Universidad de las Ciencias Informáticas

FACULTAD 2

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

Componente para incrementar la exactitud de la representación de imágenes de modelos de examen físico en el Sistema de Información Hospitalaria del CESIM

Autores: Ricardo Manuel Rodriguez Palacios

Pedro Pablo Delgado Martell

Tutor: Ing.Liuver Romel Sañudo Ortiz

Cotutor: Ing.Wilder Hernández González

La Habana, junio 2014

"Año 56 de la Revolución"

La revolución no se lleva en los labios para vivir de ella, se lleva en el corazón para morir por ella.

CHE

A mis padres abuelos y tíos por apoyarme y confiar siempre en mí.

Ricardo Manuel Rodriguez Palacios

A mi familia y amigos en especial a mi madre Rosa y mi padre Pedro Pablo.

Pedro Pablo Delgado Martell

Agradecimientos

Queremos agradecer a nuestro tutor y cotutor, Liuver y Wilder, a Josué y Adrián por su ayuda incondicional, y en general a todos las personas que contribuyenron a lo largo de la carrera en nustra formación como persona y como profesional.

A mis padres Marcia y Ricardo por siempre darme su apoyo y esfuerzo incondicional que han hecho para poder alcanzar mis metas.

A mis abuelos Mirtha y Tatín por ser mis segundos padres.

A mis tíos Nani y Pompi por siempre esrtar siempre ahí.

De Ricardo

Quiero agradecer a mis padres por dar todo su esfuerzo y sacrificio con el objetivo de guiarme y darme todo lo que está a su alcance.

A mi hermana, abuela, tia y amigos que no me alcanzaría el tiempo para mencionarlos a todos.

De Pedro

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 18 días del mes de junio del año 2014.

Firma del autor *Pedro Pablo Delgado Martell*	*Firma del autor* *Ricardo Manuel Rodriguez Palacios*
Firma del tutor *Ing.Liuver Romel Sañudo Ortiz*	*Firma del cotutor* *Ing. Wilder Hernández González*

DATOS DE CONTACTO

Tutores

Ing. Liuver Romel Sañudo Ortiz

Graduado de Ingeniería en Ciencias Informáticas en el año 2011 en la Universidad de las Ciencias Informáticas. Actualmente se desempeña como especialista en la Universidad de las Ciencias Informáticas (UCI). Desde que se graduó se vinculó a las actividades productivas en el Departamento de Gestión Hospitalaria del Centro de Informática Médica (CESIM), desempeñándose como implementador. Ha participado como cotutor en tesis de cursos anteriores obteniendo resultados satisfactorios.

Correo electrónico: lrsanudo.uci.cu

Ing. Wilder Hernández González

Graduado de Ingeniería en Ciencias Informáticas en el año 2012 en la Universidad de las Ciencias Informáticas. Actualmente se desempeña como especialista en la Universidad de las Ciencias Informáticas (UCI). Desde que se graduó se vinculó a las actividades productivas en el Departamento de Gestión Hospitalaria del Centro de Informática Médica (CESIM), desempeñándose como implementador. Ha participado como cotutor en tesis de cursos anteriores obteniendo resultados satisfactorios.

Correo electrónico: whgonzalez@uci.cu

Resumen

El Centro de Informática Médica (CESIM) desarrolló un Sistema de Información Hospitalaria (HIS por sus siglas en inglés), el mismo posee un conjunto de módulos para la gestión de la información, entre los que se encuentran los módulos de Consulta Externa y Emergencias. Para el módulo de Consulta Externa el sistema cuenta con la especialidad de cirugía colorrectal donde se realiza la representación de afecciones identificadas en el examen proctológico.

La funcionalidad antes mencionada cuenta con una sola imagen, y un solo pincel para realizar la representación de todas las lesiones. Está situación conlleva a que no se pueda representar lesiones en zonas que no estén visibles en la imagen, con el pincel solo se pueden dibujar solo seis lesiones, y no se pueden representar lesiones internas.

