Universidad de las Ciencias Informáticas

Facultad 7

Título: Desarrollo del Módulo Archivo del Sistema de Información Hospitalaria alas HIS

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

Autoras: Estela López Suárez
Claudia Flores Villa

Tutores: Ing. Pedro Ernesto Salas Oliva
Ing. Yasser Manuel Garbey Bermúdez

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

RESUMEN

En la actualidad hay una tendencia a utilizar la Historia Clínica Electrónica. No obstante ya mayoría de las instituciones hospitalarias cuentan con un gran número de historias clínicas físicas y la digitalización de las mismas resulta un proceso engorroso, que puede acarrear gran pérdida de información. Es por esta razón que en el presente trabajo se propone como objetivo principal el desarrollo del módulo Archivo del Sistema de Información Hospitalaria alas HIS.

El desarrollo del sistema está guiado por el Proceso Unificado de Desarrollo y se basa en tecnologías libres, multiplataforma y sobre una arquitectura en capas, utilizando Java como lenguaje de programación e implementando el patrón de arquitectura Modelo Vista Controlador. Como Sistema de Gestión de Bases de Datos se utiliza PostgreSQL y como servidor de aplicaciones el JBoss Server. Son utilizadas las librerías de componentes web JBoss UI y RichFaces. Para la administración de las reglas y procesos del negocio se utiliza Drools.

Con el desarrollo del sistema se espera facilitar los procesos de gestión de archivo, lograr una mejor organización de la información y optimizar la prestación de servicios tanto a pacientes como a médicos.

El trabajo presentado se encuentra en la etapa de pruebas.

Palabras clave: archivo, historia clínica, Sistema de Información Hospitalaria

ÍNDICE

INTRODUCCIÓN

En los últimos años, las Tecnologías de la Información y las Comunicaciones (TIC) han introducido un cambio de paradigma en el modo en que el hombre ve y hace las cosas. Las mismas constituyen una útil herramienta para enfrentar el reto que representa vivir en una sociedad que se desarrolla exponencialmente, y en la que la atención eficiente a las necesidades de la población es fundamental.

La informática ha ido creciendo de forma vertiginosa y se ha extendido a todos los ámbitos de la vida humana. Conocer las TIC y utilizarlas constituye una necesidad en tiempos en que la automatización de tareas y el acceso y administración, de grandes volúmenes de información, permiten mayor eficiencia y eficacia en las actividades que en una institución se realizan.

Desde la década de los 70's se vienen perfilando en el ámbito médico los primeros sistemas de información “médica” que posteriormente, habrían de dar lugar a los Sistemas de Información Hospitalaria (más conocidos por sus siglas en inglés HIS), muy útiles en la actualidad. Estos facilitan procesos internos, aumentando la calidad de atención a los pacientes y mejorando el desempeño de los trabajadores en los diferentes niveles de atención. (1)

“Los HIS están orientados a satisfacer las necesidades de generación de información para almacenar, procesar e interpretar datos médico-administrativos de cualquier institución hospitalaria. Permiten la optimización de los recursos humanos y materiales y minimizan los inconvenientes burocráticos que pudieran afrontar los pacientes en el proceso de atención médica. Su principal función es apoyar las actividades en los niveles operativos, tácticos y estratégicos dentro de un Hospital, haciendo uso de las computadoras para recabar, almacenar, procesar y comunicar información clínica y administrativa.” (2)

Estos constituyen, en su mayoría, una solución diseñada de forma modular que permite la existencia de una interdependencia entre áreas de trabajo, así como la integración del paciente, el personal médico, el personal de apoyo y el administrativo.

El “alas HIS” es una propuesta de solución informática para la gestión hospitalaria, desarrollada por el Departamento de Gestión Hospitalaria del Centro de Soluciones de Informática Médica (CESIM) de la Universidad de las Ciencias Informáticas.

Un área importante para la prestación de los servicios dentro de la gestión hospitalaria, es el Archivo. Cuando se habla de archivo, se hace referencia al local o departamento donde se conservan los

documentos generados y recibidos por una entidad, como consecuencia de la realización de sus actividades. En particular, el Archivo Hospitalario es el departamento dentro de la institución de salud, responsable de conservar, controlar y custodiar las historias clínicas de los pacientes.

“Archivo es un servicio hospitalario que proporciona a los profesionales sanitarios la documentación necesaria para una correcta atención al paciente y en el cual recae la responsabilidad de gestionar las historias clínicas correctamente. El archivar mal una Historia Clínica puede causar una disminución de la calidad del acto asistencial o su suspensión con claro perjuicio para el paciente.” (3)

La historia clínica es el conjunto de documentos clínicos obtenidos como parte de la atención al paciente (datos, valoraciones, observaciones y tratamientos) durante toda su vida. Estos documentos pueden ser tanto textos como gráficos.

Cada archivo hospitalario cuenta con un registro de historias clínicas de pacientes, en este son inscritos cada uno de los pacientes que cuentan con historia clínica en el centro. Como la historia clínica tiene un alto valor médico, gerencial, legal y académico, diariamente son muchas las solicitadas al departamento de archivo, lo cual conlleva a realizar por parte de los trabajadores del mismo, gran cantidad de consultas a los voluminosos registros de inscripción de pacientes, en aras de localizarlas. También como parte del control y custodia de las mismas, en él se lleva a cabo el mantenimiento del registro de devoluciones y préstamos. Esta actividad permite dar a conocer la localización de las historias clínicas mientras son utilizadas en los diferentes departamentos de un hospital. En muchos archivos hospitalarios también se lleva a cabo el mantenimiento de los indicadores de calidad, basados fundamentalmente en la disponibilidad de las historias clínicas.

Hay algunas desventajas que son comunes para las instituciones que no cuentan con informatización como son: la ilegibilidad y el deterioro de registros, lo cual puede conducir a ineficiencias en las actividades que en ellas se realizan así como la gran cantidad de recursos invertidos: papel, carpetas, bolígrafos, etcétera. Particularmente en el archivo hospitalario, se puede enfrentar dificultad para localizar rápidamente las historias clínicas, de modo que se puedan atender las solicitudes de las mismas eficientemente y además controlar que sean devueltas.

Con el desarrollo actual, son cada vez más los servicios prestados a los pacientes en un centro hospitalario, lo que ha conllevado al crecimiento de la cantidad de información que en torno a ellos se

genera. Este gran volumen de datos afecta notablemente su manipulación, localización, gestión y control en los archivos clínicos.

Para mejorar la calidad de la atención al paciente y disminuir los inconvenientes de legibilidad, accesibilidad y estructura de la información que trae consigo el uso de la historia clínica física o convencional, se ha creado la Historia Clínica Electrónica (HCE). La cual se ha convertido en uno de los objetivos a alcanzar por casi toda institución hospitalaria ante la instalación de un HIS.

Con esta surgen nuevos inconvenientes, por ejemplo, los mecanismos utilizados en la HCE pueden ser vulnerados si alguna persona mal intencionada encuentra un agujero de seguridad, también se da el caso de que el personal de mantenimiento del sistema de HCE tendría acceso a los datos, lo cual también constituye otro posible agujero en la seguridad del sistema. Debido a que su utilización supone un cambio drástico en cuanto al manejo de la información es posible que existan algunos profesionales sanitarios reacios al cambio, pues este les puede suponer una dificultad añadida si no están familiarizados con el manejo de ordenadores o dispositivos de nueva generación. Además, debemos adicionar el coste de poner en funcionamiento el sistema, ya que hay que adquirir nuevos equipos informáticos, aunque con el tiempo estos costes podrían acabar compensándose, ya que una vez implantada la HCE es más económica que la historia clínica tradicional.

No obstante, cuando se lleva a cabo la instalación de un HIS, existen diferentes situaciones que se deben enfrentar. Por ejemplo, en la mayoría de los casos transcurre un tiempo antes de que se pueda emplear únicamente la HCE, pues ya la institución cuenta con gran número de historias clínicas físicas y la digitalización de las mismas constituye un proceso engorroso e ineficiente, sin contar con la gran pérdida de información que puede acarrear. Otra situación, es que en algunos centros, se requiere por medida de seguridad que además de una historia clínica electrónica, se cuente con una copia o respaldo en papel.

Por las situaciones descritas anteriormente, se determina que el **problema a resolver** es: ¿Cómo facilitar los procesos de gestión en el área de archivo en las instituciones hospitalarias?

El **objeto de estudio** lo constituye el proceso de gestión de información en las instituciones hospitalarias, enmarcado en el **campo de acción** proceso de gestión de la información en el área de Archivo de las instituciones hospitalarias.

Basado en esa idea se define como **objetivo general** del presente trabajo: Implementar los procesos definidos para el módulo Archivo del sistema de información hospitalaria “alas HIS”, que facilite la gestión de información en esta área de las instituciones hospitalarias.

Para darle cumplimiento al objetivo propuesto, así como controlar y evaluar el proceso investigativo, se proponen como **tareas de la investigación**:

1. Evaluar las tendencias actuales de los sistemas de información hospitalaria que cuenten con el módulo Archivo.
2. Valorar la arquitectura definida por el Departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
3. Aplicar las pautas de desarrollo de los entregables y diseño definidas por el Departamento de Sistemas de Gestión Hospitalaria.
4. Implementar los procesos del módulo Archivo del sistema de información hospitalaria alas HIS.
5. Garantizar la integración del módulo Archivo con el resto de los módulos del sistema de información hospitalaria alas HIS.
6. Obtener el acta de liberación interna de los artefactos generados por el grupo de Calidad.

Entre los beneficios proporcionados por el módulo Archivo:

1. Localizar las historias clínicas de forma rápida y sencilla, tanto dentro como fuera del archivo.
2. Conocer las historias clínicas que han superado el tiempo estimado de devolución, minimizando de esta forma las posibilidades de pérdidas de las mismas.
3. Gestionar exámenes complementarios que llegan al archivo para ser guardados.
4. Gestionar las historias clínicas inactivas y las pertenecientes a pacientes fallecidos.
5. Listar pacientes que poseen tanto HCE como HC física.
6. Gestionar préstamos de historias clínicas solo por personal autorizado.

El presente documento consta de cuatro capítulos, el primero de ellos, **FUNDAMENTACIÓN TEÓRICA**, ubica al lector en el ambiente de desarrollo del módulo Archivo. Presenta un estudio del estado del arte de

los sistemas existentes vinculados al campo de acción, y justifica las tecnologías, metodologías y herramientas que fueron utilizadas para el desarrollo del mismo. Por su parte el capítulo dos: **DESCRIPCIÓN DE LA ARQUITECTURA**, contiene la fundamentación de la arquitectura, la estrategia de integración, la vista de despliegue, los requisitos no funcionales y la especificación de los estándares y estilos de codificación a utilizar.

En el tercer capítulo **DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA** se implementan las clases y subsistemas en términos de componentes. Presentándose una propuesta de solución, en aras lograr una gestión más eficiente de los procesos hospitalarios asociados al área en cuestión. Por último, en el cuarto capítulo: **MODELO DE PRUEBA** se caracteriza el método de prueba a utilizar y se describen los casos de prueba.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Este capítulo contiene la definición de los principales conceptos relacionados con el Módulo Archivo del alas HIS y que son básicos para la comprensión del funcionamiento del mismo, así como el análisis de diferentes sistemas de información existentes a nivel mundial que incluyen algunas de las funcionalidades presentes en él. También se describen las tecnologías, metodologías y herramientas de software definidas por el Departamento de Sistemas de Gestión Hospitalaria con las que se llevará a cabo el proceso de desarrollo.

1.1. Conceptos fundamentales relacionados con el dominio del problema

1.1.1. Tarjetón de reemplazo

Tarjetón que se coloca en lugar de la HC prestada, este consta de la numeración de la historia, la fecha y motivo de la solicitud y el nombre del solicitante. Los tarjetones indicarán el destino de la HC.

1.1.2. Registro por días y médicos

Se emplea para controlar las historias clínicas que son manipuladas por concepto de atenciones en Consulta Externa, en este se señalan aquellas historias solicitadas el mismo día de consulta.

1.1.3. Control de Historias Clínicas

Listado donde se registran los números de historias clínicas en estricto orden consecutivo. Consta del número de la HC, la fecha en que éste se inicia y el nombre(s) y dos apellidos del paciente. Este proceder se hará, utilizando la tarjeta índice de paciente, la cual será archivada en orden alfabético.

1.1.4. Tarjetero índice de paciente, Fichero Maestro de Pacientes o Fichero índice de paciente

Es el conjunto de registros que contiene los datos básicos de identificación de todos los pacientes. Debe ser único, permanente, centralizado e independiente del tipo de asistencia, incluyendo como mínimo los siguientes datos: Identificador o número de historia clínica, nombre y apellidos, sexo, fecha de nacimiento, números de identificación personal, domicilio completo incluido código postal, teléfono, (y en caso de que exista) entidad aseguradora y número de afiliación.

1.1.5. Solicitud de préstamo de historia clínica

Documento que permite solicitar el acceso a una HC. Este debe realizarse por escrito contemplando: número de la historia, nombre del paciente, fecha de solicitud y devolución, persona solicitante y aprobación de la Dirección del Hospital o personas autorizadas.

1.1.6. Tarjetero calendario

Registro en el cual se controla la fecha de devolución de las historias clínicas.

1.1.7. Archivo pasivo

Archivo al que se envían las historias clínicas que causan baja y las que pertenecen a pacientes fallecidos. Estos documentos son ordenados por número consecutivo y por año respectivamente.

1.1.8. Archivo activo

Archivo en el que se encuentran todas las historias clínicas vigentes, es decir las que han sido actualizadas en una fecha menor a 5 años, generalmente.

1.1.9. Historia clínica

Conjunto de documentos clínicos obtenidos como parte de la atención al paciente. Se considera como el único documento válido desde los puntos de vista clínico y legal. Es el eslabón principal en el sistema de información hospitalario, imprescindible en sus vertientes asistencial y administrativa.

1.1.10. Foliar

Proceso mediante el cual se le asigna un número a la HC que representa su ubicación en el archivo.

1.1.11. Complementarios o Informes

Son los resultados provenientes de laboratorios (clínico, microbiológico, neurofisiología, anatomía patológica) y radiología que son archivadas en la HC de cada paciente. Los complementarios se archivan diariamente en cada HC en la misma medida que son recepcionados.

1.1.12. Institución hospitalaria

Establecimiento destinado a la prevención, diagnóstico y tratamiento de pacientes, donde se practican también la investigación y la enseñanza.

1.1.13. HC inactiva

HC que no han tenido ningún movimiento en años, o sea, que no han sido prestadas.

1.2. Descripción específica del módulo

El archivo hospitalario es el departamento dentro de la Institución de Salud responsable de recepcionar, almacenar, dar baja y mantener en buen estado las historias clínicas de los pacientes. Este lleva además el control de los préstamos y devoluciones de historias clínicas, el mantenimiento de los registros, el mantenimiento de la HC única y los indicadores de calidad del archivo basados en la disponibilidad de la misma. El objetivo del Módulo Archivo es proporcionar una solución informática que permita agilizar estos procesos que de forma manual se tornan ineficientes. Logrando así optimizar el rendimiento en el tiempo de trabajo, un mayor control y organización, y minimizar la pérdida o destrucción de datos.

1.3. Sistemas automatizados existentes vinculados al campo de acción

El desarrollo de las tecnologías ha hecho posible la gestión de grandes volúmenes de información de forma rápida y eficiente. En el sector archivístico, estas tecnologías no solo han proporcionado una salva perdurable a la información, sino que permiten controlar el acceso a datos y contribuyen a la disponibilidad de la información, disminuyendo el papeleo y agilizando los procesos. Entre los sistemas vinculados a este campo se encuentran:

1.3.1. GoWin

"Gowin" es un software sanitario desarrollado por "Valen Computer" para gestionar un hospital, clínica, o consulta médica. Posee una estructura modular y es adaptable a las necesidades de cada tipo de centro. "Está desarrollado sobre Windows y sigue sus estándares, presentando una apariencia familiar e intuitiva que facilita la adaptación del usuario a la nueva aplicación, lo que redunda en una reducción de los costes de formación." (4)

"GowinMw es un sistema de la información que ha sido desarrollado para los HIS, tiene el objetivo de recopilar toda la información referente a la HC de los pacientes que pertenecen a la organización.

