las peticiones y optimizar el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos. Siempre que sea posible se deberá usar el patrón Singleton, optimizar el trabajo con cadenas, destruir referencias que ya no estén siendo usadas, entre otras buenas prácticas que ayudan a mejorar el rendimiento.

Seguridad

La seguridad de un sistema engloba además de la seguridad con que cuenta el software, la seguridad del ambiente en el que se usará. Por lo que se tienen que contemplar las regulaciones legales, así como los controles administrativos y la seguridad física que afectan el uso del sistema.

La seguridad de la información tiene el objetivo de garantizar:

- Confidencialidad: La información o los activos informáticos son accedidos solo por las personas autorizadas para hacerlo.

- Integridad: Los activos o la información solo pueden ser modificados por las personas autorizadas y de la forma autorizada.
- Disponibilidad: Los activos informáticos son accedidos por las personas autorizadas en el momento requerido.

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

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. Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos. El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

Rendimiento

Utilizando buenas prácticas de programación se mejorará para la máquina virtual el rendimiento de operaciones costosas como la creación de objetos. El sistema minimizará el volumen de datos en las peticiones optimizando recursos críticos como la memoria.

Soporte

Una vez terminado el desarrollo del software se tomarán acciones con motivo de asistir a los clientes, así como para el mejoramiento progresivo del sistema.

Será posible la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos. Además de se aprobará administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación. Se realizarán 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:

Para el trabajo con el Sistema de Información Hospitalaria alas HIS, se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz con sistema operativo Linux. Se recomienda Internet Explorer 7, Firefox 2 o versiones superiores.

Servidores:

Por la necesidad de servidores de alta capacidad de procesamiento, se escogieron estaciones con las siguientes características.

Servidores de Base de datos: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux.

Servidores de Aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux.

Servidores de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 2 GB de memoria y 2x72GB de disco y sistema operativo Linux.

Software

El sistema debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss AS y PostgreSQL). El sistema deberá disponer de un navegador web, estos pueden ser IE 7, Opera 9, Google chrome 1 y Firefox 3.0 o versiones superiores de estos.

Restricciones de diseño

Toda la lógica de la presentación incluyendo todas las vistas será englobada en la capa de presentación. El flujo web será manejado basándose en definiciones de procesos del negocio y de forma declarativa. La capa del negocio mantendrá el estado de las conversaciones y procesos del negocio, que puedan ser ejecutados por cada usuario de forma concurrente. 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:

Para permitir la interpretación correcta de la información que suministra el sistema el mismo contará con ventanas que muestren los datos de manera clara y bien estructurada. La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario.

Interfaces de comunicación:

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

Multiplataforma

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

2.2 **Descripción de la arquitectura**

La definición oficial planteada en la IEEE 1471-2000 especifica que 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. (19)

La misma ocurre desde el mismo inicio del ciclo de vida del proyecto y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales. Es una vista estructural de alto nivel.

En la creación del HIS se define como parte de la línea base de la arquitectura el patrón Modelo Vista Controlador (MVC),el mismo separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio. Estas características proveen al patrón de una gran fortaleza debido a que al ser utilizado se pueden cambiar o sustituir elementos de un componente causando el mínimo de alteraciones posibles en otros elementos que los utilicen.

2.3 Descripción del patrón MVC

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

En el sistema la capa de acceso a datos usa la implementación de JPA de Hibernate 3.3, utilizando las características que posee, 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 permite establecer validaciones end-to-end mediante el uso de los Hibernate Validators.

Vista: Esta es la representación de la información en un formato adecuado para interactuar con el usuario.

En el sistema, la capa de presentación está formada por páginas XHTML, las cuales están compuestas por formularios y controles JSF y RichFaces. Esta se combina fácilmente con el *framework* de integración escogido, Seam, y permite generar vistas no necesariamente basadas en HTML (PDF, etc.). Adiciona, además, controles out-of-the-box AJAX-ready y el framework de extensión AJAX para los controles JSF básicos, Ajax4Jsf. 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. 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.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

La capa del 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.

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

En el proceso de registrar los históricos de reportes estadísticos se ingresan al sistema los datos históricos necesarios para la generación de los consolidados, de acuerdo con las variables de salida

establecidas en la plantilla del reporte. Para la utilización en la generación de los reportes existentes de información histórica, la implementación de estos tiene que ser modificada.

Las clases controladoras de: Generar reportes estadísticos de Indicadores hospitalarios, Generar reportes estadísticos de Movimiento de hospitalizados según departamentos clínicos y servicios, Generar reportes estadísticos de Consulta Externa, Generar reportes estadísticos de Servicios de apoyo, Generar reportes estadísticos de Emergencia, Generar reportes estadísticos de Morbilidad hospitalaria, Generar reportes estadísticos de Mortalidad hospitalaria, Generar reportes estadísticos de Natalidad hospitalaria, Generar reportes estadísticos de comportamiento de la morbilidad o mortalidad por patología, Generar reportes estadísticos de comportamiento de la natalidad, Generar reporte de causas de cancelación de intervenciones quirúrgicas, Generar reportes estadísticos de Consolidado de departamentos clínicos y servicios; tienen que incluir, en las consultas que realizan a la base de datos, la obtención de información de las tablas correspondientes a los históricos asociados a cada reporte.

Al consultar estas entidades los datos serán devueltos en el mismo formato que necesita el reporte para mostrarlos.

Además existen algunos componentes que se definen de manera general para ser utilizados por todos en el sistema alas HIS, estos proveen un conjunto de facilidades y simplifican el trabajo de los desarrolladores. Dentro de estos están: la clase Bitacora para el control de las trazas de todas las acciones que se realizan con la aplicación; la clase UserTools para saber qué usuario está trabajando con la aplicación en tiempo real; y la interfaz IActiveModule para conocer qué modulo está activo y en qué entidad. También se utilizan las clases EntityManager con las funcionalidades para el acceso a datos, JasperReport para la manipulación de los datos al generar los reportes y ListadoControler para el paginado en memoria.

2.5 **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 pueda 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.

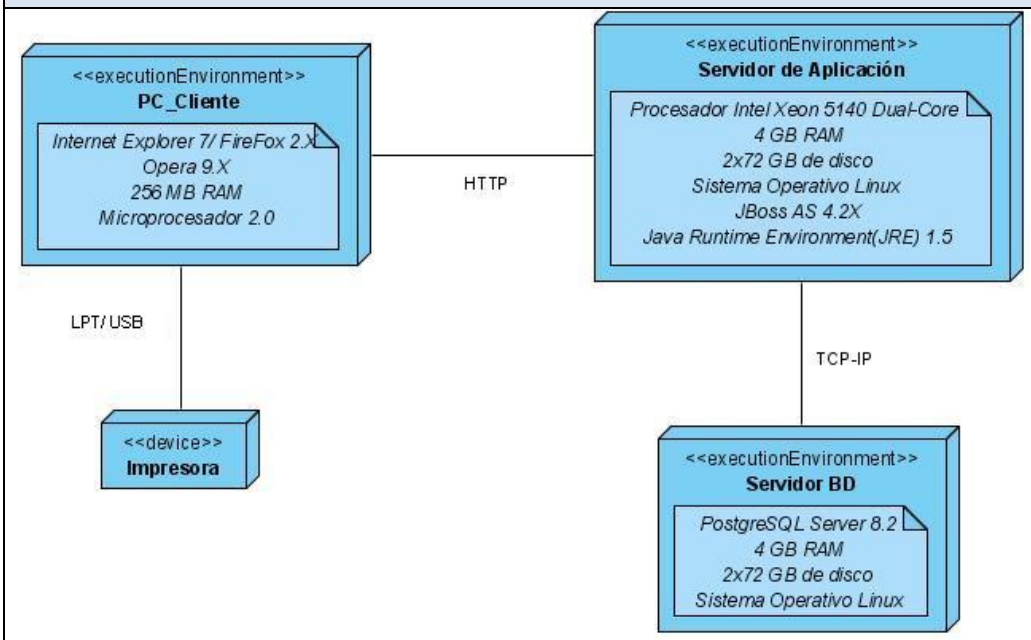
La integridad de los datos es otro de los aspectos a tener en cuenta. Esta se logra mediante el cifrado de la información proveniente de la comunicación entre los componentes internos del sistema y otros

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.

2.6 **Vista de despliegue**

La arquitectura en tiempo de ejecución del Sistema de Información Hospitalaria alas HIS se modeló utilizando tres elementos de hardware: una computadora cliente y dos servidores (servidor de aplicación y servidor de base de datos), además se dispuso de la conexión con un dispositivo, en este caso la impresora. La comunicación entre los nodos se realiza por los protocolos HTTP para asociar la computadora cliente y el servidor de aplicaciones y por TCP/IP para establecer la conexión entre los dos servidores. La computadora cliente se conecta a la impresora a través de los puertos USB o LPT.

Diagrama de despliegue



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

Variables y constantes		
Apariencia de	Todas sus letras en	Se deben declarar las constantes con todas sus letras en

constantes	mayúscula	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 1.1. Estándares de codificación - Variables y constantes.

Indentación	
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 deben 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 1.2. 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 1.3. 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 comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación Camello.
Declaración de parámetro en funciones	Agrupados por tipos Poner los string 1 numéricos 2, además,	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen,

	agrupar según valores por defecto.	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 1.4. 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 utilizará 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 será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard, todo en minúscula.
Tablas que	Las 2 primeras letras	El nombre a emplear para estas tablas de relación

representen nomencladores.	representan el tipo. Todas las letras en minúscula.	debe comenzar con el prefijo tn seguido de underscoard. El nombre 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 utilizará 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 1.5. Estándares de codificación – Base de datos, tablas, esquemas y campos.

En el presente capitulo se especificaron los requerimientos no funcionales necesarios. Se describió la definición de la arquitectura del sistema a implementar. Se analizaron los componentes o módulos que puedan ser reutilizados. Además se especificaron las estrategias de codificación y se explican los mecanismos de seguridad a utilizar en la creación del software.

Capitulo 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.

El diseño de sistemas se define como el proceso de aplicar 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. Se basa en el refinamiento del Análisis y describe gráficamente cómo implementar el sistema. Contiene información referente a clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, navegabilidad y dependencias.