El presente trabajo centra su objetivo en el desarrollo de un componente para realizar una descripción exacta de las lesiones identificadas que repercute en un diagnóstico más acertado, que la imagen resultante tenga un formato entendible y procesable para el trabajo de los especialistas del módulo Emergencias del HIS del CESIM. La imagen resultante será almacenada en formato base64, lo que posibilitará que los demás módulos del sistema puedan usar el componente y obtener la imagen legible para el análisis de la misma.

Su diseño está guiado por el Proceso Unificado de Desarrollo, empleando Eclipse en su versión 3.4 como entorno de desarrollo, Java como lenguaje de programación, *PostgreSQL 9.1* como gestor de base de datos y *Visual Paradigm 8.0* como herramienta de modelado.

Palabras claves: CESIM, Sistema de Información Hospitalaria, salud, imágenes de modelos de examen físico, cirugía colorrectal

Índice de contenidos

Introducción

El surgimiento de las Tecnologías de la Información y las Comunicaciones (TIC), y con ello la posibilidad de crear sistemas informáticos para la gestión de la información, ha permitido que el empleo de sistemas informáticos en los principales sectores de la sociedad haya tenido un aumento considerable, gracias a los beneficios que aporta. Dentro de estos beneficios podemos encontrar el acceso rápido a la información y por ende mejora en la atención a los usuarios. Además la generación de informes e indicadores, permiten corregir fallas difíciles de detectar y controlar con un sistema manual, entre otros.

Una de las áreas donde ha influido significativamente el uso de las TIC es en el área de la salud, encargada del control y supervisión de los centros sanitarios. La necesidad de mejorar estas operaciones y actividades es en ocasiones crucial para una buena administración de los recursos activos de un hospital y para emitir un diagnóstico más certero.

Gracias a la TIC es posible crear ciertas herramientas y procedimientos para establecer mejores diagnósticos a pacientes con síntomas y padecimiento, reducir los tiempos de ejecución de análisis e investigaciones médicas. Con el paso del tiempo, el desarrollo de estas herramientas ha alcanzado un gran auge, desde pequeñas aplicaciones hasta sistemas completos de gestión y procesamiento de la información. Entre estos sistemas se encuentran los Sistemas de Información Hospitalaria conocido por sus siglas en inglés como HIS (*Hospital Information System*).

Un HIS es un sistema de información orientado a satisfacer las necesidades de generación de información, para recolectar, procesar, almacenar, reinterpretar y comunicar datos médico-administrativos de cualquier institución hospitalaria. Permitiendo la optimización de los recursos humanos y materiales, así como minimizar los inconvenientes burocráticos que enfrentan los pacientes (1).

Cuba no está aislada del acontecer tecnológico mundial, por lo que se ha dado a la tarea de potenciar la informatización de todos los sectores de la sociedad, haciendo énfasis en el sector de la salud. Con el objetivo de contribuir a la informatización del mismo, un conjunto de instituciones que se dedican al desarrollo de software de este tipo, cooperan en la construcción de soluciones informáticas, siendo la Universidad de las Ciencias Informáticas (UCI) una de estas.

El departamento de Sistema de Gestión Hospitalaria, perteneciente al Centro de Informática Médica (CESIM) de la UCI, está dedicado al desarrollo de software, sistemas, servicios y soluciones de alta calidad y competitividad para la optimización del trabajo y mejoramiento de la atención médica. Este departamento desarrolla un HIS, con el propósito de centralizar la información generada en cada uno de los servicios de las instituciones de salud, desde emergencias hasta la hospitalización y lograr elevar la calidad de la atención al paciente.

El HIS está compuesto por varios módulos que interconectan las diferentes áreas de una institución hospitalaria como son: Emergencia, Epidemiología, Farmacia, Consulta Externa, Hospitalización, Admisión y otros. El módulo de Consulta Externa es el encargado de gestionar la información que se genera durante la atención a pacientes en esta área, además de crear las hojas de consultas especializadas con un conjunto de solicitudes asociadas. Entre los servicios que brinda este módulo se encuentra el de Cirugía Colorrectal, el cual tiene dentro de sus funcionalidades representar afecciones identificadas en el examen proctológico. Esta funcionalidad permite dibujar en la imagen del modelo de examen físico las lesiones encontradas durante el examen.