Principales funciones de la historia clínica:

- Búsqueda de historias
- Antecedentes personales y familiares.

- Diagnósticos previos.
- Factores de riesgo.
- Alergias.
- Tratamientos.
- Exploraciones.
- Histórico por unidades funcionales.
- Histórico de pruebas.

Módulos de la historia clínica- Sistema de la información de la historia electrónica.

- Búsqueda de historias.
- Visualización de expedientes.
- Historia clínica.” (5)

Este sistema ofrece funcionalidades muy útiles para el tratamiento de la HCE, las cuales no satisfacen las necesidades que se derivan de la manipulación y trabajo con la HC física o tradicional.

1.3.2. Sistema de Administración de Hospitales(Hospital Management System)

“Está diseñado para pequeños, medianos y grandes hospitales. Abarca un amplio rango en la administración de hospitales y procesos de gestión. Proporciona los datos de pacientes consolidados y la información pertinente en el hospital para apoyar la toma eficaz de decisiones en el cuidado del paciente, la programación, el quirófano, la facturación, etcétera, en un flujo sin errores.

Este es un sistema integrado para agregar, almacenar, visualizar, editar, consultar y recuperar registros médicos del paciente, el costo del tratamiento y el inventario del hospital, la asignación de cama del paciente, intervención teatral de asignación y el sistema de radiología. Tiene la facilidad de registrar pacientes con una programación de citas de un único ID, manejo de pacientes ambulatorios, gestión de pacientes hospitalizados y facturación. El sistema proporciona la facilidad de generar informes a medida para todas las funciones.

Automatiza el sistema de gestión del Hospital y todas las funciones que en él se realizan. Es fácil de usar, flexible, posee bases de datos seguras, autenticación de usuarios y es capaz de mantener millones registros.

La función de administración de archivos automatiza y mantiene los detalles relativos a las actividades del módulo Gestión de archivos.

Principales características:

- Nivel, Nodo y definición de archivo.
- Ubicación de los archivos de datos.
- Seguir el movimiento de documentos entre los departamentos.
- Aprobación de los documentos entrantes y salientes.
- Informes de los documentos de salida, salida a departamentos, documentos entrantes, espera de documentos, etc.” (6)

Este módulo del Hospital Management System, cuenta con todas las funcionalidades necesarias para el control y gestión de la HC física, no obstante es desarrollado y comercializado por la agencia India, “indianinfotech”, lo cual elimina la posibilidad de su adquisición de forma gratuita.

1.3.3. Hosix V

Toda la información referente a este producto fue obtenida del sitio oficial de la empresa SIVSA (7).

“Hosix-V es un Sistema de Gestión e Información Hospitalaria flexible, integrado y modular, que abarca todas las áreas de actividad de un Hospital y pretende, a través de una utilización fácil, rentabilizar los recursos existentes para organizar el trabajo desarrollado diariamente.” (…) “Es un software diseñado en una arquitectura Cliente – Servidor accesible desde clientes WEB, lo que permite ser ejecutado desde cualquier navegador ya sea vía Intranet o Internet.”

El mismo cuenta con un módulo de Archivo que “permite realizar un mantenimiento global de la documentación en un centro hospitalario, haciendo especial hincapié en la definición de número de historial clínico único por paciente para optimizar la gestión de los historiales clínicos, evitar duplicidades y facilitar los trámites administrativos.” Además “contiene potentes herramientas para la

gestión multiarchivo, multicarpeta y multivolumen que permiten al usuario configurarlas para adaptarse a la estructura del archivo del hospital.” Y “…facilita el control, verificación y la fiabilidad de los datos registrados gracias a su integración con el resto de los módulos de admisión.

Permite el almacenamiento de las carpetas y el control de los numerosos movimientos de éstas gracias a un seguimiento exhaustivo de su ubicación en cada momento.

Orientado y diseñado de cara al personal administrativo, al de codificación del archivo y a los supervisores.”

No caben dudas de que este software definitivamente se ajusta a las necesidades que se persiguen satisfacer, no obstante su adquisición puede resultar costosa debido a que es comercializado por la empresa española: SIVSA.

1.3.4. Galenhos

“El Sistema Integrado de Gestión Hospitalaria Galenhos ha sido diseñado con el propósito de apoyar a los establecimientos de salud en el correcto registro de información, clínica o administrativa, y la generación de información gerencial.” Así se plantea en el sitio El Hospital, del cual también fueron obtenidas las características del mismo que posteriormente se describen. (8)

Ventajas que presenta:

- “Información estandarizada.
- Bases de datos consolidadas y exportables.
- Generación de reportes clínicos de uso gerencial.
- Altos estándares de seguridad informática.
- Diseño modular.

El desarrollo inicial de Galenhos se ha enfocado en el propósito de hacer más eficiente la gestión de seis procesos operativos críticos de un hospital, entre los cuales se encuentran: Consulta externa, Facturación, Hospitalización, Emergencia y Archivo clínico, módulo encargado de recibir solicitudes de historias clínicas, para su búsqueda y monitorear las historias clínicas que se encuentran fuera del archivo.”

Debido a que su desarrollo nace de la asistencia técnica prestada por la Agencia de Estados Unidos para el Desarrollo Internacional (USAID), a través del Proyecto PartnersforHealthReform plus (PHRplus), una vez más se está frente a un software que cuenta con las funcionalidades buscadas pero que igualmente es un software no distribuido de forma gratuita.

1.3.5. Sistema Informativo Hospitalario MEDISYS

Según la información obtenida de CEDISAP (9), “el Sistema Informativo Hospitalario- MEDISYS, permite registrar, controlar y procesar la información necesaria del paciente para la realización de los servicios de salud que requiera, así como la información de la planificación, ejecución y supervisión de los servicios de salud prestados por la institución para facilitar la toma de decisiones con vistas a mejorar la calidad y eficiencia de los mismos.

Este Sistema Informativo Hospitalario funciona a través de una Red Local en plataforma Cliente – Servidor”, y entre los módulos que lo conforman se encuentra Registros Médicos, el cual “permite la inscripción e ingreso de los pacientes, así como el control del flujo de las historias clínicas conservadas en el archivo.”

Las desventajas que presenta este producto se manifiestan desde el punto de vista tecnológico, ya que el mismo no es compatible con el “alas HIS”.

“La plataforma tecnológica utilizada para el desarrollo del Proyecto ha sido:

- Lenguaje de Programación orientado a objeto (Borland Delphi 3,0 Client - Server).
- Sistema de gestión de Base de Datos para plataforma cliente - servidor (SQL Server 6,5).
- Lenguaje para la elaboración de página WEB (HTML).
- Sistema operativo Windows NT 4,0 en el Server y Windows ’95 en los clientes.”

1.3.6. Galen Admisión y Archivo

Ha sido creado por la empresa de origen cubano SOFTEL. Está desarrollado para una plataforma Windows de 32 bits (Windows NT o superior y Windows 98 o superior) con una configuración Cliente/Servidor y el uso del gestor de bases de datos relacional SQL Server como reservorio de la información. Puede emplearse en conjunto con el Galen Hospital.

“El módulo de Archivo abarca el área de archivo de historias clínicas, permitiendo:

- Controlar la lista de pendiente de entrega y pendientes de recepción de historias clínicas.
- Registrar salidas de las historias clínicas de los estantes.
- Registrar entradas de las historias clínicas a los estantes.
- Controlar cambios de números de historias clínicas provisionales por definitivo.
- Localiza historias clínicas por criterios.”(10)

Se ha realizado un análisis de los sistemas de archivos vinculados a la salud a nivel internacional y nacional, muchos de los cuales no brindan la información suficiente sobre las funcionalidades disponibles o sobre las herramientas con las que fueron desarrollados, por lo que se hace difícil decidir si estas soluciones se adecuan o no a las necesidades reales. A esto se le suma que la adquisición de algunos de estos sistemas puede resultar muy costosa. Es por ello que a partir de las experiencias obtenidas se decide implementar el Módulo Archivo para el Sistema de Información Hospitalaria alas HIS. Además, el análisis de estos sistemas ha permitido detectar una serie de funcionalidades básicas que no deben faltar en la implementación.

1.4. Herramientas y tecnologías de desarrollo a tener en cuenta

En todo proceso de desarrollo de software se busca lograr un producto eficaz y eficiente que reúna los requisitos del cliente. Dotar el software de portabilidad, robustez y capacidad para asimilar cambios estructurales con mayor flexibilidad y sin que ocurran cambios severos. La estructura de un software está definida por los patrones arquitectónicos lo cual resulta de gran ayuda para los desarrolladores pues esto facilita el diseño del sistema.

Existen múltiples definiciones de Arquitectura de Software una de ellas es la proporcionada por la IEEE Std 1471-2000, que plantea que:

“La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. (11)

Una vez analizadas las características que se desea posea el software, así como la capacitación y experiencia con que cuenta el equipo de desarrollo, se hace necesario escoger una serie de herramientas y tecnologías para su creación que garanticen un producto costeable y de alta calidad en el menor tiempo posible. A continuación se presenta un resumen de las tecnologías estudiadas, haciendo mayor énfasis en la arquitectura definida por el Departamento de Gestión Hospitalaria para sus aplicaciones, por ser la que se ha decido adoptar al resultar la más adecuada para el desarrollo de este módulo.

1.4.1. Java Enterprise Edition 5 (Java EE 5)

Java Enterprise Edition o Java versión 5 es una plataforma de programación distribuida para ejecutar y desarrollar software de aplicaciones en lenguaje de programación Java, desarrollada por SunMicrosystem. Esta es un conjunto de librerías que establecen un estándar para lograr un producto altamente calificado. Soporta una arquitectura de N niveles basada en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. Permite el manejo de diversos detalles mediante una programación simple y al no ser privativa, el sistema que se desarrolle usando Java, además de permitir el uso en Cuba, puede ser comercializado en el mundo entero. (12)

1.4.2. Seam

Seam es un marco de trabajo desarrollado para Java EE, creado para la realización de aplicaciones Web 2.0 de forma fácil, soportando una arquitectura unificada de componentes. Permite la integración de tecnologías como AJAX, JavaServer Faces (JSF), Enterprise Java Beans (EJB3), java Portlets, Business Process Management (BPM), Drools, Hibernate y Java Persistence API (JPA) en una única solución.

- Elimina en gran medida la complejidad existente desde el nivel de arquitectura hasta el nivel de Application Programming Interface (API).
- Permite el desarrollo de aplicaciones web basadas en Plain Old Java Objects (POJO), componentes de User Interface (UI) y la menor cantidad de XML.
- Se integra con librerías de controles de código abierto basadas en JSF como RichFaces e ICEFaces.
- Añade herramientas de gran utilidad para el desarrollo de aplicaciones Web.

- Está basado en estándares utilizados y probados (escalables, portables y reusables).

Los contextos facilitan el control del ciclo de vida del contenido involucrado en la ejecución de una tarea.

Contextos que soporta:

- Contexto de página.
- Contexto de aplicación.
- Contexto de evento.
- Contexto de sesión.
- Contexto de conversación. (13)

1.4.3. Interfaz de Usuario Seam (Seam UI)

"Serie de controles JSF altamente integrables con JBossSeam. Están dirigidos a complementar los controles JSF incorporados y los controles de otras bibliotecas externas." (14)

La Seam UI brinda a la Interfaz de Usuario JSF (JSF UI) componentes Seam para manejar información en tiempo de ejecución, así como un conjunto de componentes para solucionar fácilmente la representación de los problemas del negocio en la aplicación a desarrollar. Entre las ventajas que brinda la misma se encuentran las validaciones, el manejo de las conversaciones y el manejo de los procesos de negocio.

1.4.4. Facelets

Es un framework para el diseño de páginas Web centrado en la tecnología JSF, por lo cual se integran de manera muy factible, además es muy fácil de usar y configurar. Facelets posee características tales como:

- Trabajo basado en plantillas.
- Fácil composición de componentes.
- Creación de etiquetas lógicas a la medida.
- Funciones para expresiones.

- Desarrollo amigable para el diseñador gráfico.
- Creación de librerías de componentes. (15)

1.4.5. Drools

"Es un motor de reglas de negocio compatible con la especificación Java Specification Request 94(SR-94 Rules Engine API) para el motor de reglas de programación de interfaz de usuario. Siendo un componente que a partir de una información inicial y un conjunto de reglas, detecta qué reglas deben aplicarse en un instante determinado y cuáles son los resultados de esas reglas. Posee la ventaja de que si el negocio de una aplicación cambia, solo es necesario modificar las reglas del negocio, sin tener que realizarse la modificación del código, ni recompilar, ni detener la aplicación si se encuentra en ejecución." (16)

1.4.6. JBoss AS como servidor de aplicaciones

"JBoss Application Server es el servidor de aplicaciones de código abierto. Cuenta con licencia LGPL, por lo que puede ser distribuido o redistribuido y usarse en la implementación de cualquier aplicación comercial sin la implicación de costo alguno. Soporta todas las especificaciones correspondientes, incluyendo servicios adicionales como clusterizar, carga en memoria caché y persistencia, por ser una plataforma con certificación JEE 5. Es ideal como servidor de aplicaciones Java y aplicaciones Web. También soporta Enterprise Java Beans (EJB) 3.0. Cuenta con un conjunto de componentes claves como son: JBoss AS 4.2, Hibernate 3.2.4, Seam 2.0." (17)

Está implementado completamente en Java y puede ser empleado por cualquier sistema operativo que lo soporte.

1.4.7. JBoss Tools

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

Los módulos de JBoss Tools son:

- RichFaces VE: Editor visual proporcionado por Exadel. Brinda el apoyo para la edición visual de páginas HTML, JSF, JSP y Facelets. También incluye soporte visual para las librerías de componentes JSF incluyendo JBossRichFaces.

- Seam Tools: Incluye soporte para seam-gen, RichFaces VE.
- Hibernate Tools: Soporta el mapeo de archivos, anotaciones y JPA con la ingeniería inversa, completamiento de código, asistentes de proyecto, refactorización, ejecución interactiva de HQL/JPA-QL/Criteria.
- JBossASTools: Fácil de iniciar, detener y analizar ejecución paso a paso al estar integrado con Eclipse. También incluye funciones para el despliegue eficaz de cualquier tipo de proyecto en el IDE.
- Drools IDE: Editor de ficheros de reglas, análisis de ejecución paso a paso e inspección de reglas.
- JBPM (Java Business Process Management) Tools: Edición del flujo de trabajo del JBPM, motor de procesos BPM.el JBPM, motor de procesos BPM.
- JBossWS (JBoss Web Services) Tools: Desarrollo, invocación, inspección y pruebas de servicios web sobre http con la adición y soporte de características JBossWS.” (18)

1.4.8. Java Server Faces(JSF)

“JSF es una librería de interfaz de usuario para aplicaciones Web implementadas con Java. Diseñado para aliviar la carga de desarrollo y mantenimiento de aplicaciones que se ejecutan en Servidores de aplicaciones Java y prestar sus interfaces de usuario a un cliente objetivo, JSF aprovecha las interfaces de usuario ya existentes, estándares y conceptos de Web. JSF usa Facelets como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.” (19)

1.4.9. Rich faces 3.2 como librería de componentes JSF

“Rich Faces es una librería de código abierto que añade los componentes de Ajax (Ajax4jsf) en aplicaciones existentes de JSF sin recurrir a JavaScript. Rich aprovecha la librería de JavaServer Faces para incluir validaciones, instalaciones de conversión y la gestión de los recursos estáticos y dinámicos. De esta forma permite a los desarrolladores ahorrar tiempo y aprovechar las características de los componentes para crear aplicaciones Web, con mejor apariencia visual.” (20)

1.4.10. Ajax4jsf

“Ajax4jsf es una librería de código abierto que añade la capacidad de AJAX en las aplicaciones existentes de JSF sin recurrir a código JavaScript. Ajax4jsf se integra totalmente en la implementación JSF como librería que permite: la validación, las instalaciones de conversión y gestión de los recursos estáticos y dinámicos. Mediante esta librería se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargar por completo, realizar peticiones automáticas al servidor, control de cualquier evento de usuario, etc. Ajax4jsf es función de soporte de AJAX y altamente personalizable (look-and-feel) que pueden ser fácilmente incorporados en aplicaciones JSF.” (21)

Ajax4jsf se constituye de los siguientes módulos o componentes:

- Ajax Filter: hay que registrar un filtro de ajax4jsf en el web.xml el cual reconoce varios tipos de peticiones.
- Ajax Action components: hay tres componentes: AjaxCommandButton, AjaxCommandLink y AjaxSupport. Estos componentes se usan para mandar las peticiones Ajax desde el cliente.
- Ajax Containers: es un interfaz que describe una zona dentro de la página JSP que debería ser decodificada durante la petición Ajax.
- JavaScript Engine: se ejecuta en el lado del cliente y sabe cómo actualizar las diferentes zonas de la página JSP, basándose para ello en la respuesta Ajax. (21)

1.4.11. Postgres 8.3 como servidor de base de datos

“PosgtreSQL es un sistema gestor de bases de datos objeto-relacional basado en Postgres, versión 4.2, desarrollado en la Universidad de California en Berkeley ComputerScienceDepartment.” (22)

Pertenece al ámbito del software libre y está bajo la licencia BSD (Berkeley Software Distribution), la cual tiene menos restricciones en comparación con otras como la GPL (de la Fundación del Software Libre). Permite además la libre redistribución y modificación y provee al usuario de libertad ilimitada con respecto al software, puede decidir incluso redistribuirlo como no libre. Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac OS X, Solaris, BSD, Tru64, Unix y otros más.