Para el Diseño se siguen patrones basados en prácticas estandarizadas para el desarrollo de software orientado a objetos. Los patrones de diseño constituyen soluciones aplicables a distintos problemas característicos de diseño, que pueden encontrarse en diferentes contextos. Su uso permite mejorar la documentación y la reutilización del conocimiento.

Dentro de los patrones más utilizados se encuentran: bajo acoplamiento, alta cohesión, creador y el controlador. Estos se encuentran dentro de un grupo de patrones que se utilizan con el objetivo de asignar responsabilidades a las diferentes clases del diseño, llamados patrones GRASP.

Paralelamente a la definición de la arquitectura del sistema se modela el sistema para que soporte todos los requisitos, incluyendo los no funcionales y sus restricciones. La formación de los diagramas de clases del diseño es la representación de los casos de uso con las clases del diseño y sus objetos.

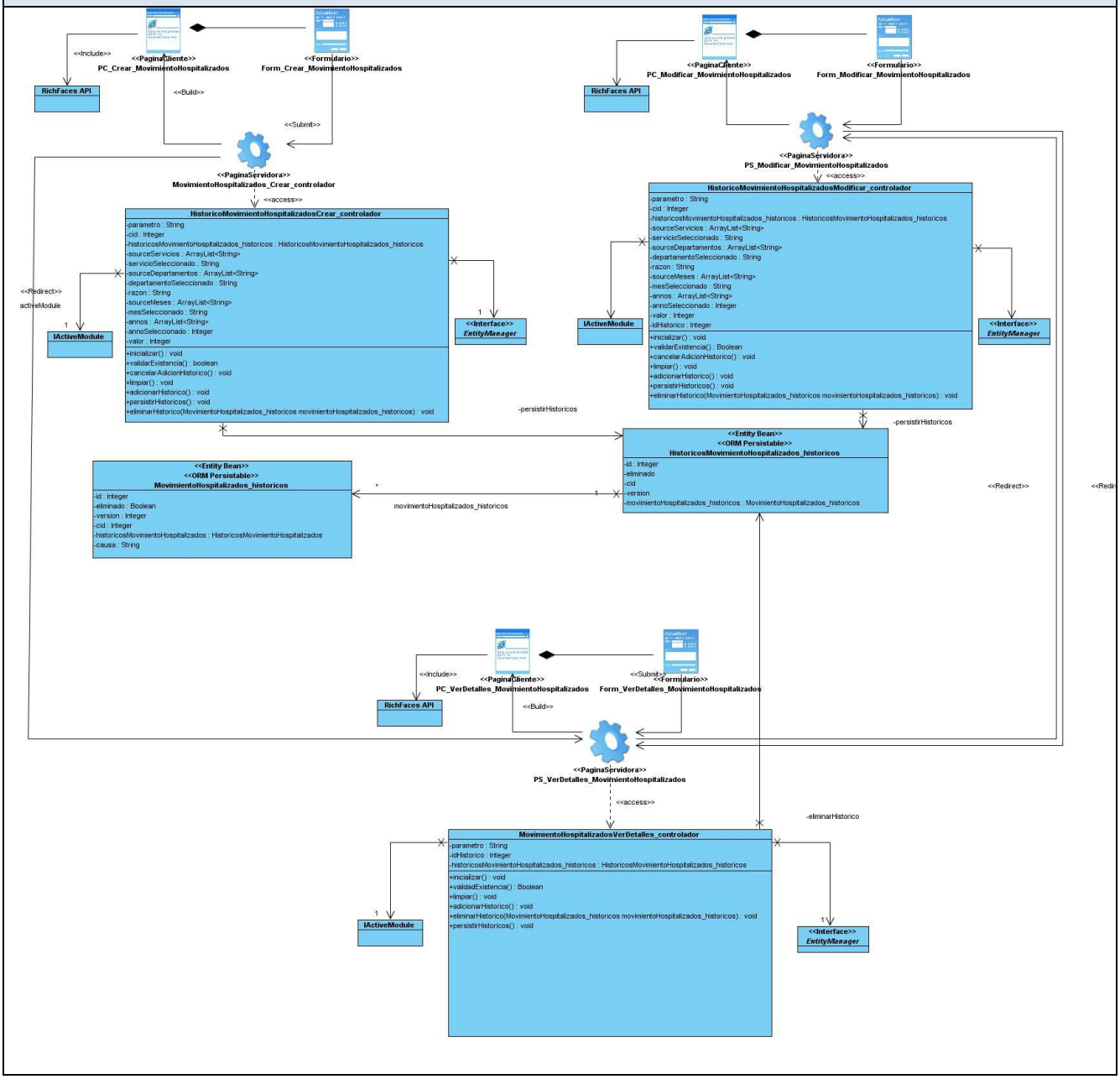
Los **diagramas de clases de diseño** se utilizan para modelar la vista de diseño estática de un sistema. Se componen de un grupo de clases e interfaces que reflejan importantes entidades del sistema que está siendo modelado, y las relaciones entre estas clases e interfaces

Los **diagramas de interacción**, muestran gráficamente como los objetos se comunican entre ellos a fin de cumplir con los requerimientos. Permiten modelar aspectos dinámicos del sistema

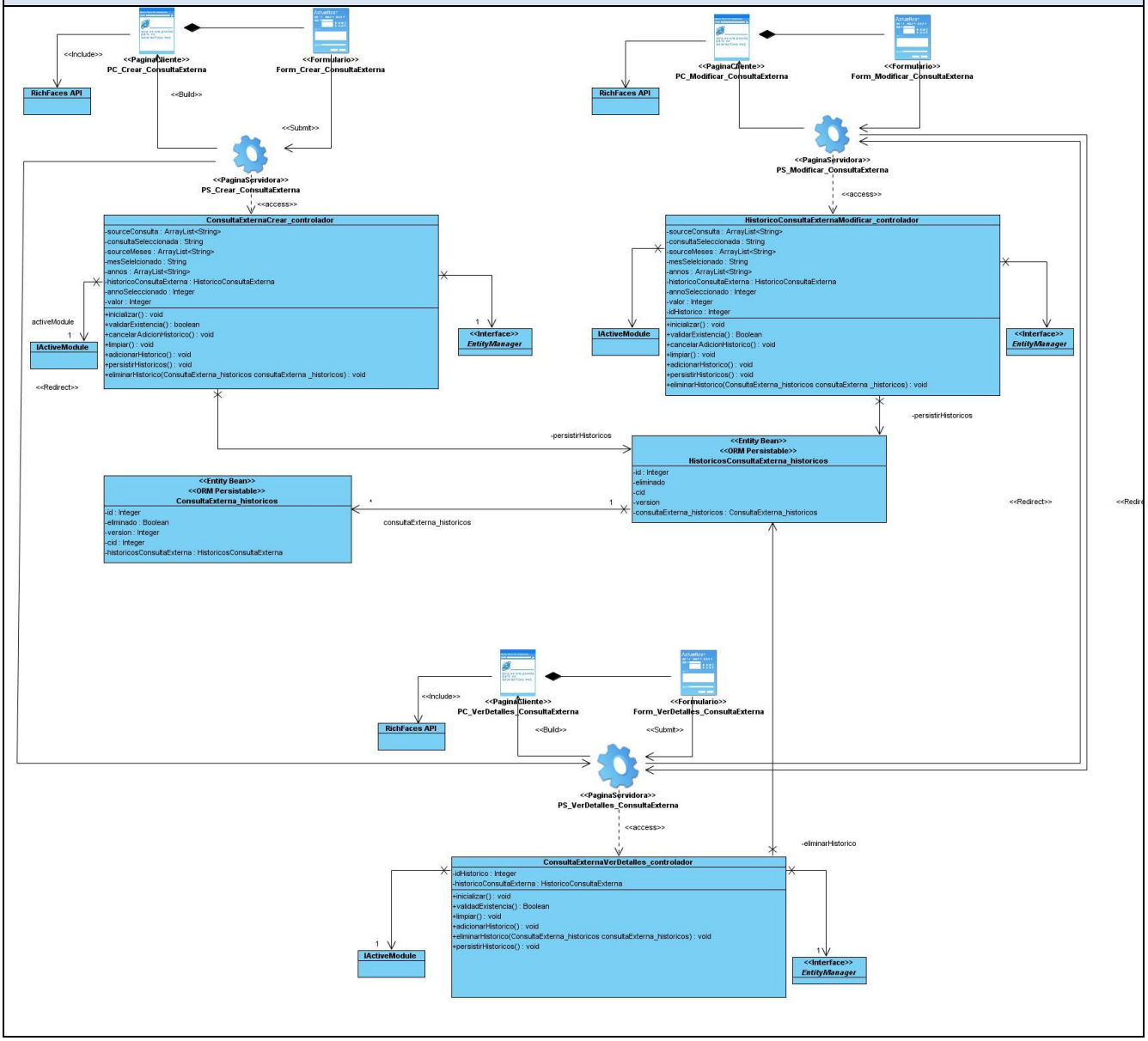
A continuación se muestran los diagramas de clases de diseños y de secuencia de la solución.