El área de trabajo donde se realizan las representaciones de lesiones identificadas en el examen proctológico, cuenta con un único pincel para dibujar todas las lesiones y la representación depende de la habilidad que tenga el especialista para dibujar. Esto trae consigo representaciones inexactas, produciendo errores y demoras en el proceso de diagnóstico médico.

La representación de las lesiones se realiza sobre una imagen externa del área rectal, por lo que no es factible señalar una lesión en el interior del recto o en otro órgano interno. Se hace necesario que el especialista explique dónde se produjo la lesión y el tipo de lesión identificada, en el campo observaciones de la hoja de consulta de la especialidad de cirugía colorrectal.

Si un doctor de otra especialidad desea representar alguna lesión en la imagen de modelo de examen físico tiene que realizarlo de forma manual, está situación puede provocar los siguientes contratiempos:

1. No homogeneidad a la hora de representar las lesiones y tratamientos.

2. Almacenamiento de los modelos de examen físico en archivos de formato duro, al no contar el HIS con una manera digital de almacenarlos, esto puede provocar deterioro y pérdida de dichos modelos.

Por lo anteriormente planteado se formula como **problema a resolver**: ¿Cómo incrementar la exactitud de la representación de afecciones y tratamientos en imágenes de modelos de examen físico en el Sistema de Información Hospitalaria del CESIM?

Por tanto se define como **objeto de estudio:** el tratamiento de imágenes en sistemas web.

Todo lo anterior define como **campo de acción:** el proceso de gestión y representación de imágenes de modelos de examen físico en el Sistema de Información Hospitalaria del CESIM.

Para darle solución al problema científico propuesto se plantea el **objetivo general** siguiente: desarrollar un componente para incrementar la exactitud de la representación de imágenes de modelos de examen físico en el Sistema de Información Hospitalaria del CESIM.

Para dar cumplimento al objetivo general propuesto se definieron las siguientes **tareas de investigación**:

1. Evaluación de las tendencias actuales de los sistemas de información hospitalaria que cuenten con visualizaciones de imágenes de modelos de examen físico.
2. Especificación de los procedimientos para la implementación del componente para incrementar la exactitud de la representación de imágenes de modelos de examen físico en el Sistema de Información Hospitalaria del CESIM.
3. Asimilación de la arquitectura definida por el departamento de Sistemas de Información Hospitalaria para el desarrollo de sus aplicaciones.
4. Modelado de los artefactos correspondientes a los flujos de trabajo “Modelado del Negocio”, “Gestión de Requisitos”, “Diseño” e “Implementación”.
5. Implementación del componente para la creación, visualización y manipulación de imágenes de modelos de examen físico.

Con el desarrollo del componente se espera obtener los siguientes **beneficios**:

1. El especialista puede realizar una descripción más específica, lo que repercute en un diagnóstico más acertado que incrementa la calidad de vida del paciente.
2. El almacenamiento de la imagen en formato base64 posibilita que cualquier sistema pueda interpretar este resultado y visualizar esta imagen sin ningún contratiempo.

Este documento se encuentra estructurado en cuatro capítulos:

Capítulo 1: Fundamentación teórica: se realiza un estudio preliminar de sistemas encargados de la gestión de imágenes de modelos de examen físico**,** caracterizando las diferentes soluciones y efectuando un análisis comparativo. Se realiza un estudio sobre los conceptos asociados al dominio del problema y de las tecnologías, metodologías y herramientas de desarrollo a utilizar y se fundamentan las mismas.

Capítulo 2: Características del componente: se realiza el modelo de dominio, en el cual se definen los principales conceptos que se emplearán. Se puntualizan los requisitos funcionales y no funcionales que debe cumplir el sistema se realiza una descripción de las funcionalidades de la solución propuesta a la problemática planteada. Se describe la propuesta de solución para el desarrollo de un componente para la gestión de imágenes de modelos de examen físico.

Capítulo 3: Diseño de la solución propuesta: se realiza una descripción y análisis de la estructura de la solución que se propone para dar respuesta a la problemática planteada, así como fundamentar la arquitectura empleada.

Capítulo 4: Implementación: se introduce el flujo de trabajo de implementación, partiendo de los resultados obtenidos en el diseño y se exponen aspectos referentes a la seguridad del sistema, las estrategias de codificación, así como la forma en que se tratarán los errores.