“Este soporta una gran parte del estándar SQL y ofrece:

- Consultas complejas.
- Claves foráneas.
- Desencadenantes o disparadores.
- Vistas.
- La integridad transaccional.
- Control de concurrencia multiversión.” (22)

PostgreSQL posee un tamaño máximo de base de datos y de índices por tabla ilimitado. Además puede ser ampliado por el usuario, por ejemplo, este puede adicionar nuevos tipos de datos, funciones, operadores, funciones de indexado, métodos de índice y lenguajes procedurales. (22)

1.4.12. Framework Hibernate para el acceso a los datos

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. Tiene soporte para más de 30 dialectos diferentes (controladores de bases de datos diferentes).

Proporciona un lenguaje de consulta rica para acceder a objetos, proporcionar almacenamiento en caché y apoyo JMX. Está destinado a suministrar la persistencia de alto rendimiento con bajos recursos. (23)

1.4.13. Java Persistence API (JPA)

“Java Persistence API (JPA) proporciona un modelo de persistencia basado en POJO's para mapear bases de datos relacionales en Java. El Java Persistence API fue desarrollado por el grupo de expertos de EJB 3.0 como parte de JSR 220, aunque su uso no se limita a los componentes software EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones clientes; incluso fuera de la plataforma Java EE, por ejemplo, en aplicaciones Java SE.

En su definición, se han combinado ideas y conceptos de las principales librerías de persistencia como Hibernate, Toplink y JDO, y de las versiones anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA.” (24)

El mapeo objeto/relacional, o sea, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML; también pueden definirse transacciones como anotaciones JPA.

Esta consta de tres áreas:

- El Java Persistence API.
- El lenguaje de consulta.
- El mapeo de los metadatos objeto/relacional.

1.4.14. Enterprise JavaBeans (EJB3)

La especificación de Enterprise JavaBeans, define una arquitectura para un sistema transaccional de objetos distribuidos basados en componentes. La especificación posee un modelo de programación; convenciones o protocolos y un conjunto de clases e interfaces que crean el API EJB. Proporciona a los desarrolladores de Beans y a los vendedores de servidores EJB, un conjunto de contratos que definen una plataforma de desarrollo común. El objetivo de estos contratos es asegurar la portabilidad a través de los vendedores y el soporte de un rico conjunto de funcionalidades. (25)

Además de las tecnologías antes expuestas fueron analizadas otras que decidimos no utilizar por ser más factible el uso de las definidas por el departamento de gestión hospitalaria. Entre estas se encuentra:

1.4.15. Spring

Spring es un marco de trabajo de Aplicación para aplicaciones escritas en Java. Es además, un “contenedor liviano basado en la técnica Inversión de Control (IoC) y una implementación de desarrollo según el paradigma de Orientación a Aspectos (AOP).” (26) Es de código abierto, cuenta con una arquitectura dividida en siete capas o módulos, e integra diferentes tecnologías existentes.

Permite integración con los ORM (Object-Relational Mapping) más populares del mercado, entre ellos: Hibernate, iBATIS SQL Mapas, Apache OIB, JDO y Oracle TopLink, además permite la integración con otros frameworks como Struts, JSF y Tapestry.

Define la forma de desarrollar aplicaciones J2EE, dando soporte y simplificando complejidad propia del software corporativo. Promueve el bajo acoplamiento a partir de la inyección de dependencias entre objetos (relacionales). Y presenta una estructura simplificada para el desarrollo y utilización de aspectos. (26)

1.6. Lenguajes

1.6.1. Java como lenguaje de programación

Java es un lenguaje de programación orientado a objetos que está inspirado en la sintaxis de C++, pero su funcionamiento tiene mayor semejanza al de Smalltalk que a este. Posee un modelo de objetos más simple que C++ y elimina herramientas de bajo nivel. Debido a esto ocurren menos errores relacionados con la manipulación directa de punteros o memoria, por lo que se conoce como un lenguaje potente.

La característica de Java de ser independiente de la plataforma despliega una multitud de posibilidades para los usuarios, pues permite que prácticamente cualquier aplicación (ya sean juegos, herramientas o programas de información y servicios) se ejecute en casi cualquier equipo o dispositivo.

Java es un lenguaje compilado e interpretado. Su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador los cuales se pueden ejecutar sobre cualquier máquina que contenga el intérprete y el sistema de ejecución en tiempo real (run-time). Estos bytecodes constituyen un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. Esta indiferencia a la arquitectura influye en la portabilidad de Java quien además especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

Este lenguaje facilita la creación de aplicaciones distribuidas proporcionando una colección de clases para ser usadas en aplicaciones de red. Por ejemplo: brinda la posibilidad de abrir sockets, así como establecer y aceptar conexiones con servidores o clientes remotos.

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. Para eso se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real. Su objetivo, es que no se pueda ejecutar en el ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas.

Para evitar las limitaciones de las aplicaciones que sólo pueden ejecutar una acción a la vez Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario, un tercero presenta una animación en pantalla y el cuarto realiza cálculos. Esto, unido a que el lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado hace de Java además un lenguaje de alto rendimiento. (27)

1.6.2. XHTML

Los documentos XHTML están basados en XML, una familia de módulos y tipos de documentos que reproduce, engloba y extiende el HTML 4.0. Los tipos de documentos de esta familia están diseñados fundamentalmente, para trabajar en conjunto con aplicaciones de usuario basados en XML. El XHTML trae consigo varias ventajas, entre las que se encuentran:

- Son conformes a XML. Por lo que son fácilmente visualizados, editados y validados con herramientas XML estándar.
- Pueden escribirse para que funcionen igual, o mejor que lo hacían antes, tanto en las aplicaciones de usuario conformadas con HTML 4.0, así como en las nuevas aplicaciones conformes a XHTML.
- Pueden usar aplicaciones que se basen, ya sea en el Modelo del Objeto Documento de HTML o XML. (28)

1.7. Sistemas distribuidos

1.7.1. Modelo Cliente- Servidor

Básicamente este esquema consiste en un programa cliente que hace peticiones a otro programa (el servidor) que le brinda una respuesta. Esta idea es aplicable a programas que se ejecutan en una misma computadora, aunque resulta más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En este modelo la capacidad de proceso está repartida entre los clientes y los servidores, aunque el mayor peso recae en estos últimos, los cuales deben estar establecidos en máquinas más potentes. Este arquetipo facilita la separación de responsabilidades y la centralización de la gestión de la información, lo cual clarifica el diseño del sistema.

La separación entre el cliente y el servidor es de tipo lógico. El servidor puede ejecutarse en más de una máquina y puede estar constituido por más de un programa. Los servidores pueden ser de diferentes tipos, por ejemplo, pueden ser servidores de archivos, servidores de correo o servidores web; cada uno con propósitos diferentes, pero con la misma arquitectura básica. Si el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras estaríamos en presencia de un sistema multicapas, los cuales aumentan el grado de distribución.

1.7.2. Estilo arquitectónico

“El estilo arquitectónico o variante arquitectónica define a una familia de sistemas informáticos en términos de su organización estructural. Describe componentes y las relaciones entre ellos con las restricciones de su aplicación, la composición asociada y el diseño para su construcción.” (29)

1.8. Modelo Vista-Controlador

“La arquitectura MVC (Model/View/Controller) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto permite que cualquier cambio producido en el Modelo, se refleje automáticamente en cada una de las Vistas.” (30)

Definición por partes:

- Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y a instrucciones de cambiar el estado (habitualmente desde el controlador). De él dependen la vista y el controlador, pero él no depende de nadie.
- Vista: Se encarga de la visualización de la información. Con el cual interactúa a través de una referencia al modelo.
- Controlador: Controla el flujo entre la vista y el modelo (los datos). Hacia el cual contiene una referencia.

Ventajas que presenta:

- Clara separación entre los componentes de un programa ya que permite implementarlos por separado.
- Adaptación de cambios: al modelo no depender de las vistas, agregar nuevas opciones de presentación generalmente no afecta el proceso.
- Cuenta con un API bien definido; quien use el API podrá reemplazar el Modelo, la Vista o el Controlador sin dificultad aparente.
- La conexión entre el Modelo y sus Vistas es dinámica; no se produce en tiempo de compilación sino de ejecución.

1.9. Metodologías de desarrollo

1.9.1. Proceso Unificado Racional (RUP por sus siglas en inglés)

RUP es un proceso de desarrollo de software que utiliza el lenguaje unificado de modelado UML, y en conjunto estos constituyen una metodología estándar muy utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es adaptable al contexto y necesidades de cada organización, permite estructurar todos los procesos y medir la eficiencia de la misma.

Constituye una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo). Contribuye a las buenas prácticas de Ingeniería de Software. Es dirigido por casos de uso,

centrado en la arquitectura y posee un desarrollo iterativo e incremental. Está preparado para desarrollar grandes y complejos proyectos, permite la administración de requisitos, el control de cambios y la revisión de la calidad y el modelado visual del software.

En este se definen nueve flujos de trabajo, seis de ingeniería y tres de apoyo. Un flujo de trabajo es una secuencia de actividades que se realizan para producir determinados artefactos que aumentarán el desarrollo del proyecto.

1.9.2. Lenguaje Unificado de Modelado (UML por sus siglas en inglés)

UML es un lenguaje de representación visual que permite combinar diversos elementos gráficos y crear diagramas. Se usa para modelar sistemas y usa tecnología orientada a objetos. El lenguaje unificado de modelado describe lo que hará un sistema pero no dice como imprentarlo. Su objetivo es visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo. Involucra todo el ciclo de vida del proyecto y está pensado para varios lenguajes y plataformas.

1.10. Herramientas de desarrollo

1.10.1. Eclipse como herramienta de desarrollo

Eclipse es un entorno integrado de desarrollo (IDE, por sus siglas en inglés) de código abierto, portable y multiplataforma. Fue creado inicialmente por IBM y en la actualidad es mantenido por la Fundación Eclipse, la cual lo define como “una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular”.

Tiene una arquitectura basada en plug-ins y posibilita integrar diversos lenguajes de programación, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc. Se puede integrar con Visual Paradigm usando el Entorno de Desarrollo Inteligente (Smart DevelopmentEnvironment), provocando un entendimiento superior entre analistas y desarrolladores. Eclipse permite el trabajo en equipo mediante el empleo de SubVersion y del Sistema de Control de Versiones (CVS), los cuales constituyen una combinación de vistas y editores que muestran los diversos aspectos de los recursos del proyecto organizados por el rol o la tarea del desarrollador.

Para hacer uso de las versiones de Eclipse es necesario tener instalado en el sistema una máquina virtual Java (JVM), preferiblemente JRE (Java RuntimeEnvironment) o JDK (Java Developer Kit) de Sun.

Eclipse se distribuye bajo licencia EPL (Eclipse PublicLicense). Esta licencia es considerada como libre por la FSF y por la OSI (Open SourceInitiative). La licencia EPL permite usar, modificar, copiar y distribuir nuevas versiones del producto licenciado. El antecesor de EPL es CPL (CommonPublicLicense) escrita por IBM. (31)

También se analizó la posibilidad de utilizar NetBeans como entorno de desarrollo en lugar de Eclipse, pero se decidió utilizar este último porque el resto de las herramientas que se utilizarán para el desarrollo del presente trabajo son creadas por JBoss, una empresa que pertenece a RedHat, y todas son desarrolladas con Eclipse.

1.10.2. NetBeans

"NetBeans IDE forma parte del proyecto de código abierto NetBeans, fundado por Sun MicroSystems, siendo un entorno de desarrollo escrito en Java - pero que puede servir para cualquier otro lenguaje de programación. Existe un número importante de módulos para extender el NetBeans IDE. Este es un producto libre y gratuito sin restricciones de uso y el código fuente está disponible para su reutilización de acuerdo con la CommonDevelopment y DistributionLicense (CDDL) v1.0 y la GNU General PublicLicense (GPL) v2." (32)

"Provee soporte total para Java EE 6. Desarrolla componentes incluyendo páginas web, servlets, servicios web, Enterprise Java Beans (EJB 3.1), y usa el nuevo soporte para contextos e inyecciones de dependencia JSR 299.

Facilita la creación de proyectos Java EE basados en JavaServer Faces 2.0 (Facelets), Spring, Struts, e Hibernate frameworks. El editor admite código de terminación, la navegación y la refactorización para el mapeo de archivos. Además permite: completamiento de código, documentación emergente, y la comprobación de errores para JSP, JSF, XML, JavaScript, CSS y PHP-incluso mixtos." (33)

1.10.3. PgAdmin como aplicación cliente para manejar la base de datos

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

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad.” (34)

1.10.4. Visual Paradigm como herramienta de modelado

“Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML contribuye a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Este trae incluidos los objetos más recientes de UML, diagramas de casos de uso, diagramas de componentes, diagramas de clase, reversa instantánea para Java, C++, XML y XML Schema, brinda soporte para Rational Rose, integración con Microsoft Visio, y posibilita generar reportes y documentación en HTML/PDF. Se integra con IDE’s como NetBeans (de Sun Microsystems), JDeveloper (de Oracle), Eclipse (de IBM), JBuilder (de Borland).

Su diseño está centrado en casos de uso y se usa lo mismo para UML que para BPMN (Business ProcessModelingNotation).” (35)

1.10.5. JRE

Java RuntimeEnvironment (JRE) es un conjunto de herramientas provisto por Sun Microsystems para la ejecución de código, es el entorno en tiempo de ejecución Java. Está compuesto por la máquina virtual de Java (JVM), permitiendo así la interpretación del código sin necesidad de tener en cuenta el hardware del ordenador. Esta tecnología posibilita disponer de disímiles utilidades, como son juegos, aplicaciones para móviles y herramientas de sistema, además incluye plug-ins para los principales navegadores (Internet Explorer, Firefox, Opera, Zafari, Netscape).

Se han mostrado algunas de las características de las tecnologías y herramientas que son utilizadas para el desarrollo del sistema propuesto por la presente investigación, logrando evidenciar la robustez, potencialidad y portabilidad que brinda el software libre en el desarrollo de proyectos a gran escala.

En este capítulo se han evaluado las tendencias actuales de los sistemas de información hospitalaria que cuentan con un módulo Archivo y además se han valorado las posibles tecnologías y herramientas a utilizar, determinándose adoptar la arquitectura definida por el departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones, por ser la más idónea y posibilitar el desarrollo de un sistema robusto y flexible.

CAPÍTULO 2. DESCRIPCIÓN DE LA ARQUITECTURA

En el presente capítulo se definen los requerimientos no funcionales, realizando una descripción detallada de los mismos. Se describe la arquitectura definida y se presenta un análisis de las posibles implementaciones y componentes o módulos ya existentes que puedan ser rehusados.

2.1. Requisitos no funcionales

Los requerimientos no funcionales son aquellos “que 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. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interfaz del sistema.” (36)

2.1.1. 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

La estructura concebida para la organización de la información agiliza el entendimiento del sistema por parte del usuario. Los usuarios serán capaces de alcanzar sus objetivos con un mínimo esfuerzo y obteniendo los resultados máximos.