3.2 **Diagrama de clases del diseño.**

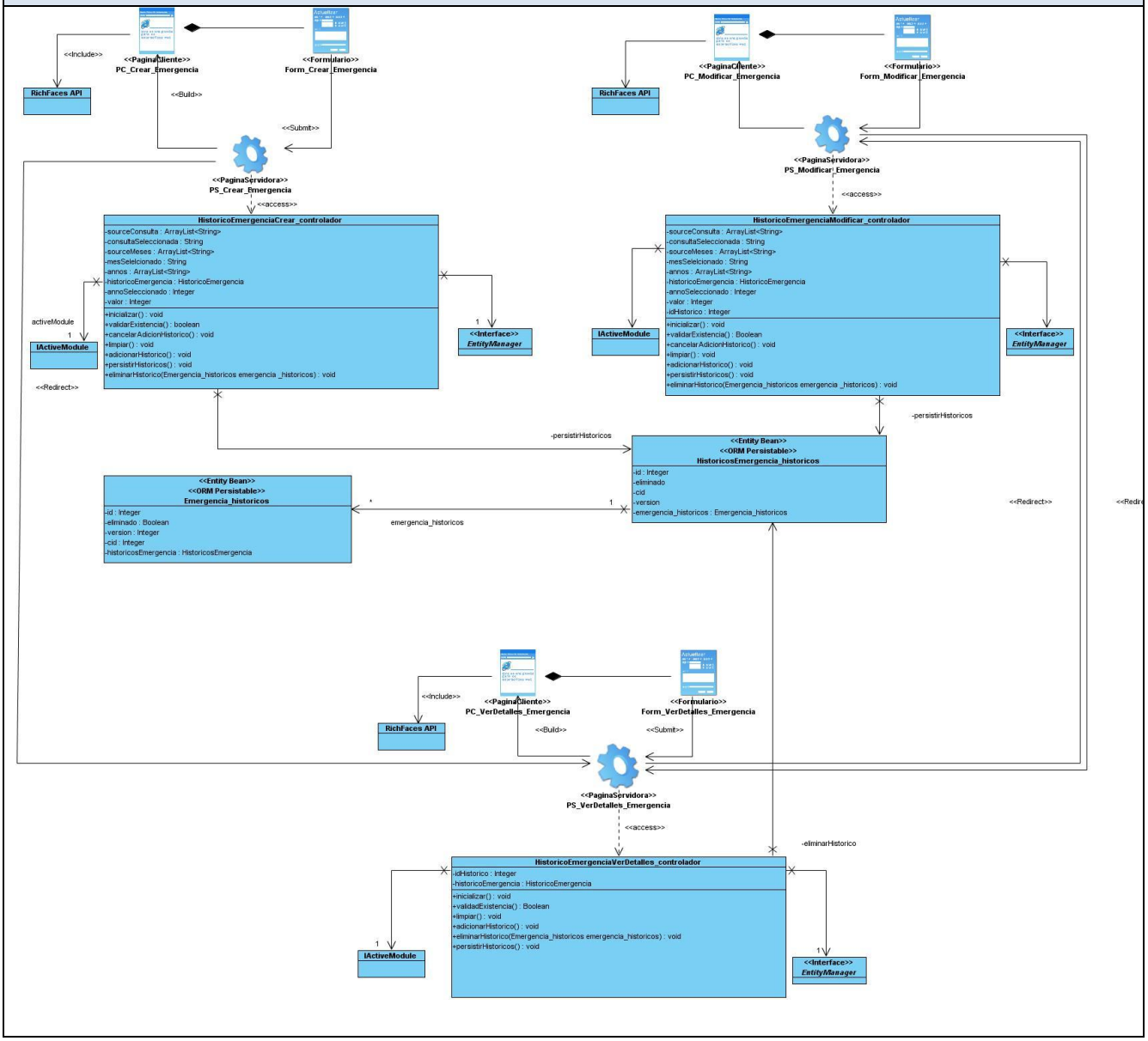
Registrar históricos de Movimientos de hospitalizados