Capítulo 1 Fundamentación teórica

En el presente capítulo se presentan los aspectos básicos de los conceptos principales, antecedentes de sistemas similares tanto en el ámbito nacional como internacional dando a conocer sus características fundamentales y se exponen las tecnologías y herramientas que se utilizarán en el desarrollo de la investigación.

1.1 Conceptos asociados al problema

Modelo de examen físico: es un modelo médico donde de señalan o describen la normalidad o alteraciones físicas producidas por enfermedades o por lesiones identificadas por el médico valiéndose solo de sus sentidos comunes e instrumentos sencillos (2).

Lesión: alteración de las características morfológicas o estructurales de un organismo en cualquiera de sus niveles de organización (molecular, celular, tisular, anatómico, corporal o social) producido por causas físicas químicas o biológicas (3).

Lienzo: término utilizado para definir el espacio de trabajo sobre el cual se ubica una imagen que posteriormente será retocada y manipulada por un software para la edición de imágenes (4)

Pincel: término utilizado para representar cualquiera de los botones de dibujo en una barra de herramientas en un software para la edición de imágenes (4).

1.2 Sistemas existentes

DDS GP (Doctor of Dental Surgery General Practitioner/Dentist)

Es una aplicación para *iPhone* e *iPad* que permite a los odontólogos mostrar a sus pacientes los tratamientos a realizar aprovechando la pantalla táctil con gestos simples como clic y arrastrar. Entre las situaciones que puede explicar el especialista se encuentran: severidad de una lesión, enfermedades periodontales que pueden conducir al uso de implantes dentales y roturas.

Esta aplicación permite una conexión directa entre el doctor y el paciente, logrando que el proceso llevado a cabo sea transparente. Incluye acerca de 200 demostraciones ilustradas, lo que hace de la misma una aplicación de competencia en el mercado mundial. A pesar de las características mencionadas es una aplicación con licencia privativa y su costo logra alcanzar como mínimo los

399.99 USD. Le fue otorgado en 2011 por revisores de *DentalShopper* el reconocimiento "Cinco Estrellas" como mejor producto de la esfera probado. Actualmente se encuentra en su versión 7.0.10 con un precio de 499.95. Trabaja bajo el sistema operativo *iOS*, utilizado en todos los productos desarrollados por la empresa *Apple*, Inc. (5).

Aquar Software

Es un aplicación informática dental fabricada por CLAS, Empresa de Soluciones de Gestión SL (Sociedad Limitada). La aplicación incluye todo lo necesario para un diagnóstico y tratamiento de los pacientes, así como una galería de imágenes que es consultada según el área de la institución que lo solicite. En las imágenes seleccionadas el especialista podrá dibujar como si lo tuviera en papel desde un *iPad*.

Cuenta con un sistema de edición de imágenes que permite adjuntar imágenes captadas por sistemas externos e interactuar con ellas para sus notas personales o mostrarlas al paciente de una forma muy atractiva. Controla además todos los tratamientos que realiza el doctor en su clínica de una manera rápida, sencilla y fiable, teniendo siempre visibles los tratamientos iniciales, propuestos, pendientes y terminados.

Es adaptable para cualquier área de una institución hospitalaria lo que hace que sea un producto altamente funcional. Posee un precio de 60 euros/módulo y posee 56 módulos. Trabaja directamente en equipos *Apple* con Sistema Operativo *iOS* por lo que su funcionamiento en Equipos Apple, *MacBook*, *Mac*, *iMac* y *iPad* es perfecto. Se encuentra actualmente en la versión 1.3. Fue creada en el 2009 aproximadamente (6).

DEN-TAL

Aplicación informática desarrollada por la Empresa *Pring Soft* de Argentina. Es un software de gestión diseñado específicamente para Consultorios Odontológicos. Permite editar y guardar un registro gráfico de las fichas dentales del paciente, consultar el estado inicial, tratamientos realizados y a realizar. También admite redactar, guardar y consultar rápidamente los antecedentes médicos de cada paciente, así como mantener un archivo de fotos dentales personalizado de cada paciente y acceder al mismo con facilidad y grandes ventajas.