2.1.2. Fiabilidad

Se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos. Se garantizarán políticas de respaldo a toda la información, evitando pérdidas en caso de desastres ajenos al sistema. Además ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista. La aplicación 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.

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

Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando sólo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión. Además de un registro de todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.

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

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

2.1.3. Eficiencia

El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos. El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

2.1.4. Rendimiento

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

2.1.5. Soporte

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP además de la administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación.

2.1.6. Respaldo y recuperación de base de datos

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

2.1.7. Auditoría

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

2.1.8. Configuración de parámetros

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

2.1.9. Réplica

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

2.1.10. 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.

2.1.11. Requerimientos de hardware

2.1.11.1. Estaciones de trabajo

En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS. Dichas estaciones necesitan capacidad de hardware que soporte un sistema operativo el cual cuente con un navegador actualizado que siga los estándares web. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz con sistema operativo Linux.

2.1.11.2. Servidores

La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.

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

2.1.12. Requerimientos de software

La aplicación debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss AS y PostgreSQL). Deberá disponer de un navegador web, estos pueden ser IE 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

2.1.13. Restricciones de diseño

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

2.1.14. 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.

2.1.15. Interfaz

- Interfaces de usuario

Las ventanas del sistema contendrán los datos claros y bien estructurados. Además permitirán la interpretación correcta de la información. La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario. Todos los textos y mensajes en pantalla aparecerán en idioma español.

- Interfaces software

  Se interactuará con el sistema alas RIS para realizar solicitudes y obtener resultados de estudios radiológicos e imagenológicos.

2.1.16. Portabilidad

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

2.2. Descripción de la arquitectura

La arquitectura de software es un conjunto de patrones que sirven de guía para la elaboración de un sistema. Esta ofrece la descripción del software en subsistemas y componentes, dejando bien claro la forma en que estos interactúan. Su objetivo principal es aportar elementos que ayuden a la toma de decisiones y al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Es, además, una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema.

La arquitectura empleada para el desarrollo del software está basada en el patrón Modelo Vista Controlador, uno de los más usados a la hora de crear aplicaciones web y cuyas características fueron abordadas en el capítulo anterior.

Siguiendo las características de este patrón, se agruparon dichos componentes en tres capas fundamentales: presentación, negocio y acceso a datos. Estructura que garantiza que el cambio o la sustitución de un elemento correspondiente a una de las capas, no altere el funcionamiento de otro que lo utilice y sea parte de alguna de las capas restantes.

La capa de presentación está formada por páginas XHTML, las cuales están compuestas por formularios y controles JSF y RichFaces, estos se encargan de validar y mostrar los datos haciendo uso del componente Ajax4jsf. Incluye establecimiento de reglas de navegación declarativas, la

internacionalización y accesibilidad de la interfaz de usuario, un modelo orientado a eventos y combinado con Facelets, además de que nos da la capacidad añadida de la tecnología de plantillas de Facelets. Por su parte los controles para UI de Seam adicionan varias mejoras a JSF.

La capa de negocio está compuesta por clases controladoras, que definen la lógica del negocio conjuntamente con los datos que se validan en la capa de presentación. Todo esto se desarrolla mediante el frameworkSeam.

La capa de acceso a datos realiza el manejo de los datos mediante Hibernate, lo cual permite seleccionar, modificar, eliminar y persistir la información en la base de datos. Esto permite llevar las consultas a un lenguaje orientado a objeto. Además se pueden establecer validaciones mediante los HibernateValidators.

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

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

La reutilización se apoya en el uso de componentes de software, vistos como colecciones de código reutilizables que facilitan el desarrollo de las aplicaciones, los cuales se pueden encontrar en el mismo sistema que se está desarrollando. Esta reduce los costes de diseño, codificación y comprobación.

El Sistema de Información Hospitalaria alas HIS, el cual se encuentra en desarrollo, está compuesto por diferentes módulos que comparten componentes entre sí. Para la implementación de los procesos enmarcados en el campo de acción de la presente investigación, se identificaron los componentes a reutilizar a partir de un estudio de las funcionalidades correspondientes a los módulos relacionados con la HCE y la atención a pacientes a partir del uso de las mismas.

Con el objetivo de asociar a cada HC física existente en el archivo, su correspondiente HCE se reutilizó del módulo Admisión la funcionalidad buscar HCE. Contribuyendo a que el proceso se realice con mayor rapidez, o sea, economizando tiempo y además minimizando la redundancia.

También, hay algunos componentes que se definen de manera general para ser utilizados por todos los módulos, los cuales brindan un conjunto de facilidades y simplifican el trabajo de los desarrolladores.

Entre ellos se encuentran: la clase Bitácora para el control de las trazas de las acciones que se realizan en la aplicación; la clase User para saber qué usuario está trabajando con la aplicación en tiempo real; y la clase ActiveModule para conocer qué módulo está activo y en qué entidad.

2.4. Seguridad

Las bases de un sistema seguro están dadas en la confidencialidad, integridad y disponibilidad de su información. En aras de lograr un sistema que cumpla con las características anteriores se propone:

- Llevar un control de acceso a nivel de usuarios y contraseñas.
- Establecer una diferenciación entre los usuarios atendiendo al rol que desempeñan, asegurando así, que cada usuario puede acceder exclusivamente a los lugares que su rol le permite.
- Las contraseñas de los usuarios solo podrán ser cambiadas por los propios usuarios cada cierto tiempo, o por los administradores del sistema.
- Se llevará una bitácora de sucesos, registrando las trazas de todas las operaciones realizadas por cada uno de los usuarios para ser revisadas por el administrador del sistema en caso de ser necesario.

2.5. Vista de Despliegue

UML proporciona diferentes vistas para visualizar, especificar, construir y documentar la arquitectura del software. Este lenguaje permite representar cada vista del software mediante un conjunto de diagramas. Por ejemplo, la vista de despliegue o implantación cuenta con el diagrama de despliegue.

El diagrama de despliegue posee como objetivo mostrar las relaciones físicas entre nodos del sistema final, hardware y software. Se le considera un grafo de nodos unidos que logran su comunicación a través de conexiones; donde un nodo es un objeto físico con capacidad operacional. Generalmente en tiempo de ejecución los nodos se consideran recursos computacionales, con capacidad de procesamiento y memoria.

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.

A continuación se muestra el Diagrama de Despliegue correspondiente al módulo Archivo.

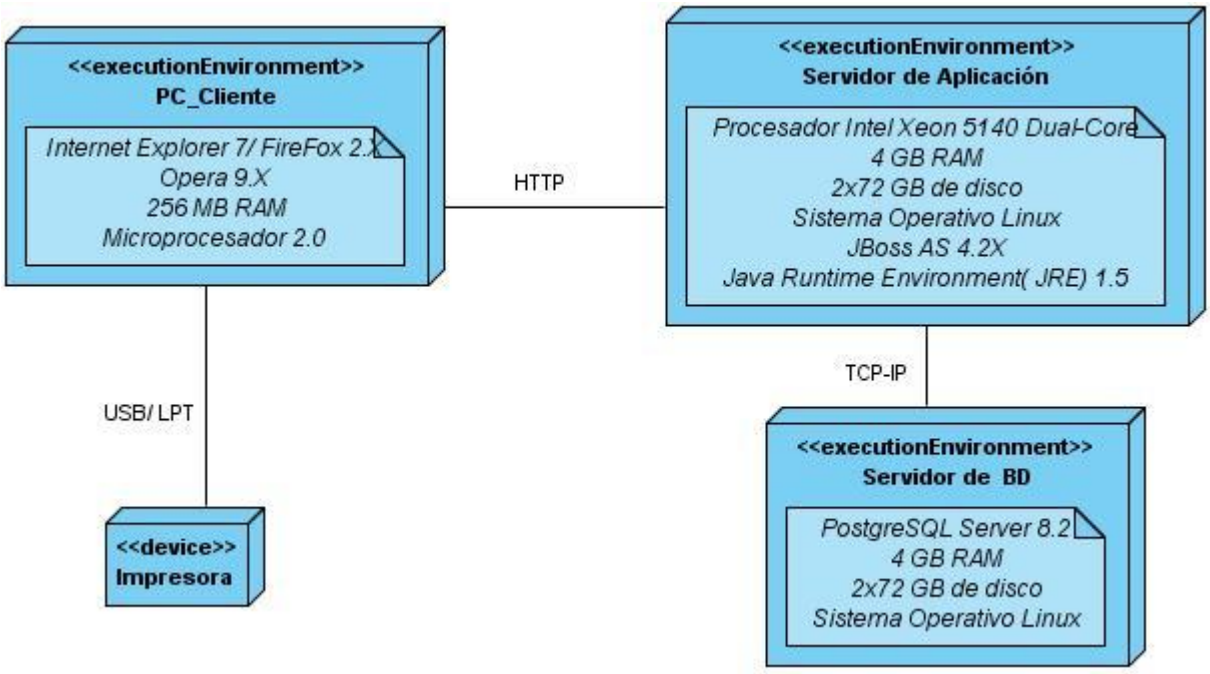


Figura 2.1 Diagrama de despliegue

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

La forma más efectiva de lograr el entendimiento del código implementado por más de una persona es la aplicación de un estándar de codificación. Este comprende todos los aspectos de la generación de código y ayuda a evitar confusiones en el momento de su revisión o a la hora de realizarle modificaciones, al agregarle más funcionalidades o características al sistema.

En el desarrollo del sistema presentado, se utiliza un estándar de codificación, garantizando la homogeneidad del estilo de programación durante la implementación del sistema. Se incluyen en el estándar el uso de las notaciones CamellCasing, para las variables y métodos con nombres compuestos por múltiples palabras juntas, el cual sugiere iniciar cada palabra con letra mayúscula excepto la primera que debe iniciar con minúscula, y PascalCasing para las clases con nombres compuestos por múltiples

palabras juntas, el cual sugiere iniciar cada palabra con letra mayúscula. En cuanto al idioma todas las palabras se deben usar en idioma español, pero sin acentuarse.

A continuación se muestran algunas restricciones de la nomenclatura del código haciendo uso de los estándares de codificación mencionados anteriormente:

Variables y constantes		
Apariencia de constantes	Todas sus letras en mayúscula.	Se deben declarar las constantes con todas sus letras en mayúscula.
Apariencia de variables	Primera letra en minúscula y si es un nombre compuesto usa notación CamellCasing.	El nombre de las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará la notación CamellCasing.
Aspectosgenerales	Nombres de las variables y constantes.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Tabla 2.1 Estándares de codificación - Variables y constantes.

Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento es el objetivo de la *identación*, para esta se definieron los siguientes aspectos:

Identación	
Inicio y fin de bloque	Se deben 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.
Aspectosgenerales	El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la computadora o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción.

	Nunca colocar ({) en la línea de un código cualquiera, esto requiere una línea propia.

Tabla 2.2 Estándares de codificación – Identación.

Los comentarios, separadores, líneas, espacios en blanco y márgenes fueron otros de los elementos a considerar en el estándar, con estos se logra establecer un modo común para comentar el código de forma tal que se comprenda con sólo leerlo una vez.

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 y al final de cada bloque de código, especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro).
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 un método.
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	No comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción
	Sobre los espacios en blanco	No se debe usar espacio en blanco: Después del corchete abierto y antes del cerrado de un arreglo. Después del paréntesis abierto y antes del cerrado. Antes de un punto y coma.

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

Para el uso de clases y objetos se definieron un conjunto de elementos que garantizan nombrar las clases e instancias de la misma forma para toda la aplicación:

Clases y Objetos		
Apariencia de clases y objetos	Primera letra en mayúscula.	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Ejemplo: MiClase(). Para el caso de las instancias se comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula.	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará la notación CamellCasing.
Apariencia de las funciones	Con verbos que denoten la acción que realiza.	El nombre de las funciones con verbos que denoten la acción que realiza y se empleará la notación PascalCasing. Las funcionen que obtienen algún datoemplean el prefijo get y si fijan algún valor se emplea el prefijo set.
Declaración de parámetros en funciones	Agrupados por tipos Poner los string 1 numéricos 2, además, agrupar según valores por defecto.	Los parámetros que se le pasan a las funciones se declaran agrupados por tipos, defiendo primero los string y luego los numéricos, además se agrupan tenido en cuenta los valores por defecto.
Aspectosgenerales.	Sobre las clases, los objetos, los atributos y las funciones.	El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

Tabla 2.4 Estándares de codificación - Clases y Objetos.

La base de datos, tablas, esquemas y campos fueron otros de los elementos que se tuvieron en cuenta en la definición del estándar.

Bases de datos, tablas, esquemas y campos		
Apariencia de la base de datos.	Las 2 primeras letras representan el tipo.	El nombre de la Base de Datos deben comenzar con el prefijo bd a continuación un underscoard y luego el nombre completamente en minúscula. Los nombres serán cortos y descriptivos.
Apariencia de las vistas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula	El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.
Apariencia de las tablas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizara underscoard para separarlo.
Tablas que representen Relaciones	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula.
Tablas que representen nomencladores.	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre debe ser corto y descriptivo, todo en minúscula.
Apariencia de los procedimientos almacenados.	Las 2 primeras letras representan el tipo. Todas las letras en	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

	minúscula.	se utilizara underscoard para separarlo.
Apariencia de los campos	Todas las letras en minúscula.	El nombre de los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.
Nombre de los campos	En caso de identificadores.	Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo.
Sentencias SQL	Todas las letras en mayúscula.	Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.
Aspectos generales	Sobre las BD, vistas, tablas atributos y procedimientos.	El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con sólo leerlos se conozca el propósito de los mismos.

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

Además, se tuvo en cuenta aspectos relacionados con los controles, los cuales se muestran a continuación:

Controles		
Apariencia de los controles.	Los controles tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a los controles debe comenzar con las primeras letras en minúscula, las cuales identificaran el tipo de datos al que se refiere. Para los nombrescompuestos se empleará notación CamellCasing.

Tabla 2.6 Estándares de codificación – Controles.

Otros aspectos a tener en cuenta fueron:

Tipo Datos	Prefijo	Ejemplo
Int	i	iCantPacientes

double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	e	eSexo
sbyte	sb	sbEdadPaciente
short	sh	shVariableShort
uint	ui	uiVariableUint
long	l	lVariableLong
Objetos	o	oPacienteHistorico

Tabla 2.7 Estándares de codificación - Tipo Datos.

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

Tabla 2.8 Estándares de codificación – Control.

En el presente capítulo ha quedado definida la arquitectura del sistema, siendo analizados los requisitos no funcionales del mismo, así como la seguridad que este debe poseer, enfocados en garantizar un producto confiable y de alta calidad. Se han analizado las posibles implementaciones de componentes o módulos que pueden ser reutilizados, se ha presentado una propuesta de la distribución física del sistema y la estrategia de codificación a emplear en el desarrollo del mismo.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se describe y analiza la solución propuesta, mostrando una descripción de las clases definidas. Se presentan el modelo de datos y el diagrama de componentes, los cuales evidencian las dependencias entre las partes del código del sistema propuesto. Además, se realiza una breve valoración de las técnicas de validación.

3.1. Valoración crítica del diseño

Una vez concluido el análisis, el diseño permite que aplicando un conjunto de principios, conceptos y prácticas se formule un producto o sistema. Creándose un modelo de software y proporcionando detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes de software que son necesarios para implementar el sistema, sin ambigüedades. Su meta es crear un modelo de software que implemente todos los requisitos del cliente de manera correcta y complazca a aquellos que lo usen.

Generalmente, para elaborar el diseño se utiliza un grupo de patrones o modelos. Estos se definen como soluciones elegantes y simples a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que han probado ser exitosos en el pasado. Se adaptan a varias circunstancias, permiten un ahorro de tiempo considerable, y posibilitan la creación de un sistema más eficiente y dinámico.

Para elaborar el diseño de la presente solución se emplearon varios patrones, entre los que destacan los GRASP. Estos describen los principios fundamentales de diseño de objetos para la asignación de responsabilidadesteniendo en cuento a la información que contienen o su comportamiento, la creación de instancias de otras clases en correspondencia con la responsabilidad dada, la colaboración entre los componentes del diseño y la reutilización de los mismos, poniéndose de manifiesto los patrones Experto, Creador, Bajo Acoplamiento y Alta Cohesión. Además, la creación de clases controladoras permitió asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas, facilitando la centralización de actividades.