Registrar históricos de Consulta Externa

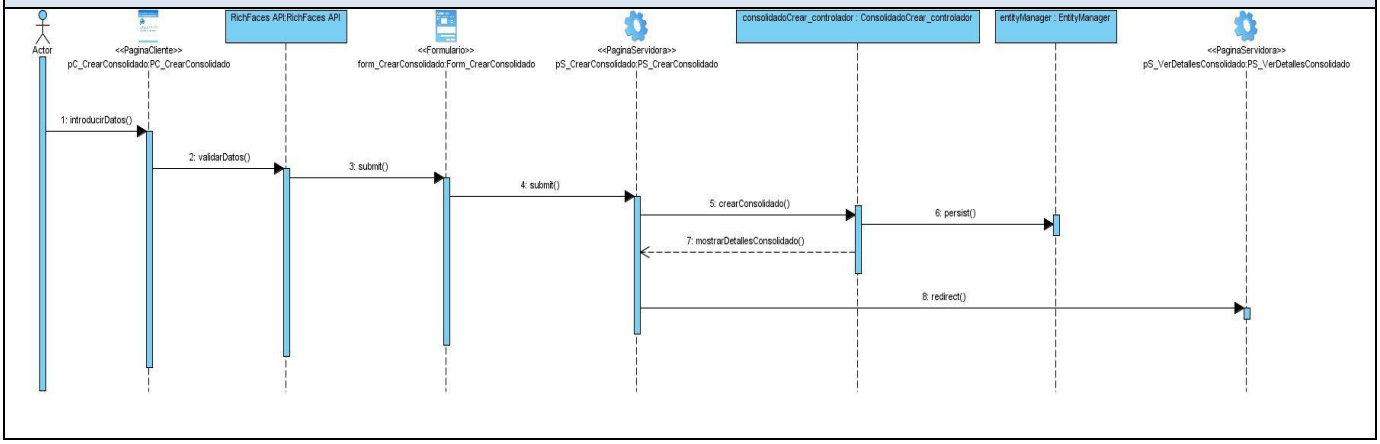


Registrar históricos de Emergencia

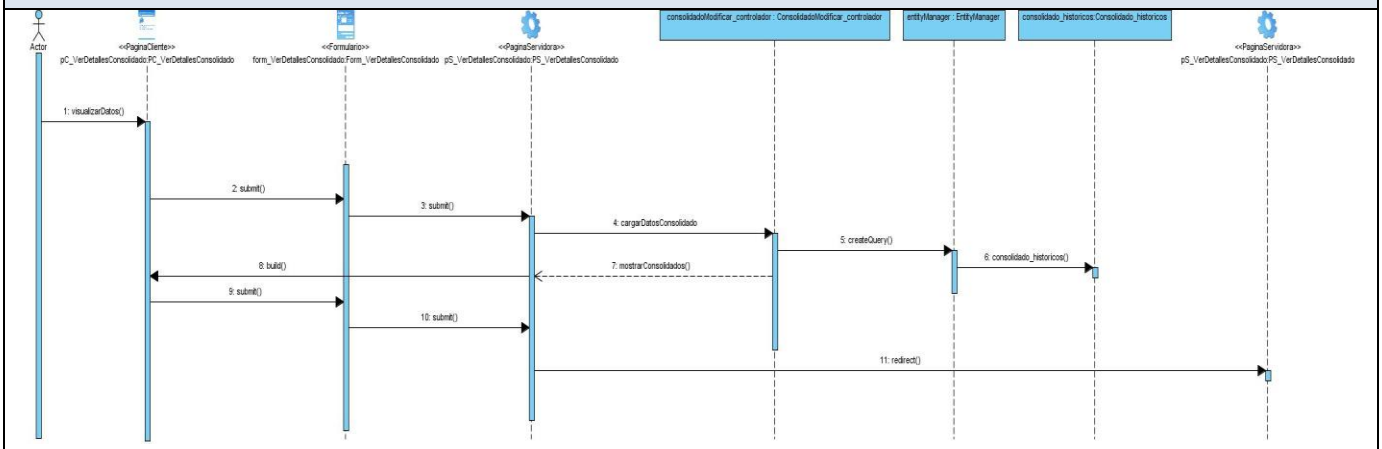


3.3 **Diagramas de interacción.**

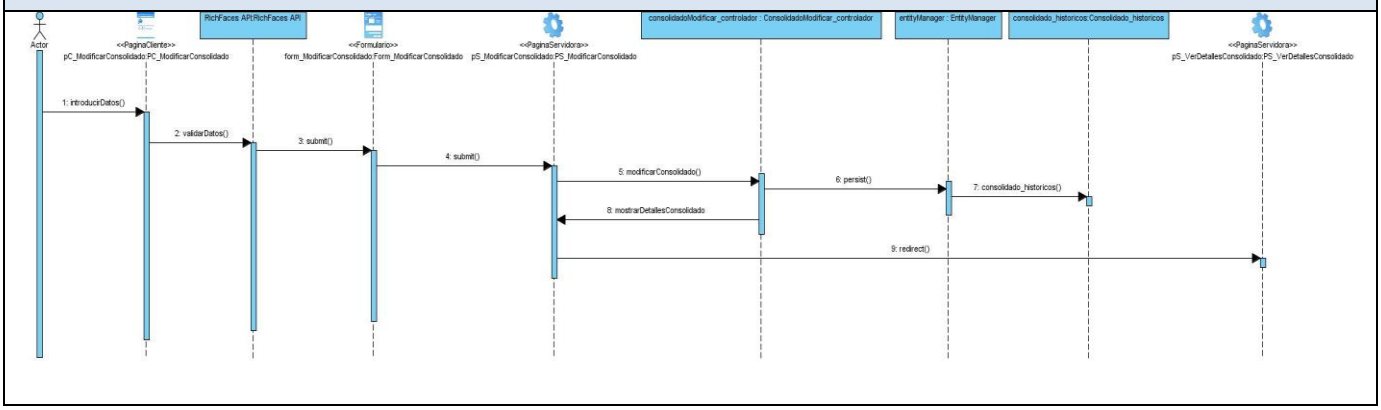
DS_CrearConsolidado



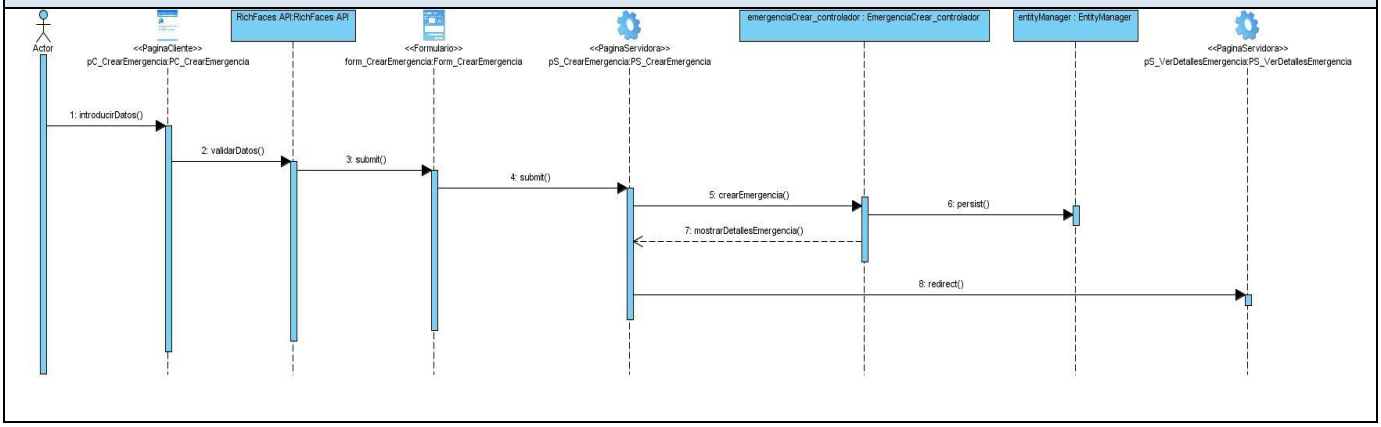
DS_VerDetallesConsolidado



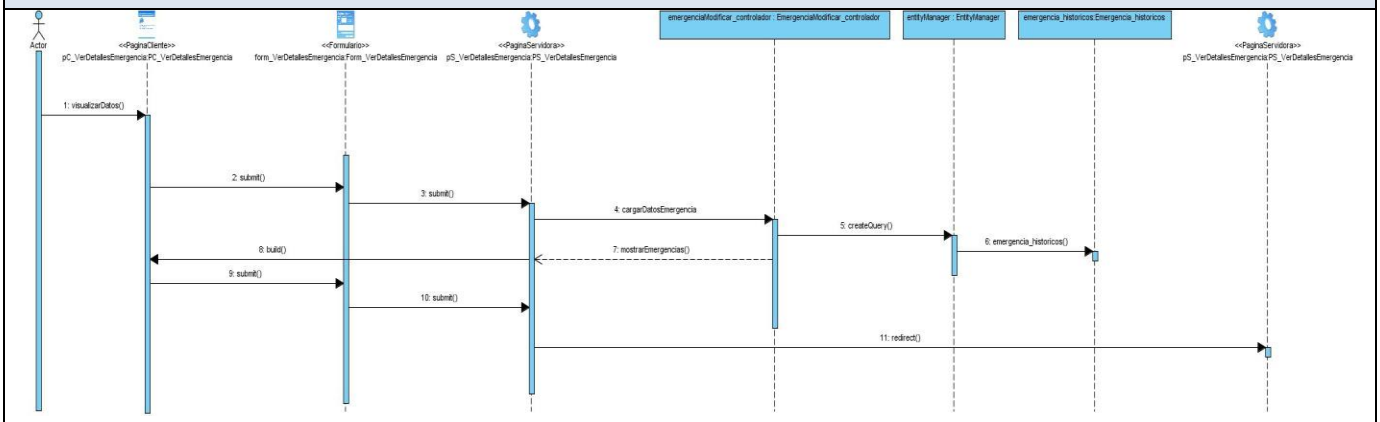
DS_ModificarConsolidado



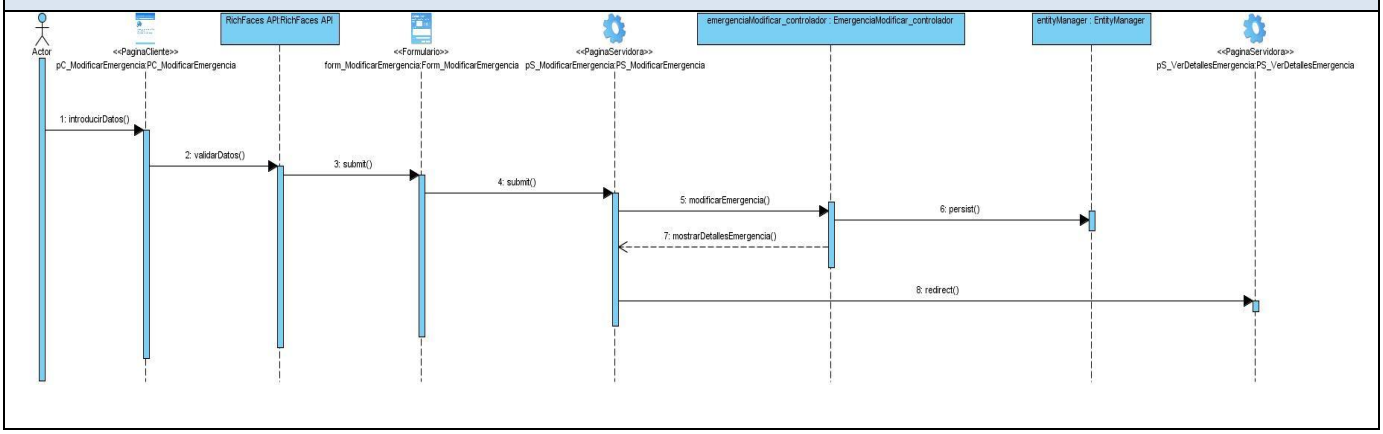
DS_CrearEmergencia



DS_VerDetallesEmergencia



DS_ModificarEmergencia



3.4 **Descripción de las nuevas clases u operaciones necesarias.**

Nombre: HistoricoMovimientoHospitalizadosCrearControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
parametro	String
historicosMovimientoHospitalizados	HistoricosMovimientoHospitalizados_historicos
sourceServicios	ArrayList<String>
servicioSeleccionado	String
sourceDepartamentos	ArrayList<String>
departamentoSeleccionado	String
razon	String
sourceMeses	ArrayList<String>
mesSeleccionado	String
annos	ArrayList<String>
annoSeleccionado	Integer
valor	Integer
Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	validarExistencia() : Boolean

Descripción:	Valida que el histórico no se haya adicionado anteriormente.
Nombre:	cancelarAdicionHistorico() : void
Descripción:	Cancela la adición del histórico.
Nombre:	limpiar() : void
Descripción:	Restaura el valor por defecto de varios atributos.
Nombre:	adicionarHistorico() : void
Descripción:	Adiciona el histórico a la lista de históricos a persistir.
Nombre:	eliminarHistorico(Consolidado_historicos consolidado_historicos) : void
Descripción:	Elimina un histórico de la lista de históricos a persistir.
Nombre:	persistirHistoricos() : void
Descripción:	Persiste en la base de datos los históricos.

Tabla 1.1. Descripción de la clase HistoricoMovimientoHospitalizadosCrearControlador

Nombre: HistoricoMovimientoHospitalizadosModificarControlador	
Tipo de clase : Controladora	
cid	Integer
parametro	String
idHistorico	Integer
historicosMovimientoHospitalizados	HistoricosMovimientoHospitalizados_historicos
historicosMovimientoHospitalizadosEliminados	HistoricosMovimientoHospitalizados_historicos
sourceServicios	ArrayList<String>
servicioSeleccionado	String

sourceDepartamentos	ArrayList<String>
departamentoSeleccionado	String
razon	String
sourceMeses	ArrayList<String>
mesSeleccionado	String
annos	ArrayList<String>
annoSeleccionado	Integer
valor	Integer

Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	validarExistencia() : Boolean
Descripción:	Valida que el histórico no se haya adicionado anteriormente.
Nombre:	cancelarAdicionHistorico() : void
Descripción:	Cancela la adición del histórico.
Nombre:	limpiar() : void
Descripción:	Restaura el valor por defecto de varios atributos.
Nombre:	adicionarHistorico() : void
Descripción:	Adiciona el histórico a la lista de históricos a persistir.
Nombre:	eliminarHistorico(MovimientoHospitalizados_historicos movimientoHospitalizados _historicos) : void
Descripción:	Elimina un histórico de la lista de históricos a persistir.

Nombre:	persistirHistoricos() : void
Descripción:	Persiste en la base de datos los históricos.

Tabla 1.1. Descripción de la clase HistoricoMovimientoHospitalizadosModificarControlador

Nombre: HistoricoMovimientoHospitalizadosVerDetallesControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
historicosMovimientoHospitalizados	HistoricosMovimientoHospitalizados_historicos
idHistorico	Integer
cid	Integer
parametro	String
Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	eliminarHistorico(MovimientoHospitalizados_historicos movimientoHospitalizados _historicos) : void
Descripción:	Elimina en histórico de la lista de históricos a persistir.

Tabla 1.2. Descripción de la clase HistoricoMovimientoHospitalizadosVerDetallesControlador

Nombre: HistoricoConsultaExternaCrearControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer

parametro	String
historicosConsultaExterna	HistoricosConsultaExterna_historicos
sourceConsultas	ArrayList<String>
consultaSeleccionada	String
sourceMeses	ArrayList<String>
mesSeleccionado	String
annos	ArrayList<String>
annoSeleccionado	Integer
valor	Integer

Para cada responsabilidad:

Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	validarExistencia() : Boolean
Descripción:	Valida que el histórico no se halla adicionado anteriormente.
Nombre:	cancelarAdicionHistorico() : void
Descripción:	Cancela la adición del histórico.
Nombre:	limpiar() : void
Descripción:	Restaura el valor por defecto de varios atributos.
Nombre:	adicionarHistorico() : void
Descripción:	Adiciona el histórico a la lista de históricos a persistir.
Nombre:	eliminarHistorico(ConsultaExterna_historicos consultaExterna_historicos) : void
Descripción:	Elimina en histórico de la lista de históricos a persistir.

Nombre:	persistirHistoricos() : void
Descripción:	Persiste en la base de datos los históricos.

Tabla 1.3. Descripción de la clase HistoricoConsultaExternaCrearControlador

Nombre: HistoricoConsultaExternaModificarControlador	
Tipo de clase : Controladora	
cid	Integer
parametro	String
idHistorico	Integer
historicosConsultaExterna	HistoricosConsultaExterna_historicos
historicosConsultaExternaEliminados	HistoricosConsultaExterna_historicos
sourceConsultas	ArrayList<String>
consultaSeleccionada	String
sourceMeses	ArrayList<String>
mesSeleccionado	String
annos	ArrayList<String>
annoSeleccionado	Integer
valor	Integer

Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	validarExistencia() : Boolean
Descripción:	Valida que el histórico no se halla adicionado anteriormente.

Nombre:	cancelarAdicionHistorico() : void
Descripción:	Cancela la adición del histórico.
Nombre:	limpiar() : void
Descripción:	Restaura el valor por defecto de varios atributos.
Nombre:	adicionarHistorico() : void
Descripción:	Adiciona el histórico a la lista de históricos a persistir.
Nombre:	eliminarHistorico(ConsultaExterna_historicos consultaExterna _historicos) : void
Descripción:	Elimina un histórico de la lista de históricos a persistir.
Nombre:	persistirHistoricos() : void
Descripción:	Persiste en la base de datos los históricos.