Otras de las prestaciones con que cuenta el sistema es poder revisar y comentar gráficamente con sus pacientes las dolencias y/o tratamientos a realizar por medio de un programa de dibujo incorporado, y con imágenes de fondo previas. Es compatible con *Windows* 95/98/2000, actualmente pocas instituciones hospitalarias la usan debido a que trabaja solo en la Odontología y a las pocas funcionalidades que ofrece. El producto fue creado en el 2001 (7).

Simulador de Cirugía Plástica Virtual (SCPV)

Aplicación potente y sencilla que consiste en la simulación de cirugías plásticas con el objetivo de conocer un posible resultado de una cirugía antes que la misma sea llevada a cabo. Es una aplicación diseñada al igual que otras para que el doctor consulte con el paciente antes de realizar un procedimiento médico. La misma permite aplicar ciertas transformaciones a una fotografía con la intención de mostrar los cambios derivados de una posible intervención quirúrgica.

Una de las potencialidades de este producto es que su uso no se limita solamente al rostro del paciente, sino que puede ser usado para mostrar liposucción de caderas y glúteos, el aumento/disminución del pecho, rinoplastia, barbilla, labios, abdominoplastia y aumento de musculatura en el paciente. Si en el momento de la visita no se cuenta con una imagen del paciente, la aplicación permite que la misma pueda ser tomada usando la webcam del ordenador.

La aplicación es compatible con los sistemas operativos *Windows XP, VISTA, MAC, iOS y ANDROID*. Fue fabricada por *Kaeria EURL* compañía multimedia francesa radicada en París, creada en el 2004, con en el propósito de desarrollar de sitios web de alta tecnología y aplicaciones. La licencia de esta aplicación es privativa y su última versión conocida es la 1.3, lanzada en el 2012 (8).

Análisis crítico de los softwares en existencia vinculados al campo de acción

Luego de identificar los sistemas existentes encargados de gestionar la representación de lesiones y tratamiento de imágenes de modelos de examen físico se realizó el análisis posterior de los mismos.

A partir del estudio realizado sobre los sistemas encargados de gestionar la representación de lesiones y tratamiento de imágenes de modelos de examen físico, se encontró que los mismo no están desarrollados para cualquier especialidad de la medicina sino para una en específico, hecho que restringe el uso de los mismos en una institución hospitalaria. También, estos sistemas no funcionan como componente hecho que limita su integración con otros sistemas externos.

Otro inconveniente de los sistemas analizados es que no son multiplataforma, hecho que dificulta su uso en cualquier estación de trabajo. Por las razones expuestas se hace evidente la necesidad de crear un componente para la gestión de imágenes de modelos de examen físico para el HIS del CESIM.

1.3 Metodologías, Tecnologías, Herramientas y Lenguajes utilizados

Como requisito principal para que el componente para incrementar la exactitud de la representación de imágenes de modelos de examen físico forme parte del HIS del CESIM, se hace necesario asimilar la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones. De esta forma siguiendo la línea de desarrollo de dicho departamento, a continuación se describe la metodología, así como las tecnologías y herramientas empleadas.

1.3.1 Arquitectura del componente

Patrón de arquitectura Modelo-Vista-Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC, se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML *(HyperText Markup Language)* y el código que provee de datos dinámicos a la página. El modelo lo constituyen los datos con los que trabaja la aplicación y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. (9).

Descripción del patrón

Modelo**:** representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.

Vista**:** es la información presentada al usuario. Una vista puede ser una página web o una parte de ella.

Controlador**:** actúa como intermediario entre el modelo, la vista y cualquier otro recurso necesario para generar una página web. (10)

1.3.2 Metodologías de desarrollo

Proceso Unificado de Desarrollo (RUP)

Las siglas RUP en inglés significa *Rational Unified Process* (Proceso Unificado de Desarrollo) es un producto del proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo. Su meta es asegurar la producción del software de alta calidad que resuelve las necesidades de los usuarios dentro de un presupuesto y tiempo establecido. RUP es dirigido por casos de uso, centrado en arquitectura e iterativo incremental; además permite clasificar el desarrollo de software en fases (Comienzo o Inicio, Elaboración, Construcción, Transición) permitiendo comprobar los objetivos del desarrollo de software para cada una de las fases (11).