Otros patrones utilizados son los de diseño a objetos, como el Abstract Factory que permite el acceso a base de datos empleando EntityManager. Este constituye la interfaz principal de JPA y es responsable de llevar a cabo las operaciones de los CRUD.

Gracias a que Hibernate utiliza el patrón ActiveRecord, una fila de una tabla de la base de datos puede convertirse en un objeto y asociarse con objetos del lenguaje de programación usado. Hibernate también

utiliza patrones de mapeo, entre los que se encuentran el Foreign Key Mapping, AssociationTableMapping y el Identity Field. Estos permiten mapear relaciones de uno a mucho, de mucho a mucho y generar un único ID que identifique una entidad, respectivamente.

Lazy Load e Identity map son patrones de comportamiento que evitan que existan en memoria dos instancias distintas del mismo objeto en una transacción de negocio, y resuelven problemas de dependencias circulares y carga desmedida.

Por su parte el Query object interpreta la estructura de objetos y los traduce a consultas SQL (Structured Query Language). Así, los desarrolladores pueden realizar consultas trabajando con las clases y sus atributos en lugar de referirse a las tablas y columnas.

En el presente diseño se han tenido en cuenta también los diagramas de clases de diseño y los diagramas de interacción. Los primeros con el objetivo de visualizar las relaciones entre las clases implicadas en el sistema. Y los segundos que consisten en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos, y se utilizan para modelar aspectos dinámicos. Los diagramas de interacción se clasifican en dos tipos, de colaboración y secuencia, estos últimos destacan el orden temporal de los mensajes.

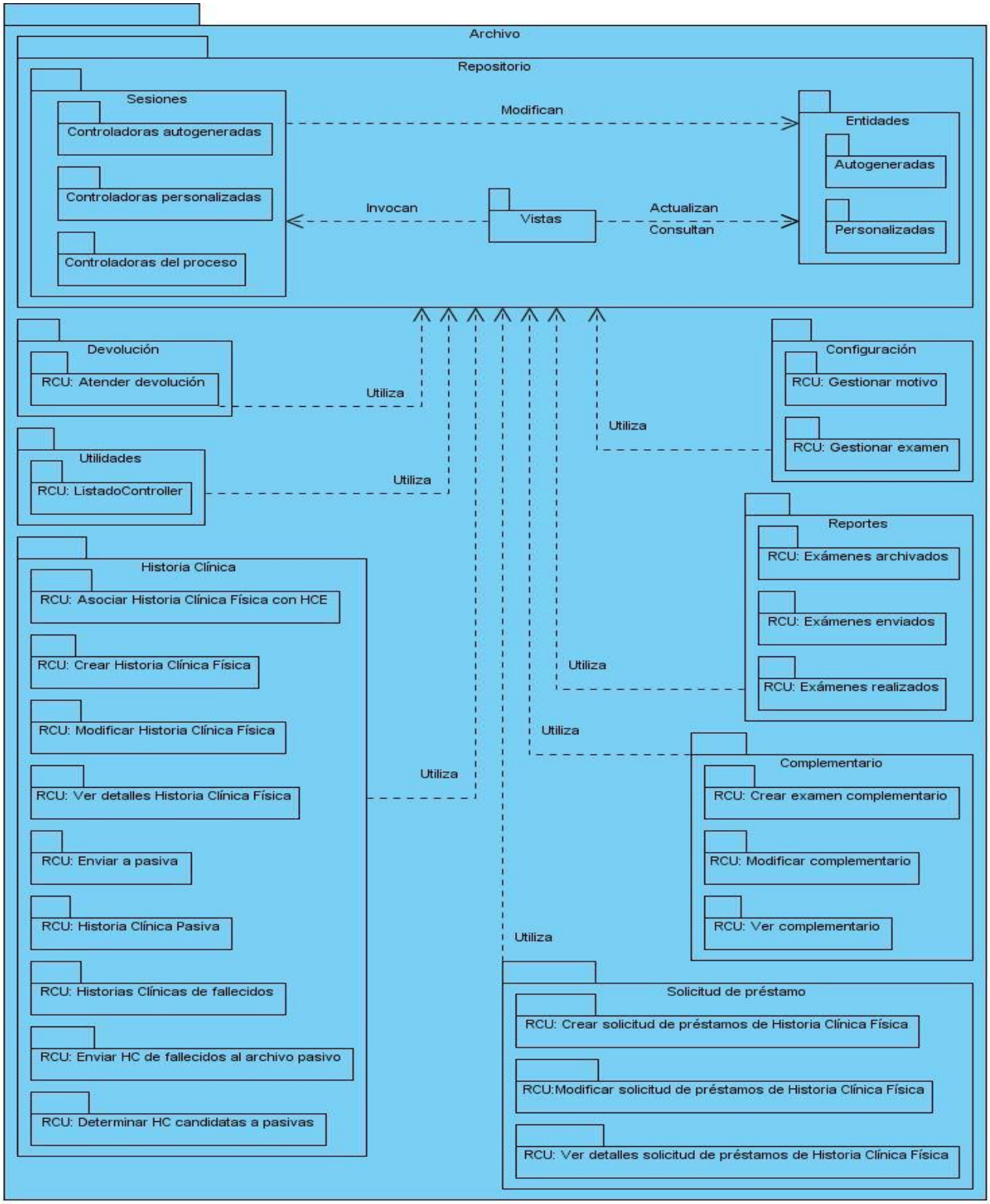
Se ha definido una estructura de paquetes, de esta forma todas las clases estarán agrupadas en el paquete repositorio de clases que a su vez contendrá 3 paquetes que se relacionan entre sí:

El paquete Sesiones: con todas las clases controladoras agrupadas en paquetes, uno tendrá las controladoras autogeneradas, otro las personalizaciones que se hacen sobre algunas controladoras autogeneradas y un paquete para las controladoras propias por procesos.

El paquete Entidades: que contendrá un paquete con las entidades autogeneradas y otro con las entidades personalizadas.

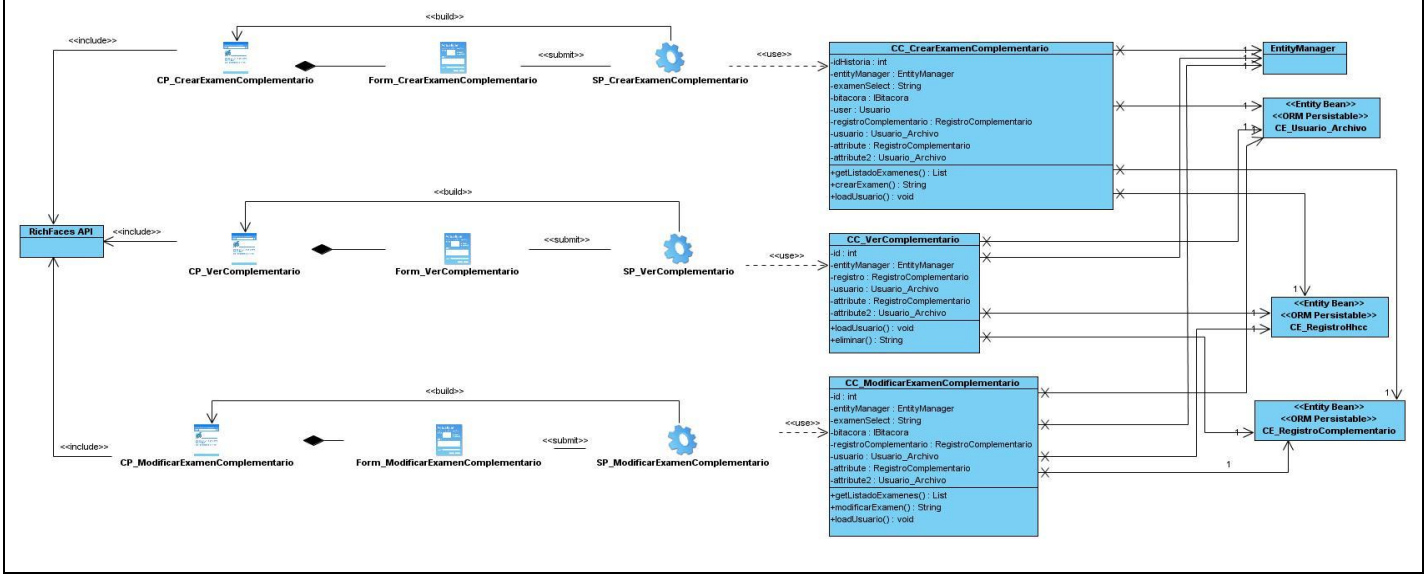
Y por último y conteniendo todas las vistas: el paquete Vistas.

A continuación se presenta el diagrama de paquetes del diseño:

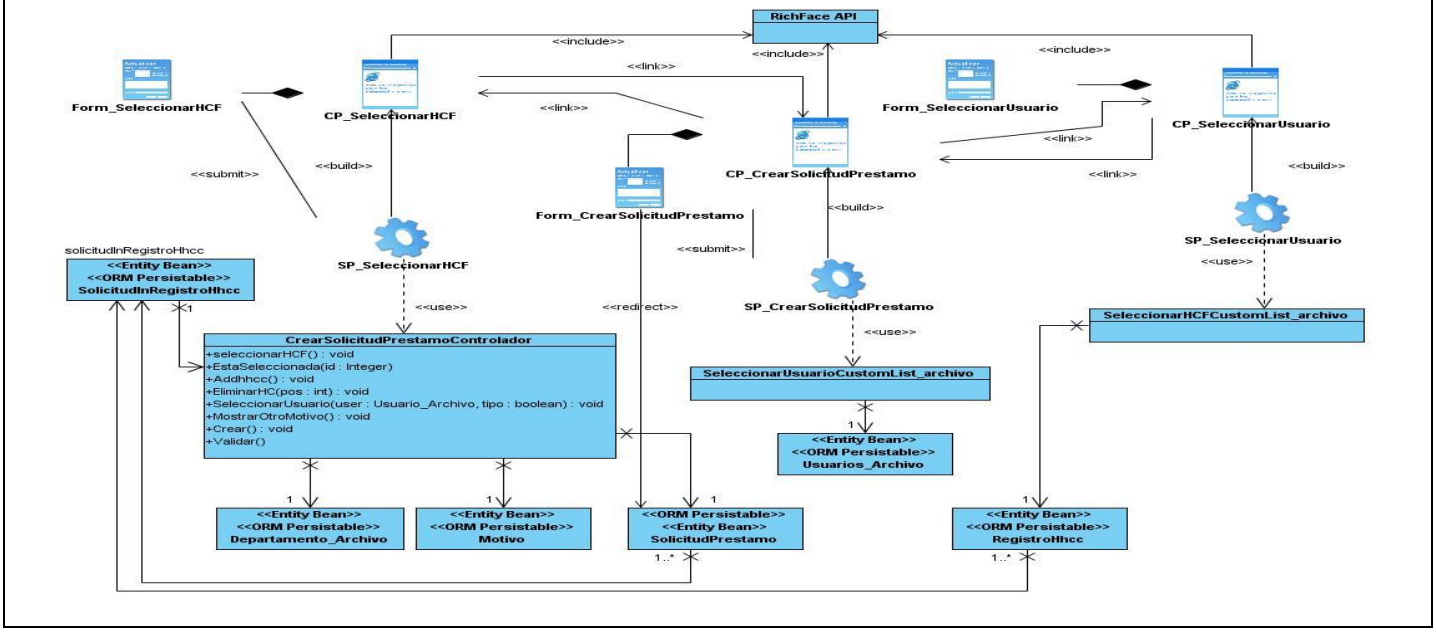

Archivo
Repositorio
Sesiones
Controladoras autogeneradas
Controladoras personalizadas
Controladoras del proceso
Modifican
Entidades
Autogeneradas
Personalizadas
Invocan
Vistas
Actualizan
Consultan
Devolución
RCU: Atender devolución
Utiliza
Configuración
RCU: Gestionar motivo
RCU: Gestionar examen
Utilidades
RCU: ListadoController
Reportes
RCU: Exámenes archivados
RCU: Exámenes enviados
RCU: Exámenes realizados
Historia Clínica
RCU: Asociar Historia Clínica Física con HCE
RCU: Crear Historia Clínica Física
RCU: Modificar Historia Clínica Física
RCU: Ver detalles Historia Clínica Física
RCU: Enviar a pasiva
RCU: Historia Clínica Pasiva
RCU: Historias Clínicas de fallecidos
RCU: Enviar HC de fallecidos al archivo pasivo
RCU: Determinar HC candidatas a pasivas
Complementario
RCU: Crear examen complementario
RCU: Modificar complementario
RCU: Ver complementario
Solicitud de préstamo
RCU: Crear solicitud de préstamos de Historia Clínica Física
RCU:Modificar solicitud de préstamos de Historia Clínica Física
RCU: Ver detalles solicitud de préstamos de Historia Clínica Física

A continuación se presentan algunos diagramas de clases de diseño acompañados de los diagramas de secuencia:

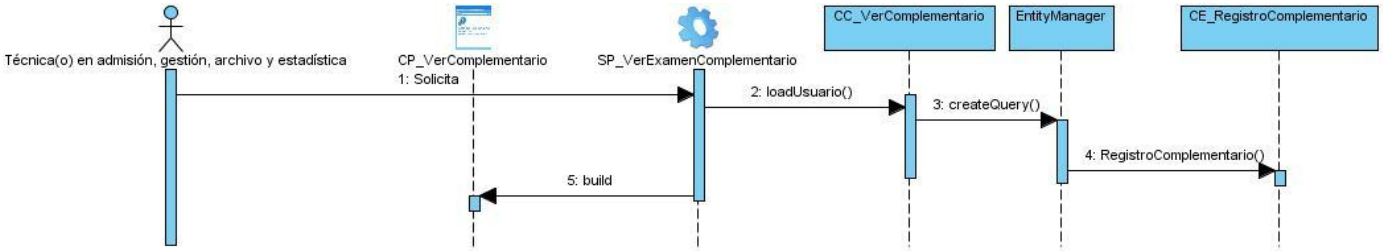
DCD_GestionarComplementario



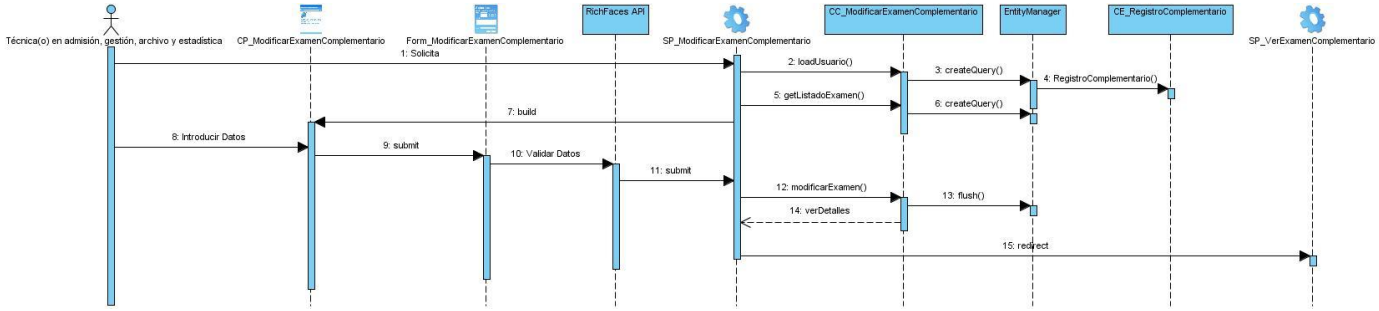
DCD_ Crear solicitud de préstamo



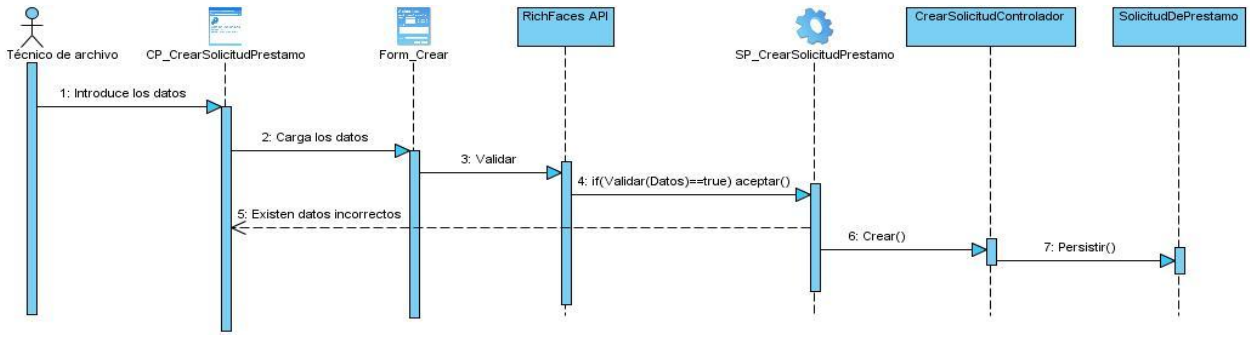
DS_VerComplementario



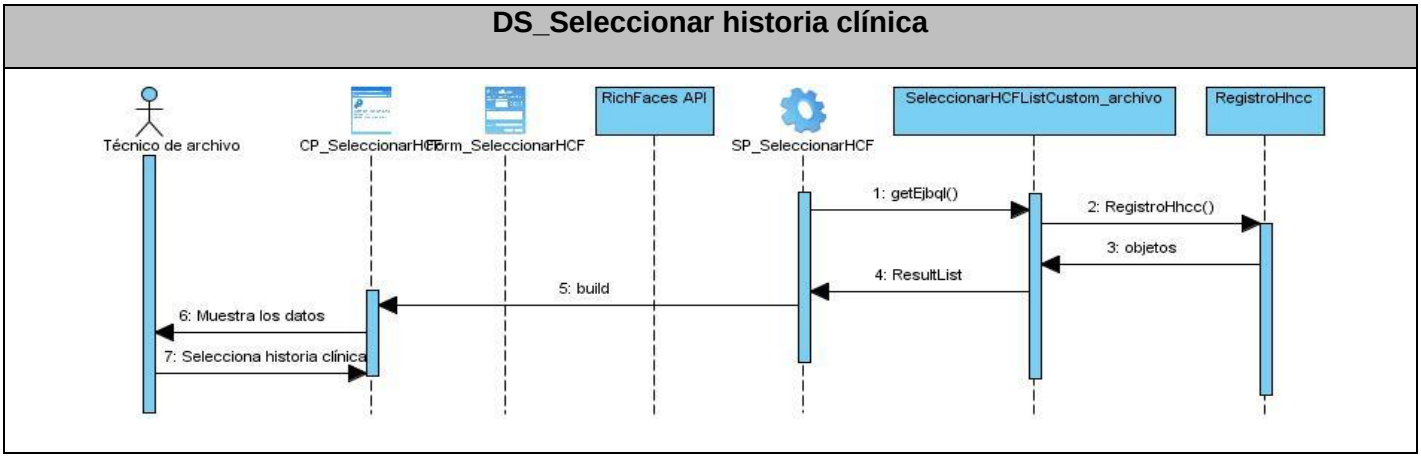
DS_ ModificarComplementario



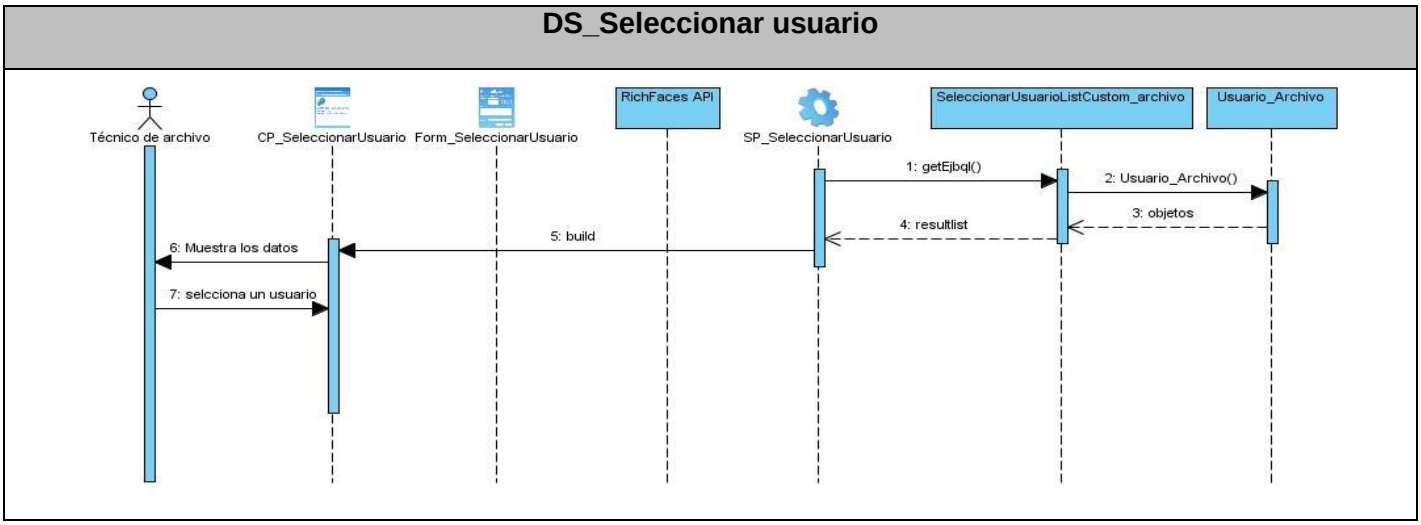
DS_Crear solicitud de préstamo



DS_Seleccionar historia clínica



DS_Seleccionar usuario



3.2. Descripción de las nuevas clases u operaciones necesarias

Nombre: CrearExamenComplementario	
Tipo de clase: Controladora	
Atributo	**Tipo**
-idHistoria	int
-entityManager	EntityManager
-examenSelect	String
-bitacora	IBitacora
-user	Usuario_Archivo
-attribute	RegistroComplementario

-attribute2	Usuario_Archivo
Para cada responsabilidad:	
Nombre:	getListadoExamenes
Descripción:	Permite obtener el listado de los tipos de examen.
Nombre:	crearExamen
Descripción:	Permite crear el examen complementario.
Nombre:	loadUsuario
Descripción:	Permite obtener los datos del usuario al que se debe enviar este examen.

Nombre: VerComplementario	
Tipo de clase: Controladora	
Atributo	**Tipo**
-id	int
-entityManager	EntityManager
-registro	RegistroComplementario
-usuario	Usuario_Archivo
-attribute	RegistroComplementario
-attribute2	Usuario_Archivo
Para cada responsabilidad:	
Nombre:	eliminar
Descripción:	Permite eliminar el complementario.
Nombre:	loadUsuario
Descripción:	Permite obtener los datos del usuario al que se debe enviar este examen.

Nombre: ModificarComplementario	
Tipo de clase: Controladora	
Atributo	**Tipo**
-id	int
-entityManager	EntityManager

-examenSelect	String
-bitacora	IBitacora
-registroComplementario	RegistroComplementario
-usuario	Usuario_Archivo
-attribute	RegistroComplementario
-attribute2	Usuario_Archivo

Para cada responsabilidad:

Nombre:	getListadoExamenes
Descripción:	Permite obtener el listado de los tipos de examen.
Nombre:	modificarExamen
Descripción:	Permite modificar los datos del examen complementario.
Nombre:	loadUsuario
Descripción:	Permite obtener los datos del usuario al que se debe enviar este examen.

Nombre: SeleccionarUsuarioListCustom_archivo

Tipo de clase: Lista

Atributo	Tipo
RESTRITIONS	String
EJBQL	String
usuarioarchivo	Usuario_Archivo

Para cada responsabilidad:

Nombre:	SeleccionarUsuarioListCustom_archivo
Descripción:	Permite listar los usuarios del sistema.
Nombre:	isMostrarBusqueda
Descripción:	Devuelve verdadero o falso según el tipo de búsqueda, avanzada o simple.
Nombre:	limpiar
Descripción:	Se ejecuta cuando se cambia de la búsqueda avanzada a la simple.

Nombre:	limpia
Descripción:	Se ejecuta para cambiar el tipo de búsqueda y mantener los criterios de búsqueda de la simple a la avanzada.

Nombre: SeleccionarHCFListCustom_archivo	
Tipo de clase: Lista	
Atributo	**Tipo**
RESTRITIONS	String
EJBQL	String
Para cada responsabilidad:	
Nombre:	SeleccionarHCFListCustom_archivo
Descripción:	Permite listar los usuarios del sistema.
Nombre:	isMostrarBusqueda
Descripción:	Devuelve verdadero o falso según el tipo de búsqueda, avanzada o simple.
Nombre:	limpiar
Descripción:	Se ejecuta cuando se cambia de la búsqueda avanzada a la simple.
Nombre:	limpia
Descripción:	Se ejecuta para cambiar el tipo de búsqueda y mantener los criterios de búsqueda de la simple a la avanzada.

Nombre: CrearSolicitudPrestamoControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
solicitud	SolicitudDePrestamo
registroHhcc	RegistroHhcc
receptor	Departamento_Archivo
fecha	Date

otroMotivo	String
usuarioByIdAprobador	Usuario_Archivo
aprobador	String
usuarioByIdSolicitante	Usuario_Archivo
solicitante	String
solregis	SolicitudPrestamoInRegistroHhcc
fechaDev	Date
hhccaux	ListadoController
prestamoInRegistroHhcc	List
hcfseleccionadas	Hashtable
hcfselected	int
mostrar	boolean
error	boolean
motivo	Motivo

Para cada responsabilidad:

Nombre:	CrearSolicitudControlador
Descripción:	Constructor de la clase
Nombre:	seleccionarHCF
Descripción:	Son almacenadas en una tabla hash las historias clínicas seleccionadas por el usuario para ser asignadas a la solicitud de préstamo correspondiente.
Nombre:	EstaSeleccionada
Descripción:	Devuelve si una historia clínica está seleccionada o no.
Nombre:	Addhhcc
Descripción:	Son adicionadas a una lista las historias clínicas seleccionadas. Para agregarles un atributo y persistirlas.
Nombre:	EliminarHC
Descripción:	Este método permite eliminar historias clínicas de la selección hecha anteriormente.
Nombre:	SeleccionarUsuario
Descripción:	Permite identificar al usuario que ha sido seleccionado, si es responsable o el receptor.
Nombre:	SeleccionarUsuario

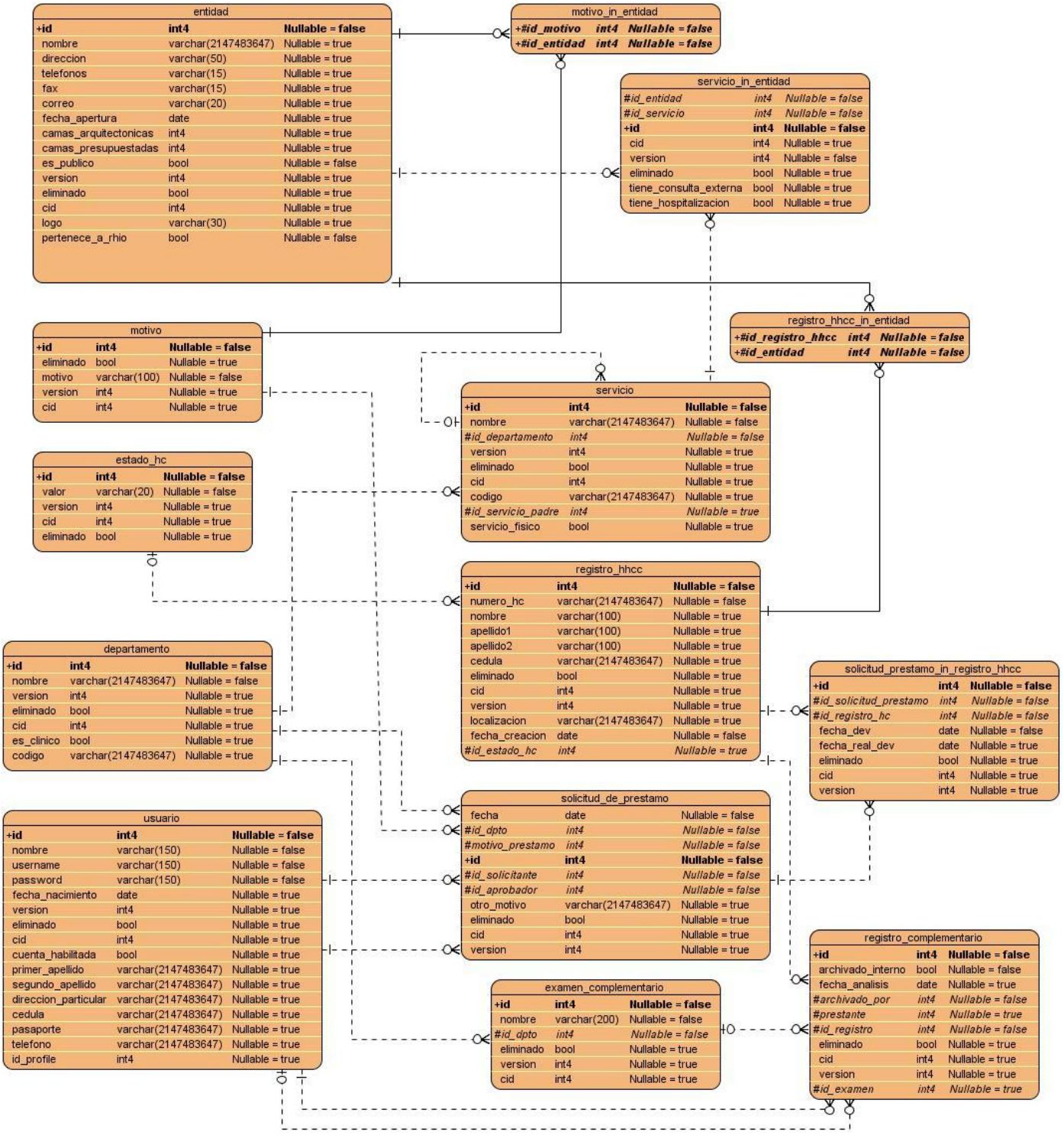
Descripción:	Permite eliminar a un usuario que ha sido seleccionado.
Nombre:	MostrarOtroMotivo
Descripción:	Este método es para propiciar que se muestre en la página cliente una vez deseado el componente para introducir el dato “Otro motivo”.
Nombre:	Crear
Descripción:	Permite crear una solicitud de préstamo y persistir esta información en la base de datos.
Nombre:	Validar
Descripción:	Permite validar la información que es entrada por el usuario.