Tabla 1.4. Descripción de la clase HistoricoConsultaExternaModificarControlador

Nombre: HistoricoConsultaExternaVerDetallesControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
historicosConsultaExterna	HistoricosConsultaExterna_historicos
idHistorico	Integer
cid	Integer
parametro	String
Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.

Nombre:	eliminarHistorico(ConsultaExterna_historicos consultaExterna _historicos) : void
Descripción:	Elimina en histórico de la lista de históricos a persistir.

Tabla 1.5. Descripción de la clase HistoricoConsultaExternaVerDetallesControlador

Nombre: HistoricoEmergenciaCrearControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
parametro	String
historicosEmergencia	HistoricosEmergencia_historicos
sourceConsultas	ArrayList<String>
consultaSeleccionada	String
sourceMeses	ArrayList<String>
mesSeleccionado	String
annos	ArrayList<String>
annoSeleccionado	Integer
valor	Integer
Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	validarExistencia() : Boolean
Descripción:	Valida que el histórico no se haya adicionado anteriormente.

Nombre:	cancelarAdicionHistorico() : void
Descripción:	Cancela la adición del histórico.
Nombre:	limpiar() : void
Descripción:	Restaura el valor por defecto de varios atributos.
Nombre:	adicionarHistorico() : void
Descripción:	Adiciona el histórico a la lista de históricos a persistir.
Nombre:	eliminarHistorico(Emergencia_historicos emergencia_historicos) : void
Descripción:	Elimina en histórico de la lista de históricos a persistir.
Nombre:	persistirHistoricos() : void
Descripción:	Persiste en la base de datos los históricos.

Tabla 1.6. Descripción de la clase HistoricoEmergenciaCrearControlador

Nombre: HistoricoEmergenciaModificarControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
idHistorico	Integer
cid	Integer
parametro	String
historicosEmergencia	HistoricosEmergencia_historicos
historicosEmergenciaEliminados	HistoricosEmergencia_historicos
sourceConsultas	ArrayList<String>
consultaSeleccionada	String

sourceMeses	ArrayList<String>
mesSeleccionado	String
annos	ArrayList<String>
annoSeleccionado	Integer
valor	Integer
Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	validarExistencia() : Boolean
Descripción:	Valida que el histórico no se haya adicionado anteriormente.
Nombre:	cancelarAdicionHistorico() : void
Descripción:	Cancela la adición del histórico.
Nombre:	limpiar() : void
Descripción:	Restaura el valor por defecto de varios atributos.
Nombre:	adicionarHistorico() : void
Descripción:	Adiciona el histórico a la lista de históricos a persistir.
Nombre:	eliminarHistorico(Emergencia_historicos emergencia_historicos) : void
Descripción:	Elimina en histórico de la lista de históricos a persistir.
Nombre:	persistirHistoricos() : void
Descripción:	Persiste en la base de datos los históricos.

Tabla 1.7. Descripción de la clase HistoricoEmergenciaModificarControlador

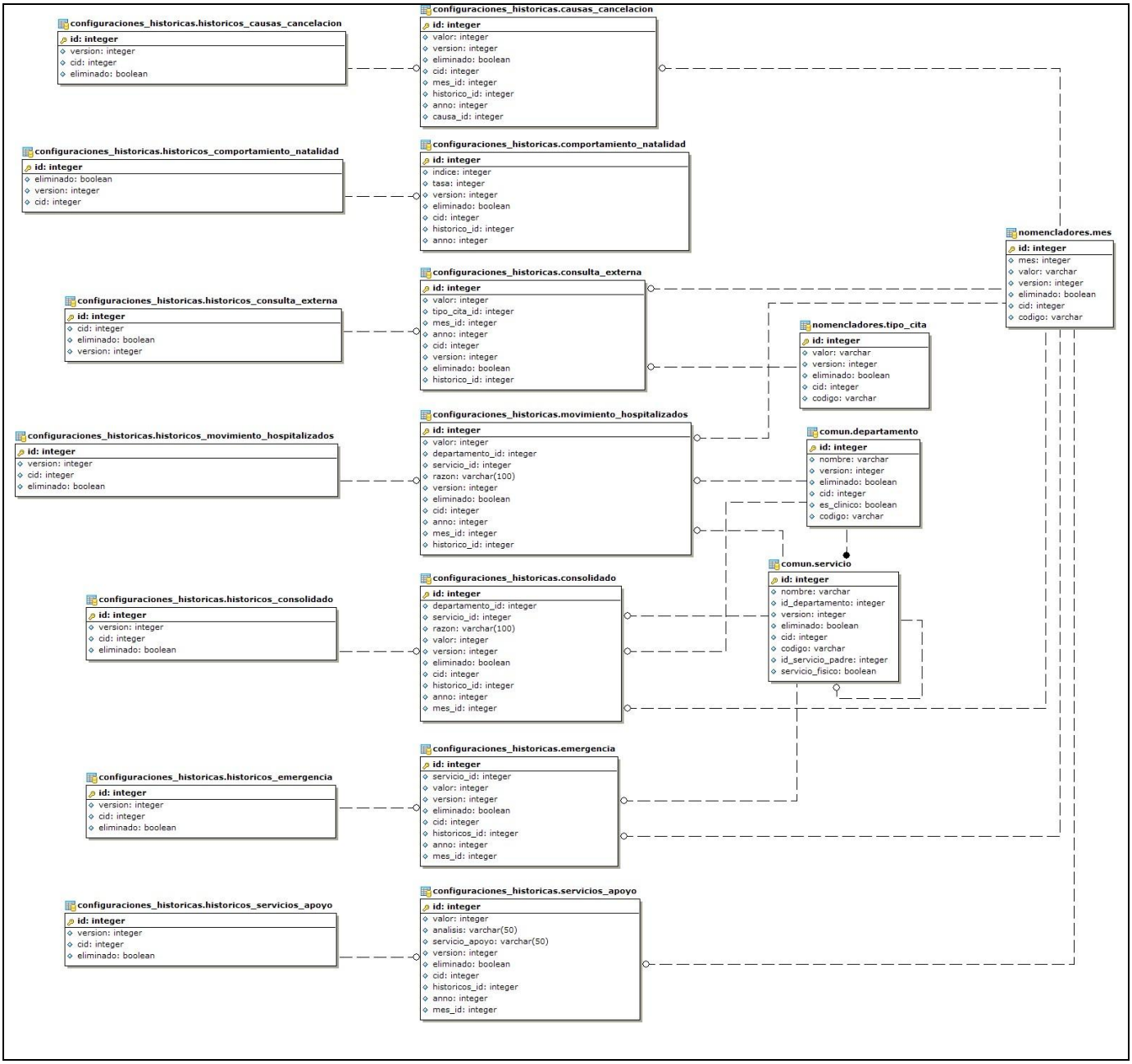
Nombre: HistoricoEmergenciaVerDetallesControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
historicosEmergencia	HistoricosEmergencia_historicos
idHistorico	Integer
cid	Integer
parametro	String
Para cada responsabilidad:	
Nombre:	inicializar() : void
Descripción:	Inicializa los datos necesarios para crear el histórico.
Nombre:	eliminarHistorico(Emergencia_historicos emergencia _historicos) : void
Descripción:	Elimina en histórico de la lista de históricos a persistir.

Tabla 1.8. Descripción de la clase HistoricoEmergenciaVerDetallesControlador

3.5 **Modelo de datos**

Un Modelo de Datos no es más que un conjunto de conceptos que definen la estructura de una base de datos, ya sean los datos, las relaciones entre estos y las restricciones que rigen dichos datos, sirviendo de representación física de las tablas en una base de datos determinada. Uno de los aportes que brindan los modelos de datos, es el hecho de ofrecer la base conceptual para llevar a cabo el diseño de los sistemas que trabajan con bases de datos. Además cabe destacar que poseen un conjunto de operaciones básicas para la realización de consultas y actualización de los datos.

Modelos de datos



3.6 **Breve valoración de las técnicas de validación**

Existen un gran número de técnicas de validación orientadas a tratar de impedir un comportamiento erróneo de la aplicación y los datos que maneja. El tratamiento de los errores y las excepciones son fundamentales para evitar que eventos en tiempo de ejecución atenten contra el correcto funcionamiento del sistema. Esto posibilita que aun ante la ocurrencia de un error al ser lanzada la excepción, se maneje el mismo, posibilitando que continúe la ejecución de las tareas definidas para la aplicación.

En los .properties, se encuentran todos los mensajes a mostrar en caso de errores y en los page.xml las páginas a las que el sistema redirecciona en caso de ser necesario.

Normalmente al realizarse el desarrollo de cualquier aplicación, se lleva a cabo varias validaciones a los datos que son introducidos por los usuarios del sistema. Con el objetivo de facilitar este tipo de trabajo se propone la utilización del framework JSF, el cual permite definir las validaciones en la vista, por ejemplo, la definición de los campos requeridos, pero esto trae consigo el hecho de que se repiten las validaciones una y otra vez. Este inconveniente se puede solucionar gracias a la utilización de Hibernate Validator, al definirse las validaciones en los objetos del negocio, y de esta forma dichas validaciones pueden ser llamadas en el momento necesario. En este caso se usa la clase ValidadorGeneral que permite validar la entrada de los tipos de datos correctos al intentar hacer una petición al servidor.

El utilizar Hibernate como herramienta de mapeo objeto-relacional, permite validaciones en el proceso de acceso a datos, por ejemplo, al ponerle una anotación del tipo @Transaccional a un método que acceda a la base de datos prevé que si ocurre algún error en las operaciones los cambios no surtan efecto en el gestor y se mantenga el estado de los datos.

3.7 **Descripción de las tablas**

Nombre: historicos_movimiento_hospitalizados		
Descripción: Agrupa a los históricos registrados en un mismo momento.		
Atributo	**Tipo**	**Descripción**
id	Integer	Clave.
eliminado	Boolean	Muestra si ha sido eliminado el grupo de históricos.
cid	Integer	Valor asignado por la bitácora.
version	Integer	Versión del histórico.