En RUP se han agrupado las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo, estos son: Modelamiento del Negocio, Requisitos, Análisis y Diseño, Implementación, Prueba o Testeo, Instalación, Administración de Configuración y Cambios y Ambiente. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente la generación de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software (11).

Lenguaje Unificado del Modelado (UML)

UML es un lenguaje de modelado visual que se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información generada en el desarrollo de software. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar (11).

1.3.3 Lenguaje de programación

Java es un lenguaje de programación desarrollado por la compañía Sun Microsystems, que utiliza el paradigma de la Programación Orientada a Objetos (POO). Es un lenguaje robusto pues no permite el manejo directo del hardware ni de la memoria. Dentro de sus principales ventajas se encuentra la de ser multiplataforma. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general.

Con Java se pueden programar aplicaciones web dinámicas, con acceso a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. Este lenguaje es utilizado de manera horizontal en el desarrollo del sistema, pues puede estar presente en las diferentes capas de la aplicación (12).

JavaScript

JavaScript es un lenguaje orientado a objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (13).

Cascading Style Sheets (CSS)

Las hojas de estilo en cascada son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (*World Wide Web Consortium*) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. Lo que se persigue con el desarrollo de CSS es separar la estructura de un documento de su presentación (14).

1.3.4 Tecnologías utilizadas

Java Server Faces

Java Server Faces (JSF) es un *framework* basado en el patrón MVC. Ofrece una clara separación entre el comportamiento y la presentación. Une los componentes de la interfaz de usuario (UI, por sus siglas en inglés) con los conceptos de la capa-web sin limitarse a una tecnología de script o lenguaje de marcas particular (15).

JSF brinda un conjunto de componentes de interfaz de usuario del lado del servidor para aplicaciones Web basadas en Java. Permite el manejo de estados y eventos, así como la asociación entre datos de la interfaz y los datos de la aplicación Web.

RichFaces

RichFaces es una librería de componentes visuales de código abierto la cual permite añadir capacidades Ajax a aplicaciones desarrolladas con JSF sin la utilización de *JavaScript*. Esta librería se integra perfectamente con *Seam*, también incluye ciclo de vida, validaciones, conversores y gestiona los recursos tanto estáticos como dinámicos (16).

Ajax4JSF

Ajax4jsf es una librería de código abierto que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax (acrónimo de *Asynchronous JavaScript* +XML) de forma limpia y sin añadir código *JavaScript*. Mediante este *framework* se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. En definitiva Ajax4jsf permite dotar a nuestra aplicación JSF de contenido mucho más profesional con muy poco esfuerzo (17).

Facelets

Facelets es un framework simplificado de presentación, en donde es posible diseñar de forma libre una página web y luego asociarle los componentes JSF específicos. Aporta mayor libertad al diseñador y mejora los informes de errores que tiene JSF. Tener un código ordenado y reutilizable (sobre todo en aplicaciones grandes), simplificación de desarrollo y facilidad en el mantenimiento son algunas de las ventajas que trae consigo su uso. Además, permite la definición de disposición de páginas basada en plantillas, la composición de componentes, creación de etiquetas personalizadas, desarrollo amigable para el diseñador gráfico y creación de librerías de componentes (18).

JBoss Seam

JBoss Seam 2.1 (Seam) es un *framework* para el desarrollo de aplicaciones web en *Java*, que define un modelo de componentes uniforme para toda la lógica de negocio de las aplicaciones que sean desarrolladas mediante su utilización. Integra fácilmente tecnologías estándares como Java Server

Faces (JSF), modelo de componentes para la capa de presentación; *Enterprise JavaBeans* (EJB3), modelo de componentes para la lógica de negocio y persistencia del lado del servidor; *Java Persistence* API (19).

Seam UI

Seam UI es una serie de controles JSF altamente integrables con *JBoss Seam*. Adicionan varias mejoras a JSF, desde validación, expresiones Extended EL e integración de la navegación en la interfaz de usuario (20).