3.3. Modelo de datos

Es aquel que se obtiene como resultado de la actividad: “Diseño de la Base de Datos”. Es el modelo que describe los datos, las relaciones existentes entre ellos, la semántica y las restricciones de consistencia de una aplicación o sistema de información. Estos se clasifican en tres grupos:

- “Modelos externos o lógicos basados en objetos: nos permite representar los datos que necesita cada usuario con las estructuras propias del lenguaje de programación que se vaya a usar.
- Modelos globales o lógicos basados en registros: ayuda a escribir los datos para el conjunto de usuarios.
- Modelos físicos de datos: orientado a la máquina.”(37)

Por lo general, un modelo de datos presenta dos sublenguajes: un Lenguaje de Definición de Datos o DDL (Data DefinitionLanguage), cuya función es describir, de forma abstracta, las estructuras de datos y las restricciones de integridad; y un Lenguaje de Manipulación de Datos o DML (Data ManipulationLanguage), que se orienta a describir las operaciones de manipulación de los datos. A la parte del DML enfocada a la recuperación de datos, se la suele conocer como Lenguaje de Consulta o QL (QueryLanguage). (37)
A continuación se presenta el Modelo de Datos para el módulo Archivo:


entidad
+id int4 Nullable = false
nombre varchar(2147483647) Nullable = true
direccion varchar(50) Nullable = true
telefonos varchar(15) Nullable = true
fax varchar(15) Nullable = true
correo varchar(20) Nullable = true
fecha_apertura date Nullable = true
camas_arquitectonicas int4 Nullable = true
camas_presupuestadas int4 Nullable = true
es_publico bool Nullable = false
version int4 Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
logo varchar(30) Nullable = true
pertenece_a_rhio bool Nullable = false
motivo_in_entidad
+#id_motivo int4 Nullable = false
+#id_entidad int4 Nullable = false
servicio_in_entidad
#id_entidad int4 Nullable = false
#id_servicio int4 Nullable = false
+id int4 Nullable = false
cid int4 Nullable = true
version int4 Nullable = false
eliminado bool Nullable = true
tiene_consulta_externa bool Nullable = true
tiene_hospitalizacion bool Nullable = true
motivo
+id int4 Nullable = false
eliminado bool Nullable = true
motivo varchar(100) Nullable = false
version int4 Nullable = true
cid int4 Nullable = true
registro_hhcc_in_entidad
+#id_registro_hhcc int4 Nullable = false
+#id_entidad int4 Nullable = false
servicio
+id int4 Nullable = false
nombre varchar(2147483647) Nullable = false
#id_departamento int4 Nullable = false
version int4 Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
codigo varchar(2147483647) Nullable = true
#id_servicio_padre int4 Nullable = true
servicio_fisico bool Nullable = true
estado_hc
+id int4 Nullable = false
valor varchar(20) Nullable = false
version int4 Nullable = true
cid int4 Nullable = true
eliminado bool Nullable = true
registro_hhcc
+id int4 Nullable = false
numero_hc varchar(2147483647) Nullable = false
nombre varchar(100) Nullable = true
apellido1 varchar(100) Nullable = true
apellido2 varchar(100) Nullable = true
cedula varchar(2147483647) Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
version int4 Nullable = true
localizacion varchar(2147483647) Nullable = true
fecha_creacion date Nullable = false
#id_estado_hc int4 Nullable = true
departamento
+id int4 Nullable = false
nombre varchar(2147483647) Nullable = false
version int4 Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
es_clinico bool Nullable = true
codigo varchar(2147483647) Nullable = true
solicitud_prestamo_in_registro_hhcc
+id int4 Nullable = false
#id_solicitud_prestamo int4 Nullable = false
#id_registro_hc int4 Nullable = false
fecha_dev date Nullable = false
fecha_real_dev date Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
version int4 Nullable = true
solicitud_de_prestamo
fecha date Nullable = false
#id_dpto int4 Nullable = false
#motivo_prestamo int4 Nullable = false
+id int4 Nullable = false
#id_solicitante int4 Nullable = false
#id_aprobador int4 Nullable = false
otro_motivo varchar(2147483647) Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
version int4 Nullable = true
usuario
+id int4 Nullable = false
nombre varchar(150) Nullable = false
username varchar(150) Nullable = false
password varchar(150) Nullable = false
fecha_nacimiento date Nullable = true
version int4 Nullable = true
eliminado bool Nullable = true
cid int4 Nullable = true
cuenta_habilitada bool Nullable = true
primer_apellido varchar(2147483647) Nullable = true
segundo_apellido varchar(2147483647) Nullable = true
direccion_particular varchar(2147483647) Nullable = true
cedula varchar(2147483647) Nullable = true
pasaporte varchar(2147483647) Nullable = true
telefono varchar(2147483647) Nullable = true
id_profile int4 Nullable = true
examen_complementario
+id int4 Nullable = false
nombre varchar(200) Nullable = false
#id_dpto int4 Nullable = false
eliminado bool Nullable = true
version int4 Nullable = true
cid int4 Nullable = true
registro_complementario
+id int4 Nullable = false
archivado_interno bool Nullable = false
fecha_analisis date Nullable = true
#archivado_por int4 Nullable = false
#prestante int4 Nullable = true
#id_registro int4 Nullable = false
eliminado bool Nullable = true
cid int4 Nullable = true
version int4 Nullable = true
#id_examen int4 Nullable = true

3.4. Descripción de las tablas

Nombre: comun.departamento		
Descripción: Tabla para registrar los datos de los departamentos.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del departamento (PK).
nombre	varchar	Nombre del departamento.
version	integer	Indica con que versión de la entidad se está trabajando.
eliminado	boolean	Permite saber si el área ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.
es_clinico	boolean	Indica si el departamento es clínico o no.
codigo	varchar	Indica el código del departamento.

Tabla 3.4.3 archivo.estado_hc

Nombre: archivo.examen_complementario		
Descripción: Tabla para registrar los datos que contiene un examen complementario.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del departamento (PK)
nombre	varchar[200]	Nombre del departamento
id_dpto	integer	Identificador del departamento (FK)
version	integer	Indica con que versión de la entidad se está trabajando
eliminado	boolean	Permite saber si el área ha sido eliminada
cid	integer	Identificador de modificaciones registradas en la bitácora

Tabla 3.4.4 archivo.examen_complementario

Nombre: archivo.motivo		
Descripción: Tabla para registrar los motivos por los que se puede prestar una historia clínica.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del motivo (PK).
motivo	varchar[100]	Nombre del motivo.
version	integer	Indica con que versión de la entidad se está trabajando.
eliminado	boolean	Permite saber si el área ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 3.4.5 archivo.motivo

Nombre: archivo.motivo_in_entidad		
Descripción: Tabla para registrar la relación entre las tablas archivo.motivo y comun.entidad.		
Atributo	**Tipo**	**Descripción**
id_motivo	integer	Identificador del departamento (PK y FK).
id_entidad	integer	Identificador de la entidad (PK y FK).

Tabla 3.4.6 archivo.motivo_in_entidad

Nombre: archivo.registro_complementario		
Descripción: Tabla para registrar los exámenes complementarios llegados al archivo.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del complementario (PK).
archivado_interno	boolean	Indica si el examen complementario fue archivado en la historio clínica.
fecha_analisis	date	Fecha en que se realizó el análisis.
archivado_por	integer	Identificador de la técnica(o) en admisión, gestión, archivo y estadística que archivó el examen (FK).
id_registro	integer	Identificador de la historia clínica (FK).
id_examen	integer	Identificador del examen complementario (FK).
version	Integer	Indica con que versión de la entidad se está trabajando.
eliminado	Boolean	Permite saber si el área ha sido eliminada.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 3.4.9 archivo.registro_complementario

Nombre: archivo.registro_hhcc		
Descripción: Tabla para registrar los datos de las historias clínicas.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la historia clínica física (PK).
numero_hc	varchar	Número de la historia clínica.
nombre	varchar[100]	Nombre del paciente al que pertenece la historia clínica.
apellido1	varchar[100]	Primer apellido del paciente al que pertenece la historia clínica.
apellido2	varchar[100]	Segundo apellido del paciente al que pertenece la

		historia clínica.
id_hce	integer	Identificador de la historia clínica electrónica del paciente en caso de que exista (FK).
cedula	varchar	Cédula del paciente al que pertenece la historia clínica.
localizacion	varchar	Localización de la historia clínica física en el archivo.
id_estado_hc	integer	Identificador del estado de la historia clínica (FK).
fecha_creacion	date	Fecha en que se creó la historia clínica.
version	integer	Indica con que versión de la entidad se está trabajando.
eliminado	boolean	Permite saber si el área ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 3.4.11 archivo.registro_hhcc_in_entidad

Nombre: archivo.solicitud_de_prestamo		
Descripción: Tabla para registrar los datos de las solicitudes de préstamos de las historias clínicas.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la solicitud de préstamo (PK).
fecha	date	Fecha de la solicitud.
departamento_receptor	integer	Identificador del departamento que solicita la historia clínica (FK).
motivo_prestamo	integer	Identificador del motivo por el cual se solicita el préstamo (FK).
id_solicitante	integer	Identificador de la persona que busca personalmente la historia clínica (FK).
id_aprobador	integer	Identificador de la persona que solicita la historia clínica (FK).
id_registro	integer	Identificador de la historia clínica solicitada (FK)
otro_motivo	varchar	Nombre del motivo del préstamo en caso que no sea uno de los que se brinda.
version	integer	Indica con que versión de la entidad se está trabajando.
eliminado	boolean	Permite saber si el área ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 3.4.12 archivo.solicitud_de_prestamo

Nombre: archivo.solicitud_prestamo_in_registro_hhcc
Descripción: Tabla para registrar los datos de las historias clínicas asociadas a un préstamo.

Atributo	Tipo	Descripción
id	integer	Identificador de la asociación (PK).
id_solicitud_prestamo	integer	Identificador de la solicitud de préstamo (FK).
id_registro_hc	integer	Identificador de la historia clínica solicitada (FK)
fecha_dev	date	Fecha en que se debe devolver la historia clínica.
fecha_real_dev	date	Fecha en que se devolvió la historia clínica.
version	integer	Indica con que versión de la entidad se está trabajando.
eliminado	boolean	Permite saber si el área ha sido eliminada.
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 3.4.13 archivo.solicitud_prestamo_in_registro_hhcc

3.5. Valoración de las técnicas de validación

Generalmente cuando se realiza un sistema se verifica que los datos introducidos por el usuario no contengan errores, a este proceso se le llama validar. Para facilitar las validaciones en la vista se propone utilizar los componentes de validación presentes en el framework JSF y para establecer las validaciones en los objetos del negocio, y garantizar que estas puedan ser utilizadas las veces que sean necesarias se utilizaron las anotaciones presentes en el paquete HibernateValidator.

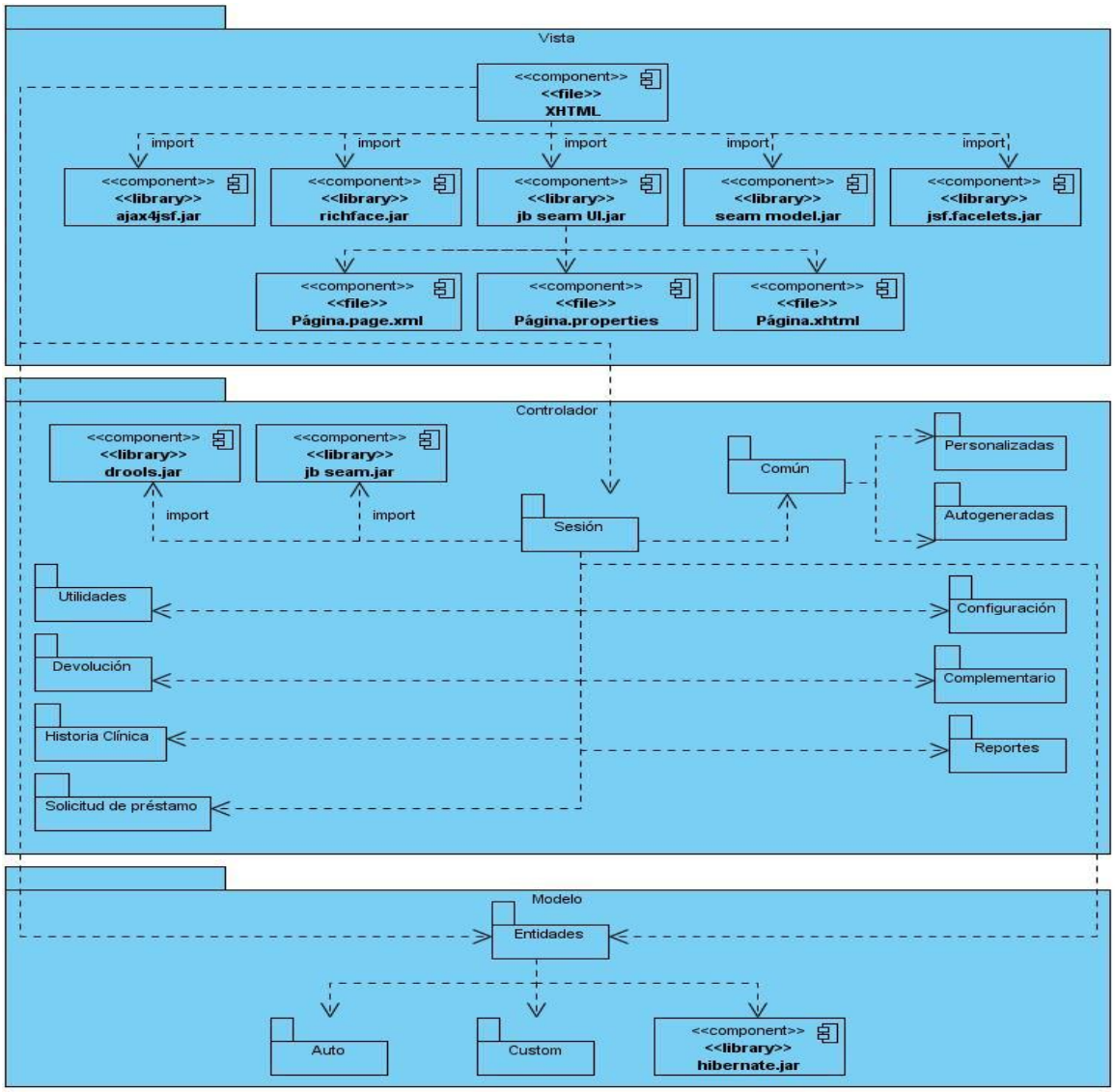
3.6. Vista de implementación

La vista de implementación es uno de los tipos de vistas físicas definidas por UML, la misma muestra el empaquetado físico de las partes reutilizables del sistema en unidades sustituibles, llamadas componentes.

“Una vista de implementación muestra los elementos físicos del sistema mediante componentes, así como sus interfaces y dependencias entre componentes. Los componentes son piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas.

El diagrama de componentes describe la descomposición física del sistema software en componentes, a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos. Los componentes identifican objetos físicos que hay en tiempos de ejecución, de compilación o de desarrollo y tienen identidad propia con una interfaz bien definida. Los componentes incluyen código en cualquiera de sus formatos, DLL, Active X, bases de datos,... Cada componente incorpora la implementación de ciertas clases del diseño del sistema.” (38)

Seguidamente se presenta el diagrama de componentes correspondiente al módulo Archivo:



En el presente capítulo se ha realizado una valoración crítica del diseño propuesto y se ha presentado el Modelo de Datos para los procesos del módulo. Además se han descrito las clases a utilizar y las tablas de la base de datos. En fin se han abordado los elementos principales del diseño y la implementación, obteniéndose los artefactos correspondientes como son los diagramas de: clase del diseño, secuencia, paquetes y componentes.

CAPÍTULO 4: MODELO DE PRUEBAS

En el este capítulo se presenta el modelo de pruebas, definiendo además que tipo de pruebas se realizarán en aras de lograr un sistema con la calidad requerida.

Una prueba de software es la ejecución de un programa de software con el objetivo de detectar si el mismo realiza correctamente las funciones para las cuales fue creado. Los resultados de dichas pruebas deben ser revisados en profundidad y ha de evitarse que las mismas sean realizadas por los programadores que participaron en el desarrollo de la aplicación. Además estas deben incluir tanto entradas válidas y esperadas como entradas inválidas e inesperadas.

Entre las estrategias de pruebas utilizadas se encuentran las pruebas de caja negra. Estas son pruebas funcionales sin acceso al código fuente de las aplicaciones que trabajan con entradas y salidas. (39)

Para la realización de estas pruebas se pueden emplear diferentes técnicas, entre ellas están:

- Métodos de prueba basados en grafos.
- Análisis de valores límites.
- Prueba de la tabla ortogonal.
- Partición equivalente.

Siendo esta último la técnica a utilizar en el presente trabajo. Pressman presenta la partición equivalente como un “método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. ” (40)

“El objetivo de la partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases.” (41)

Vale destacar que la partición equivalente es subjetiva y dos probadores que prueban un programa complejo pueden llegar a diferentes conjuntos de particiones.

4.1. Descripción de los casos de pruebas

Caso de uso: Crear examen complementario

Escenarios del crear examen complementario	Descripción de la funcionalidad	Flujo Central
EC 1: Crear examen complementario	Crear un examen complementario satisfactoriamente.	Se selecciona la opción Crear registro. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se crea el examen complementario. Muestra la interfaz Ver detalles de examen complementario. Ver DCP: **Ver detalles de examen complementario.**
EC 2: Cancelar operación	Cancelar la opción de Crear examen complementario.	Se selecciona la opción Crear registro. 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 registro. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.