Nombre: movimiento_hospitalizados_epidemia

Descripción: (2)		
Atributo	**Tipo**	**Descripción**
id	Integer	Clave.
eliminado	Boolean	Muestra si ha sido eliminado el histórico.
cid	Integer	Valor asignado por la bitácora.
version	Integer	Version del histórico.
historico_id	Integer	Id del histórico asociado.
servicio_id	Integer	Servicio asociado.
mes_id	Integer	Mes asociado.
departamento_id	Integer	Departamento asociado.
razon	String	Razón del movimiento.
valor	Integer	Valor numérico.

Nombre: historicos_consolidado		
Descripción: Agrupa a los históricos registrados en un mismo momento.		
Atributo	**Tipo**	**Descripción**
id	Integer	Clave.
eliminado	Boolean	Muestra si ha sido eliminado el grupo de históricos.
cid	Integer	Valor asignado por la bitácora.
version	Integer	Versión del histórico.

Nombre: consolidado
Descripción: Históricos de consolidados

Atributo	Tipo	Descripción
id	Integer	Clave.
eliminado	Boolean	Muestra si ha sido eliminado el histórico.
cid	Integer	Valor asignado por la bitácora.
version	Integer	Versión del histórico.
historico_id	Integer	Id del histórico asociado.
servicio_id	Integer	Servicio asociado.
mes _id	Integer	Mes asociado.
departamento_id	Integer	Departamento asociado.
razon	String	Razón del consolidado.
valor	Integer	Valor numérico.

Nombre: historicos_ consulta_externa		
Descripción: Agrupa a los históricos registrados en un mismo momento.		
Atributo	**Tipo**	**Descripción**
id	Integer	Clave.
eliminado	Boolean	Muestra si ha sido eliminado el grupo de históricos.
cid	Integer	Valor asignado por la bitácora.
version	Integer	Versión del histórico.

Nombre: consulta_externa		
Descripción: Históricos de consultas externas		
Atributo	**Tipo**	**Descripción**

id	Integer	Clave.
eliminado	Boolean	Muestra si ha sido eliminado el histórico.
cid	Integer	Valor asignado por la bitácora.
version	Integer	Versión del histórico.
tipoCita_id	Integer	Servicio asociado.
mes_id	Integer	Mes asociado.
anno	Integer	Año de la consulta.
historico_id	Integer	Id del histórico asociado.
valor	Integer	Valor numérico.

3.8 **Vista de Implementación. (Diagrama de Componentes)**

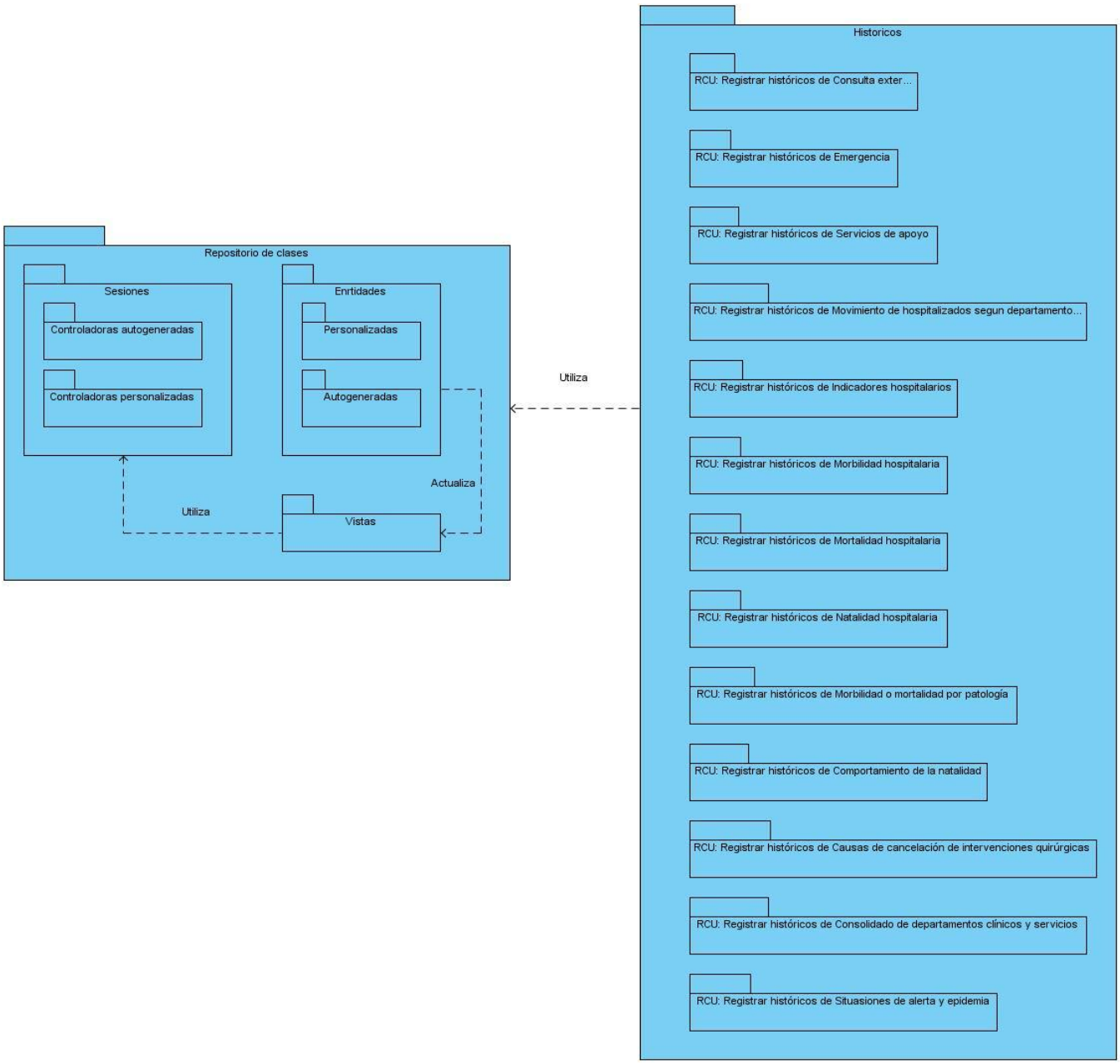


Figura 1.1. Diagrama de componentes.

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.

Para revelar la calidad de un producto de software es necesario realizarle un conjunto de pruebas orientadas a encontrar errores en su desarrollo. Las pruebas de software se integran dentro de las diferentes fases del ciclo de vida del software para garantizar el refinamiento continuo de la solución. Estas permiten validar los requerimientos y las acciones del diseño. Verificar la interacción de componentes, así como la integración adecuada de los componentes. Este flujo de trabajo brinda soporte para encontrar, documentar y solucionar defectos en el sistema.

El proceso de pruebas no va orientado a demostrar la ausencia de fallos en el software, sino más bien a tratar de encontrar todos los fallos posibles. Para ello es necesario diseñar casos de pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Un caso de prueba tiene éxito si descubre un error no detectado hasta entonces.

Un caso de prueba es un conjunto de condiciones o variables cuyo resultado determina si el cumplimiento un requisito es parcial o completamente satisfactorio. Por lo tanto debe haber al menos un caso de prueba para cada requisito del sistema. Como característica fundamental un caso de prueba tiene una entrada conocida y una salida esperada los cuales son definidos antes de realizar la prueba.

4.1 Prueba de caja negra

Las pruebas se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra pretende demostrar que:

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

Se derivan conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del programa. La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

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

Los casos de prueba deben satisfacer los siguientes criterios:

- Reducir, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales.
- Que digan algo sobre la presencia o ausencia de clases de errores. (20)

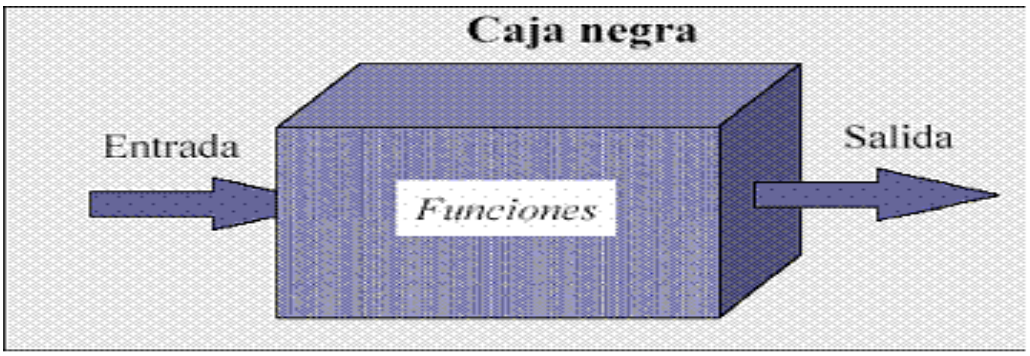


Figura 5.1. Enfoque de diseño de pruebas de caja negra.

En este tipo de pruebas existen algunas técnicas y criterios que pueden ayudar a seleccionar el conjunto de casos de prueba óptimo. Por ejemplo:

- Técnica de la partición de equivalencia: esta técnica divide el campo de entrada en clases de equivalencia es decir clases que representan un conjunto de datos entonces es suficiente probar con un valor de cada clase, pues los demás datos de la misma clase son equivalentes.
- Técnica del análisis de valores límites: esta técnica complementa a la anterior se basa en que muchos errores aparecen en torno a los puntos de cambio de clase de equivalencia por ejemplo si una condición de entrada especifica un rango de valores, se deben generar casos para los extremos del rango y casos no validos para las situaciones justo más allá de los

extremos .Esta técnica prueba la habilidad del software para manipular datos que se encuentran dentro de los límites aceptables.

- Técnica de grafos de causa-efecto: proporciona una concisa representación de las condiciones lógicas y sus correspondientes acciones. En esta técnica se listan las causas o condiciones de entrada y los efectos o acciones, asignando un identificador a cada uno de ellos. A partir de estos se desarrolla un grafo causa – efecto. El grafo se convierte en una tabla de decisión y se convierten las reglas de la tabla de decisión a casos de prueba. Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica ha sido escogida para realizar estas pruebas es la de partición de equivalencia, la misma permite representar las clases de equivalencias en grupos de estados válidos e inválidos para las condiciones de entrada. Esto se logra representando en clases de equivalencia el dominio de entrada de la aplicación.