Tabla 4.1.1. Crear examen complementario

Id del escenario	EC 1	EC 2	EC 3
Escenario	Crear examen complementario	Cancelar operación	Existen datos incompletos
Variable 1 (Examen)	V		I
Variable 2 (Fecha)	V		I
Botón 1 (Aceptar)	NA		NA
Botón 1 (Cancelar)		NA	
Respuesta del Sistema	Crea el examen complementario y se visualiza la interfaz ver detalles.	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.
Resultado de la Prueba			

Tabla 4.1.2. SC Crear examen complementario

Caso de uso: Crear solicitud de préstamo

Escenarios del crear solicitud de préstamo	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Crear solicitud de préstamo	Crea una solicitud de préstamo satisfactoriamente.	Se selecciona la opción Crear solicitud de préstamo. Se muestra la interfaz para Seleccionar historia clínica. Ver caso de prueba

		Seleccionar historia clínica. Se muestra un listado con las historias clínicas seleccionadas y la opción de eliminar historia clínica de la selección. Se introduce el dato correspondiente. Se introducen los datos generales correspondientes de la solicitud de préstamo. Se selecciona el usuario responsable que hace la solicitud. Muestra la interfaz para seleccionar usuario. Ver DCP **Seleccionar usuario**. Se selecciona el usuario al que se le entregó la historia clínica. Ver DCP **Seleccionar usuario**. Se selecciona la opción aceptar. Se muestra la interfaz Ver detalles de solicitud. Ver DCP **Ver detalles de solicitud**. El caso de uso termina.
EC 2: Cancelar operación	Cancelar la operación de Crear solicitud de préstamo.	Se selecciona la opción Crear solicitud de préstamo. Se muestra la interfaz para Seleccionar historia clínica. Ver DCP **Seleccionar historia clínica.** Se muestra un listado con las historias clínicas seleccionadas y la opción de eliminar historia clínica de la selección. Se introduce el dato correspondiente. Se introducen los datos generales

		correspondientes de la solicitud de préstamo. Se selecciona el usuario responsable que hace la solicitud. Muestra la interfaz para seleccionar usuario. Ver DCP **Seleccionar usuario**. Se selecciona el usuario al que se le entregó la historia clínica. Ver DCP **Seleccionar usuario**. Se selecciona la opción cancelar. Se cancela la creación de la solicitud de préstamo. Se vuelve a la vista anterior. El caso de uso termina.
EC 3: Eliminar historia clínica de la selección.	Permite eliminar una historia clínica que halla sido seleccionada.	Se selecciona la opción Crear solicitud de préstamo. Se muestra la interfaz para Seleccionar historia clínica. Ver DCP **Seleccionar historia clínica**. Se muestra un listado con las historias clínicas seleccionadas y la opción de eliminar historia clínica de la selección. Se introduce el dato correspondiente. Se selecciona la opción eliminar historia clínica de la selección. Se elimina la historia clínica del listado de historias clínicas seleccionadas. Se introducen los datos generales correspondientes de la solicitud de préstamo.

		Se selecciona el usuario responsable que hace la solicitud. Muestra la interfaz para seleccionar usuario. Ver DCP **Seleccionar usuario**. Se selecciona el usuario al que se le entregó la historia clínica. Ver DCP **Seleccionar usuario**. Se selecciona la opción aceptar. Se muestra la interfaz Ver detalles de solicitud. Ver DCP **Ver detalles de solicitud**. El caso de uso termina.
EC 4: 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 solicitud de préstamo. Se muestra la interfaz para Seleccionar historia clínica. Ver DCP **Seleccionar historia clínica**. Se muestra un listado con las historias clínicas seleccionadas y la opción de eliminar historia clínica de la selección. Se introduceny seleccionan los datos correspondientes de forma incompleta. Se selecciona la opción aceptar. Se muestra el mensaje de error: “Existen datos incorrectos.”
EC 5: Existen datos incorrectos.	Luego de haber introducido los datos, el sistema los verifica y valida, de haber datos incorrectos muestra un indicador sobre los campos	Se selecciona la opción Crear solicitud de préstamo. Se muestra la interfaz para Seleccionar historia clínica. Ver DCP **Seleccionar historia clínica.**

	incorrectos.	Se muestra un listado con las historias clínicas seleccionadas y la opción de eliminar historia clínica de la selección. Se introducen y seleccionan los datos correspondientes de la forma incorrecta. Se selecciona la opción aceptar. Se muestra el mensaje de error: “Existen datos incompletos.”

Tabla 4.1.7. Crear solicitud de préstamo

En el presente capítulo se ha mostrado la estrategia y la técnica de pruebas a utilizar. Además se han diseñado los casos de prueba que guiarán el proceso de revisión de la calidad del sistema.

CONCLUSIONES

1. Los sistemas de información hospitalaria analizados que realizan procesos de gestión de archivo, no se adecúan a las necesidades reales del “alas HIS”.
2. A partir de la valoración de las posibles tecnologías y herramientas a utilizar, se determinó adoptar la arquitectura definida por el departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones, por ser la más idónea y posibilitar el desarrollo de un sistema robusto y flexible.
3. La utilización de las pautas definidas garantiza la uniformidad visual de todas las interfaces del sistema.
4. El análisis de los patrones de diseño a utilizar ha facilitado la construcción del mismo.

RECOMENDACIONES

1. Implementar el proceso Gestionar Archivo que permita el diseño de este, teniendo en cuenta la cantidad de secciones, estantes y pisos de cada estante. Permitiendo también la selección del método de organización de las historias clínicas en el mismo, ya sea el convencional o el dígito terminal.
2. Crear el nomenclador de exámenes del sistemas alas HIS-RIS e integrar la gestión de los exámenes complementarios con este.
3. Incluir mediante validación de usuario un nivel de seguridad que garantice que las historias clínicas solo sean entregadas al personal designado para ello.

REFERENCIAS BIBLIOGRÁFICAS

1. García Nogueira, Keila Ing. PROCESOS BÁSICOS EPIDEMIOLÓGICOS PARA UN SISTEMA DE GESTIÓN HOSPITALARIO. 2010/09/16. [Disponible en http://informatica2009.sld.cu/]
2. Florina, Gatica Lara y Fernando J., Fernández Puerto. Sistema de Información Hospitalaria. s.l.: UNAM - Facultad de Medicina.
3. Yetano J, Montero AB, Saracho R. Disminución de los errores de archivado en un archivo hospitalario. Rev Calidad Asistencial 1995; 6:118-20.
4. InterSystems. 2010/07/21. [Disponible en: http://www.intersystems.es/page/es/valen_computer.html]
5. Valen Computer S.A. 2010/07/20. [Disponible en: http://www.valen.es/cas/gowin_hc.html]
6. Indian. 2010/09/25. [Disponible en: http://www.indianinfotech.in/hospital_management_system.htm]
7. SIVSA. 2010/09/27. [Disponible en: http://www.sivsa.com/]
8. El Hospital. 2010/09/27. [Disponible en: http://www.elhospital.com/eh/secciones/EH/ES/seccion_HTML.html]
9. CEDISAP. 2010/10/03. [Disponible en: http://www.sld.cu/instituciones/cedisap/cedi1.htm]
10. Sistema de Información Hospitalaria. Módulo de Archivo. Ayuda de Archivo.
11. Josep Casanovas. Usabilidad y arquitectura del software. 2010/07/16. [Disponible en: http://www.desarrolloweb.com/articulos/1622.php]
12. Java Enterprise Edition(Versión Empresarial de Java). Java Enterprise Edition(Versión Empresarial de Java).2010/07/03 [Disponible en: http://java.sun.com/javaee/, http://java.sun.com/j2ee/overview.html]
13. Allen, Dan**.**Seam in Action.2008.
14. Devesa, Arianna; Muñoz, Reiniel. Módulo Hospitalización del Sistema de Información Hospitalaria alas HIS. Facultad 7. Universidad de las Ciencias Informáticas. La Habana, 2009. Página 19.
15. Facelets. 2010/09/24. [Disponible en: http://www.synaptic-it.com/blog/?p=8]
16. Drools I. Introducción a los motores de reglas de negocios. 2010/02/18. [Disponible en: http://rincew.blogspot.com/2005/11/drools-i-introduccin-los-motores-de.html]
18. Jamae David y Johnson, Peter**.**JBoss in Action*.* 2009.

19. Crespo, David; Fernández, Nelson Francisco. Desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS. Facultad 7. Universidad de las Ciencias Informáticas. La Habana, 2010. Página 11.
20. Jackwind Li Guojie. UI Development with JaveServer Faces.
21. Hat, Red.RichFaces developer Guide.2007.
22. Hat, Red. Ajax4jsf Developer Guide.2007.
23. PostgreSQL. PostgreSQL. [Disponible en: http://www.postgresql.org/]
24. Bauer, Cristian y King, Gavin.Java Persistence with Hibernate. 2005.
25. Crespo David, Fernández Nelson Francisco. Desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS. Facultad 7. Universidad de las Ciencias Informáticas. La Habana, 2010. Página 14.
26. JavaBeans Enterprise. 2010/02/17. [Disponible en: http://www.programacion.com/java/tutorial/javabeans/]
27. Agüero Martín. Introducción a Spring Framework. Universidad de Palermo. 2007. [Disponible en : http://www.palermo.edu/ingenieria/downloads/introduccion_spring_framework_v1.0.pdf]
28. XHTML™ 1.0: El Lenguaje de Etiquetado Hipertextual Extensible. 2010/02/19. [Disponible en: http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm]
29. Lenguaje Java. Lenguaje Java. 2010/07/01. [Disponible en: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html]
30. Crespo, David; Fernández, Nelson Francisco. Desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS, Facultad 7. Universidad de las Ciencias Informáticas. La Habana, 2010. Página 18.
31. Arquitectura Modelo/Vista/Controlador. 2010/09/15. [Disponible en: http://www.ulpgc.es/otros/tutoriales/java/Apendice/arq_mvc.html]
31. Breve guía. Breve guía. 2010/07/03. [Disponible en: http://www.slideshare.net/Benedeti/ide-eclipse-breve-gua-201399]
32. NetBeans. 2011/03/16. [Disponible en: http://netbeans.org/index_es.html]
33. NetBeans. 2011/03/16. [Disponible en: http://netbeans.org/features/web/index.html]
34. PgAdmin III. PgAdmin III.2010/07/03. [Disponible en: http://www.guia-ubuntu.org/index.php?title=PgAdmin_III]

35. Free Download Manager. 2010/07/15. [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/. 15]
36. Especificaciones de requerimientos. 2010/07/11. [Disponible en: http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos]
37. Fernández Rivera, Javier. Modelo de datos. 2011/03/15. [Disponible en: www.aurea.es]
38. Platero Dueñas Carlos. UML. El Modelado dinámico y de implementación. 2009. 2011/03/15. [Disponible en: http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf]
39. ITM. It-Mentor. Capacitación y guía para el desarrollo de software. Pruebas de software. 2011/04/26. [Disponible en: http://bycpx.files.wordpress.com/2011/01/pruebas-intro.pdf]
40. Roger S. Pressman. Ingeniería del software, un enfoque práctico, páginas 281-322. McGrawHill, quinta edition, 2002.
41. Ron Patton. Software Testing, página 408. Sams Publishing, 2005.

BIBLIOGRAFÍA

1. Agüero Martín. Introducción a Spring Framework. Universidad de Palermo. 2007. [Disponible en : http://www.palermo.edu/ingenieria/downloads/introduccion_spring_framework_v1.0.pdf]
2. Allen, Dan**.**Seam in Action.2008.
3. Arquitectura Modelo/Vista/Controlador. 2010/09/15. [Disponible en: http://www.ulpgc.es/otros/tutoriales/java/Apendice/arq_mvc.html]
4. Bauer, Cristian y King, Gavin.Java Persistence with Hibernate. 2005.
5. Breve guía. Breve guía. 2010/07/03. [Disponible en: http://www.slideshare.net/Benedeti/ide-eclipse-breve-gua-201399]
6. Casanovas, Josep. Usabilidad y arquitectura del software. 2010/07/16. [Disponible en: http://www.desarrolloweb.com/articulos/1622.php]
7. CEDISAP. 2010/10/03. [Disponible en: http://www.sld.cu/instituciones/cedisap/cedi1.html]
8. Crespo, David; Fernández, Nelson Francisco. Desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS. Facultad 7. Universidad de las Ciencias Informáticas. La Habana, 2010.
9. CEDISAP. 2010/10/03. [Disponible en: http://www.sld.cu/instituciones/cedisap/cedi1.html]
10. Devesa, Arianna; Muñoz, Reiniel. Módulo Hospitalización del Sistema de Información Hospitalaria alas HIS. Facultad 7. Universidad de las Ciencias Informáticas. La Habana, 2009.
11. Drools I. Introducción a los motores de reglas de negocios. 2010/02/18. [Disponible en: http://rincew.blogspot.com/2005/11/drools-i-introduccin-los-motores-de.html]
12. El Hospital. 2010/09/27. [Disponible en: http://www.elhospital.com/eh/secciones/EH/ES/seccion_HTML.html]
13. Especificaciones de requerimientos. 2010/07/11. [Disponible en: http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos]
14. Facelets. 2010/09/24. [Disponible en: http://www.synaptic-it.com/blog/?p=8]
15. Fernández Rivera, Javier. Modelo de datos. 2011/03/15. [Disponible en: www.aurea.es]
16. Florina, Gatica Lara y Fernando J., Fernández Puerto. Sistema de Información Hospitalaria. s.l.: UNAM - Facultad de Medicina.

17. Free Download Manager. 2010/07/15. [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/. 15]
18. García Nogueira, Keila Ing. PROCESOS BÁSICOS EPIDEMIOLÓGICOS PARA UN SISTEMA DE GESTIÓN HOSPITALARIO. 2010/09/16. [Disponible en http://informatica2009.sld.cu/]
19. Guerrero, Luis A. UML - Diagramas de interacción. 2011/03/15. [Disponible en: http://eva.uci.cu/mod/resource/view.php?id=14072]
20. Hat, Red. Ajax4jsf Developer Guide.2007.
21. Hat, Red. RichFaces developer Guide.2007.
22. Indian. 2010/09/25. [Disponible en: http://www.indianinfotech.in/hospital_management_system.html]
23. InterSystems. 2010/07/21. [Disponible en: http://www.intersystems.es/page/es/valen_computer.html]
24. ITM. It-Mentor. Capacitación y guía para el desarrollo de software. Pruebas de software. 2011/04/26. [Disponible en: http://bycpx.files.wordpress.com/2011/01/pruebas-intro.pdf]
25. Jackwind Li Guojie. UI Development with JaveServer Faces.
26. Jamae, David y Johnson, Peter. JBoss in Action. 2009.
27. JavaBeans Enterprise. 2010/02/17. [Disponible en: http://www.programacion.com/java/tutorial/javabeans/]
28. Java Enterprise Edition(Versión Empresarial de Java). Java Enterprise Edition(Versión Empresarial de Java).2010/07/03 [Disponible en: http://java.sun.com/javaee/, http://java.sun.com/j2ee/overview.html]
29. Lenguaje Java. Lenguaje Java. 2010/07/01. [Disponible en: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html]
30. NetBeans. 2011/03/16. [Disponible en: http://netbeans.org/]
31. PgAdmin III. PgAdmin III.2010/07/03. [Disponible en: http://www.guia-ubuntu.org/index.php?title=PgAdmin_III]
32. Platero, Dueñas, Carlos. UML. El Modelado dinámico y de implementación. 2009. 2011/03/15. [Disponible en: http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf]

33. PostgreSQL. PostgreSQL. [Disponible en: http://www.postgresql.org/]
34. Roger S. Pressman. Ingeniería del software, un enfoque práctico, páginas 281-322. McGrawHill, quinta edition, 2002.
35. Ron Patton. Software Testing, página 408. Sams Publishing, 2005.
36. Sistema de Información Hospitalaria. Módulo de Archivo. Ayuda de Archivo.
37. SIVSA. 2010/09/27. [Disponible en: http://www.sivsa.com/]
38. Valen Computer S.A. 2010/07/20. [Disponible en: http://www.valen.es/cas/gowin_hc.html]
39. XHTML™ 1.0: El Lenguaje de Etiquetado Hipertextual Extensible. 2010/02/19. [Disponible en: http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm]
40. Yetano J, Montero AB, Saracho R. Disminución de los errores de archivado en un archivo hospitalario. Rev Calidad Asistencial 1995; 6:118-20.

GLOSARIO DE TÉRMINOS

AJAX: Técnica para el desarrollo Web que posibilita la creación de aplicaciones interactivas.

API (ApplicationProgramming Interface): Conjunto de funciones y procedimientos que poseen algunas librerías con el objetivo de ser utilizadas por otro software como una capa de abstracción.

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

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

HTML: Siglas de HyperTextMarkupLanguage (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

ICEFaces: Biblioteca de componentes JSF Ajax.

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

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

UI: Interfaz de usuario.

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