4.2 Descripción de los casos de pruebas

4.2.1 Crear histórico de consulta externa

Descripción General

El caso de uso inicia cuando el actor accede a la opción Crear histórico de consulta externa el sistema brinda la posibilidad de introducir o seleccionar los datos para crear el histórico de consulta externa el actor introduce o selecciona los datos del histórico de consulta externa el sistema crea el histórico de consulta externa el caso de uso termina.

Condiciones de Ejecución:

No existe

Secciones a probar en el Caso de Uso:

Escenarios del crear histórico de consulta externa	Descripción de la funcionalidad	Flujo Central
EC 1: Crear histórico de consulta externa	Crear un histórico de consulta externa satisfactoriamente.	Se selecciona la opción Crear histórico de consulta externa. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar.

		Se crea el histórico de consulta externa. Muestra la interfaz Ver detalles de histórico de consulta externa. Ver DCP: **Ver detalles de histórico de consulta externa.**
EC 2: Cancelar operación	Cancelar la opción de Crear histórico de consulta externa.	Se selecciona la opción Crear histórico de consulta externa. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incompletos muestra un indicador sobre los campos incompletos.	Se selecciona la opción Crear de histórico de consulta externa. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.
EC 4: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incorrectos muestra un indicador sobre los campos incorrectos.	Se selecciona la opción Crear de histórico de consulta externa. Se introducen los datos incorrectos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incorrectos.

SC 1: Crear histórico de consulta externa

Id del escenario	EC 1	EC 2	EC 3	EC 4

Escenario	Crear histórico de consulta externa	Cancelar operación	Existen datos incompletos	Existen datos incorrectos
Variable 1 **Anno** **Variable 2** **Mes** **Variable 3** **Consulta** **Variable 4** **Valor (numérico)**	2008 Enero Laboratorio 55		2009 Marzo Laboratorio	2009 Marzo Laboratorio lala
Botón 1 (Aceptar)	NA		NA	NA
Botón 1 (Cancelar)		NA		
Respuesta del Sistema	Crea el histórico de consulta externa y se visualiza la interfaz ver detalles.	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.	Muestra un indicador sobre los campos incorrectos.

4.2.2 **Modificar histórico de consulta externa**

Descripción General

El caso de uso se inicia cuando el actor accede a la opción Modificar históricos de consulta externa el sistema muestra los datos de los históricos de consulta externa y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos de los históricos de consulta externa el caso de uso termina.

Condiciones de Ejecución:

No existe

Secciones a probar en el Caso de Uso:

Escenarios del modificar históricos de consulta externa	**Descripción de la funcionalidad**	**Flujo Central**

EC 1: Modificar históricos de consulta externa	Modificar un grupo de históricos de consulta externa satisfactoriamente.	Muestra la interfaz para modificar históricos de consulta externa. Se modifican los datos correspondientes. Se selecciona la opción Aceptar. Se modifica los históricos de consulta externa. Muestra la interfaz Ver detalles de históricos de consulta externa Ver DCP **Ver detalles de históricos de consulta externa.**
EC 2: Cancelar operación	Cancelar la opción de Modificar históricos de consulta externa.	Muestra la interfaz para modificar históricos de consulta externa. Se introducen los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incompletos muestra un indicador sobre los campos incompletos.	Muestra la interfaz para modificar históricos de consulta externa. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.
EC 4: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incorrectos muestra un indicador sobre los campos incorrectos.	Muestra la interfaz para modificar históricos de consulta externa. Se introducen los datos incorrectos. Se selecciona la opción Aceptar.

		Muestra un indicador sobre los campos incorrectos.

SC 1: Modificar históricos de consulta externa

Id del escenario	EC 5	EC 6	EC 7	EC 8
Escenario	Modificar históricos de consulta externa satisfactoriamente	Cancelar operación	Existen datos incompletos	Existen datos incorrectos
Variable 1 **Anno** **Variable 2** **Mes** **Variable 3** **Consulta** **Variable 4** **Valor (numérico)**	2009 Marzo Laboratorio 60		2009 Marzo Laboratorio	2009 Marzo Laboratorio lala
Botón 2 (Aceptar)	NA		NA	NA
Botón 3 (Cancelar)		NA		
Respuesta del Sistema	Modifica los históricos de consulta externa y se visualiza la interfaz ver detalles.	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.	Muestra un indicador sobre los campos incorrectos.

4.2.3 **Ver detalles de históricos de consulta externa**

Descripción General

El caso de uso inicia se hace necesario ver detalles de históricos de consulta externa, el sistema muestra los datos de los históricos de consulta externa, el caso de uso termina.

Condiciones de Ejecución:

No existe

Secciones a probar en el Caso de Uso:

Escenarios del ver detalles de históricos de consulta externa	**Descripción de la funcionalidad**	**Flujo Central**
EC 5: Ver detalles de históricos de consulta externa	Ver detalles de históricos de consulta externa satisfactoriamente.	Se muestra la interfaz Ver detalles de históricos de consulta externa satisfactoriamente. Se muestran los datos de los históricos de consulta externa. Selecciona la opción Salir. Regresa a la vista de la página inicial del módulo Estadísticas.
EC 6: Modificar históricos de consulta externa	Permite acceder al caso de uso modificar históricos de consulta externa.	Se muestra la interfaz Ver detalles de históricos de consulta externa. Se muestran los datos de los históricos de consulta externa Se selecciona la opción Modificar. Ver DCP **Modificar históricos de consulta externa.**
EC 7: Eliminar históricos de consulta externa	Permite acceder al caso de uso eliminar históricos de consulta externa.	Se muestra la interfaz Ver detalles de históricos de consulta externa. Se muestran los datos de los históricos de consulta externa. Se selecciona la opción Eliminar. Ver DCP **Eliminar históricos de consulta externa.**

SC 1: Ver detalles de históricos de consulta externa

Id del escenario	EC 9	EC 10	EC 11
Escenario	Ver detalles de históricos de consulta externa satisfactoriamente	Modificar históricos de consulta externa.	Eliminar históricos de consulta externa.
(Salir)	NA		
(Modificar)		NA	
(Eliminar)			NA
Respuesta del Sistema	Regresa a la vista de la página inicial del módulo Estadísticas.	Muestra la interfaz para modificar.	Muestra la interfaz para eliminar.

4.2.4 Buscar históricos de consulta externa

Descripción General

El caso de uso inicia cuando el actor accede a la opción Buscar históricos de consulta externa el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar los históricos de consulta externa el actor introduce los datos que considera como criterios para realizar una búsqueda, el sistema busca y muestra las históricos de consulta externa que cumplen dichos criterios, el caso de uso termina.

Condiciones de Ejecución:

No existe

Secciones a probar en el Caso de Uso:

Escenarios del buscar histórico de consulta externa	Descripción de la funcionalidad	Flujo Central
EC 8: Buscar histórico de consulta externa	Buscar un histórico de consulta externa dado criterios.	Muestra la interfaz para buscar el histórico de consulta externa. Se introducen los datos correspondientes a la búsqueda

		simple. Se selecciona la opción Buscar. Muestra un listado de los históricos de consulta externa que cumplen con los criterios de búsqueda.
EC 9: No se encuentra información	No se encuentra información que cumpla con los criterios de búsqueda.	Muestra la interfaz para buscar el histórico de consulta externa Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra el mensaje de información “No se encontró información que cumpla con los criterios de búsqueda.”.
EC 10: Cancelar operación	Cancelar la opción de Buscar históricos de consulta externa.	Muestra la interfaz para buscar el histórico de consulta externa. Se introducen los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.
EC 11: Ver datos de histórico de consulta externa	Ver los datos de un histórico de consulta externa	Muestra la interfaz para buscar el histórico de consulta externa. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de los históricos de consulta externa que cumplen con los criterios de

		búsqueda. Se selecciona la opción Ver. Ver DCP **Ver datos de histórico de consulta externa**
EC 12: Modificar histórico de consulta externa	Permite acceder al caso de uso modificar histórico de consulta externa.	Muestra la interfaz para buscar el histórico de consulta externa. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de los históricos de consulta externa que cumplen con los criterios de búsqueda. Se selecciona la opción Modificar. Ver DCP **Modificar histórico de consulta externa.**
EC 13: Eliminar histórico de consulta externa	Permite acceder al caso de uso eliminar histórico de consulta externa	Muestra la interfaz para buscar el histórico de consulta externa. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de los históricos de consulta externa que cumplen con los criterios de búsqueda. Se selecciona la opción Eliminar. Ver DCP **Eliminar histórico de consulta externa.**
EC 14: Ordenar	Permite ordenar el resultado ascendente o descendentemente por un atributo.	Muestra la interfaz para buscar el histórico de consulta externa.

		Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de los históricos de consulta externa que cumplen con los criterios de búsqueda. Se ordenan los campos del listado por atributos.

SC 1: Buscar histórico de consulta externa

Id del escenario	EC 12	EC 13	EC 14	EC 15	EC 16	EC 17	EC 18
Escenario	Buscar histórico de consulta externa	No se encuentra información	Cancelar operación	Ver datos de histórico de consulta externa	Modificar histórico de consulta externa	Eliminar histórico de consulta externa	Ordenar
Variable 1 **Anno** **Variable 2** **Mes** **Variable 3** **Consulta** **Variable 4** **Valor (numérico)**	2009 Marzo Laboratorio 60	2007					
Variable 5							NA

Anno **Variable 6** **Mes** **Variable 7** **Consulta** **Variable 8** **Valor (numérico)**							
(Buscar)	NA	NA					
(Cancelar)			NA				
(Ver)				NA			
(Modificar)					NA		
(Eliminar)						NA	
Respuesta del Sistema	• anno. Muestra el año seleccionado. • mes. Muestra el mes seleccionado. • consulta. Muestra la consulta seleccionada. • valor. Muestra el valor ingresado **(numérico)** • Y los iconos para modificar y eliminar	Muestra el mensaje de información: "No se encuentra información que cumpla con los criterios de búsqueda.".	Regresa a la vista anterior.	Muestra la interfaz para ver datos.	Muestra la interfaz para modificar.	Muestra la interfaz para eliminar.	Muestran los atributos del listado ordenados.

En este capítulo se describieron las pruebas realizadas a la aplicación, orientadas a encontrar errores en su desarrollo lo que contribuye a garantizar la calidad de las funcionalidades implementadas.

Conclusiones

Una vez concluida la presente investigación se determina lo siguiente:

- Los sistemas analizados no se adaptan a las necesidades del Sistema de Información Hospitalario alas HIS por lo que se propone una solución orientada a la entrada manual de datos.
- Se determinaron las tecnologías, herramientas y patrones a utilizar, las cuales permitieron la construcción de un sistema robusto y flexible, adaptable a diferentes entornos de despliegue.
- La utilización de pautas para el proceso de desarrollo, garantizó uniformidad y homogeneidad en los artefactos obtenidos a partir de este.
- La implementación de las funcionalidades para la gestión de datos históricos y su incorporación en los reportes que brinda el módulo Estadísticas contribuye a un mejor aprovechamiento de los mismos.
- La ejecución de pruebas de caja negra garantizó la calidad necesaria para la futura explotación de las funcionalidades implementadas.

Recomendaciones

Se recomienda:

- Implementar las funcionalidades que permitan la incorporación de datos históricos en todos los módulos del sistema alas HIS que necesiten esta información.

Referencias bibliográficas

1. MIC.Ministerio de la informática y las comunicaciones. [Online] [Cited: Diciembre 10, 2010.] http://www.mic.gov.cu/sitiomic/servlet/hinfosoc.

2. **Santoja Herrero, Joaquín.** *La experiencia de Kewan - Cosmosalud.* España : s.n., 2008.

3. SIGHO. Sistema de Información para la Gerencia Hospitalaria. [Online] [Cited: Enero 10, 2010.] http://www.sigho.gob.mx/quees.htm.

4. X-HIS.Sistema de Información para la Gerencia Hodpitalaria. [Online] [Cited: Diciembre 20, 2010.] http://www.isoftsanidad.es/dummy/xHIS_080108.pdf.

5. OpenHIS. [Online] [Cited: Noviembre 3, 2010.] http://www.openhis.com.ar/blog/openhis-solucion-a-u-viejo-problema/.

6. Eclipse . [Online] Mayo 2009. http://plataformaeclipse.com/.

7. PostgreSQL About. [Online] [Cited: Enero 16, 2011.] http://www.postgresql.org/about..

8. Jboss Seam. [Online] [Cited: Enero 15, 2011.] http://seamframework.org/.

9. Jboss. [Online] 2008. [Cited: Diciembre 20, 2010.] http://www.jboss.org/.

10. Hibernate. [Online] [Cited: Enero 3, 2011.] http://www.hibernate.org/about/why-hibernate.

11. **Medina, Delia Ing.** Manual de ireport y jasper . [Online] [Cited: Septiembre 25, 2010.] http://es.geocities.com/mdelia_99/ireport/manual_de_ireport_y_jasper.html.

12. **Sanromán, Javi.** [Online] [Cited: Enero 21, 2011.] http://www.jsanroman.net/2007/11/20/%C2%BFque-es-jasper-reports-2/.

13. **Junta de Andalucía.** RichFaces. [Online] Febrero 2009. [Cited: Septiembre 26, 2010.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces#caracteristicas%20Seam.

14. **Arnold, Ken. Gosling, James and Holmes, David.** Java(TM) Programming Language, The (4th Edition). [Online] [Cited: Enero 17, 2011.] http://portal.acm.org/citation.cfm?id=1051069.

15. Metodologías de desarrollo de software. [Online] [Cited: Diciembre 13, 2010.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.

16. **Kruchten, Philippe.** What Is the Rational Unified Process? [Online] [Cited: Enero 9, 2011.] http://www.ida.liu.se/~HKGC10/www-pub-old/vt05/art_rup/WhatIstheRationalUnifiedProcessJan01.pdf.

17. Instituto Tecnologico de la Laguna. [Online] [Cited: Enero 15, 2011.] http://www.docstoc.com/docs/21085442/capitulo-IV-definici%c3%93n-del-lenguaje-de-modelado-unificado-(uml).

18. JBOSS. [Online] [Cited: Enero 19, 2011.] http://www.jboss.com/products/jbpm/.

19. **Piattini, M.** Análisis y diseño detellado de aplicaciones Informáticas de Gestión. Capítulos 6 y 7.

20. IEEE-1471. The Architecture of Software-Intensive Systems. [Online] http://www.iso-architecture.org/ieee-1471/architecture_track.html.

Bibliografía

1. MIC.Ministerio de la informática y las comunicaciones. [En línea] [Citado el: 10 de Diciembre de 2010.] http://www.mic.gov.cu/sitiomic/servlet/hinfosoc.

2. Kofax Capture. Software de captura de documentos. [En línea] [Citado el: 10 de Enero de 2010.] http://www.kofax.com/es/capture/.

3. ABBYY FlexiCapture 9.0. [En línea] [Citado el: 10 de enero de 2011.] http://www.abbyy.com/data_capture_software/.

4. ABBYY FlexiCapture Engine 9.0. [En línea] [Citado el: 11 de febrero de 2011.] http://www.abbyy.com/flexicapture_engine/.

5. EpiData. [En línea] [Citado el: 15 de enero de 2011.] http://www.epidata.dk/downloads/epidataanalysis_introduction_es.pdf.

6. Eclipse . [En línea] Mayo de 2009. http://plataformaeclipse.com/.

7. PostgreSQL About. [En línea] [Citado el: 16 de Enero de 2011.] http://www.postgresql.org/about..

8. Jboss Seam. [En línea] [Citado el: 15 de Enero de 2011.] http://seamframework.org/.

9. JBOSS. [En línea] [Citado el: 19 de Enero de 2011.] http://www.jboss.com/products/jbpm/.

10. Hibernate. [En línea] [Citado el: 3 de Enero de 2011.] http://www.hibernate.org/about/why-hibernate.

11. **Ing. Medina, Delia.** Manual de ireport y jasper . [En línea] [Citado el: 25 de Septiembre de 2010.] http://es.geocities.com/mdelia_99/ireport/manual_de_ireport_y_jasper.html..

12. **Sanromán, Javi.** [En línea] [Citado el: 21 de Enero de 2011.] http://www.jsanroman.net/2007/11/20/%C2%BFque-es-jasper-reports-2/..

13. **Junta de Andalucía.** RichFaces. [En línea] Febrero de 2009. [Citado el: 26 de Septiembre de 2010.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces#caracteristicas%20Seam.

14. **Arnold, Ken, Gosling, James y Holmes, David.** Java(TM) Programming Language, The (4th Edition). [En línea] [Citado el: 17 de Enero de 2011.] http://portal.acm.org/citation.cfm?id=1051069.

15. Metodologías de desarrollo de software. [En línea] [Citado el: 13 de Diciembre de 2010.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.

16. **Kruchten, Philippe.** What Is the Rational Unified Process? [En línea] [Citado el: 9 de Enero de 2011.] http://www.ida.liu.se/~HKGC10/www-pub-old/vt05/art_rup/WhatIstheRationalUnifiedProcessJan01.pdf.

17. Instituto Tecnologico de la Laguna. [En línea] [Citado el: 15 de Enero de 2011.] http://www.docstoc.com/docs/21085442/capitulo-IV-definici%c3%93n-del-lenguaje-de-modelado-unificado-(uml).

18. http://www.dsi.uclm.es/asignaturas/42541/pdf/metodologia_elicitacion.pdf. [En línea] [Citado el: 25 de Octubre de 2010.] http://www.dsi.uclm.es/asignaturas/42541/pdf/metodologia_elicitacion.pdf.

19. IEEE-1471. The Architecture of Software-Intensive Systems. [En línea] http://www.iso-architecture.org/ieee-1471/architecture_track.html.

20. EpiData. [En línea] [Citado el: 3 de Noviembre de 2010.] http://www.epidata.dk/downloads/epidataanalysis_introduction_es.pdf.

21. **Santoja, Joaquín Herrero.** *La experiencia de Kewan - Cosmosalud.* España : s.n., 2008.

22. Jboss. [En línea] 2008. [Citado el: 20 de Diciembre de 2010.] http://www.jboss.org/.

23. **Vidal Ledo, María Lic. y otros.** Información, informática y estadísticas de salud: un perfil de la tecnología de la salud. . [En línea] http://scielo.sld.cu/pdf/aci/v12n4/aci08404.pdf.

24. **Piattini, M.** *Análisis y diseño detellado de aplicaciones Informáticas de Gestión. Capítulos 6 y 7.*

25. Jasper Report. [En línea] [Citado el: 15 de Diciembre de 2010.] http://jasperreports.sourceforge.net/.

26. UML. [En línea] [Citado el: 15 de Noviembre de 2010.] http://www.lcc.uma.es/~av/Publicaciones/04/UMLProfiles-Novatica04.pdf.

Glosario

Base de datos: Serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Consolidado: Resumen que contiene los principales parámetros medidos para un programa en específico. Muestra resultados numéricos o estadísticos en dependencia de la periodicidad con que se elabore.

Entorno de desarrollo integrado (en inglés Integrated Development Environment ('IDE'): Es un programa compuesto por un conjunto de herramientas para un programador.

***Framework*:** En el desarrollo de software, un *framework* es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Gestor de Base de Datos: Es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

Hardware: Es un conjunto de elementos materiales que conforman una computadora, sin embargo, es usual que sea utilizado en una forma más amplia, generalmente para describir componentes físicos de una tecnología

HL7 (Health Level Seven): Es un conjunto de estándares para el intercambio electrónico de información médica.

Log: Es un registro oficial de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y cual evento ocurre para un dispositivo en particular o aplicación.

Navegador web (en inglés, browser): Es un programa o aplicación informática que permite moverse por la web y acceder al contenido de los sitios, blogs, foros, galerías fotográficas, etc., de Internet.

Orientado a Objetos: Significa que el software se organiza como una colección de objetos discretos que contiene tanto estructura de datos como también un comportamiento.

Programa: Orientado a patologías, áreas, entre otros aspectos, que contiene un grupo de directrices que describen los procedimientos y normas establecidas para cada uno de estos. Brinda además un grupo de instrumentos para la recolección de la información.

Servidor: Es una computadora central, de gran capacidad, compartida por las otras computadoras de la red, llamadas Clientes o estaciones de trabajo (workstations), ya que reciben el servicio de almacenar, controlar y compartir la información contenida en el servidor.

XML: Sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.