1.3.4.1 Tecnologías horizontales

Java Runtime Environment (JRE 6)

El *Java Runtime Environment* (JRE) está destinado a desarrolladores de software y proveedores para la distribución de sus aplicaciones. Este contiene la máquina virtual Java, las bibliotecas de clases en tiempo de ejecución, y el lanzador de aplicaciones Java que son necesarios para ejecutar los programas escritos en este lenguaje de programación. Está compuesto además por las librerías de clases estándar que implementan las API de Java (21).

1.4 Herramientas

Visual Paradigm 8.0

Visual Paradigm para UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. VP-UML aporta a los desarrolladores de software una plataforma de desarrollo puntera para construir aplicaciones de calidad, mejores y más baratas con rapidez. Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE líderes del mercado (22).

Eclipse 3.4.2 Ganymede

Eclipse es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés), de código abierto y multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-Liviano" basadas en navegadores. Es una

potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java (23).

PostgreSQL 9.1

PostgreSQL es un Sistema Gestor de Base de Datos (SGBD) relacional orientado a objeto y libre, publicado bajo la licencia *Berkeley Software Distribution* (BSD). Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad, vistas, etc. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales que trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*) (24).

PgAdmin III 1.14.0

PgAdmin III es una aplicación gráfica para gestionar el SGBD PostgreSQL, siendo la más completa y popular con licencia de código abierto. Está escrita en C++. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados (25).

JBoss Server 4.2.2 GA

JBoss es un servidor de aplicaciones J2EE implementado en Java. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte (26). Sus principales características son:

- Producto de licencia de código abierto sin coste adicional
- Confiable a nivel de empresa
- Orientado a arquitectura de servicios
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Soporte completo para Java Management Extensions (JMX)

Conclusiones del Capítulo

En este capítulo se fundamentaron los resultados del estudio realizado sobre los sistemas para la gestión de imágenes de modelo de examen físico, donde se concluye que los mismos no representan la solución al problema a resolver, por lo que se hace necesario el desarrollo de un nuevo componente para la gestión de imágenes de modelos de examen físico.

Se realizó un estudio de las tecnologías y herramientas empleadas en el departamento Sistemas de Gestión Hospitalaria, lo que posibilitó conocer sus características, las cuales serán de mucha ayuda para el desarrollo del componente para la gestión de imágenes de modelo de examen físico y garantizan una completa integración con el HIS del CESIM.

Capítulo 2 Características del componente

En este capítulo se tiene como objetivo abordar los diferentes elementos que brindan la base teórica y conceptual para el desarrollo de las funcionalidades del componente. Debido a que no existe una definición clara de los procesos del negocio en el entorno en que se desplegará el componente, surge la necesidad de desarrollar un modelo de dominio que abarque las definiciones asociadas a la aplicación, así como las relaciones existentes entre ellas. Además, se enuncian los requisitos funcionales agrupados en los diagramas de casos de uso correspondientes.

2.1 Flujo actual del proceso de representación de lesiones identificadas en el examen proctológico

La representación de lesiones en imágenes de modelos de examen físico que se realiza en la especialidad de cirugía colorrectal cuenta con un único pincel y una sola imagen. La funcionalidad de dibujo tiene además una leyenda con seis lesiones que se pueden representar en la imagen y cuatro colores para diferenciar las representaciones.

Para realizar el proceso de dibujo, el especialista se fija en la lesión y mira si esta se puede representar con la leyenda que existe. En el caso de que pueda representar la lesión, realiza el dibujo sobre la imagen en el lugar determinado, en caso contrario no puede representar esa lesión y lo que puede hacer es escribir en las observaciones el lugar y el tipo de lesión encontrada.

2.2 Modelo de dominio

Modelo de dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases conceptuales y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema (27).

2.2.1 Conceptos fundamentales del dominio

Con la finalidad de una mejor comprensión del Diagrama del Modelo de Dominio a continuación se dará una breve descripción de los conceptos encontrados en el ámbito del problema.

Configuración: es el que contiene las preferencias del especialista.

Pincel: forma que toma el puntero para dibujar en la imagen.

Lienzo: marco de trabajo que contiene una imagen.

Especialista: persona encargada de interactuar con el sistema.

Operación: son las acciones que el especialista realizará en el componente de dibujo (Ejemplo: dibujar lesión, seleccionar configuración, entre otras).

2.2.2 Diagrama del Modelo de Dominio

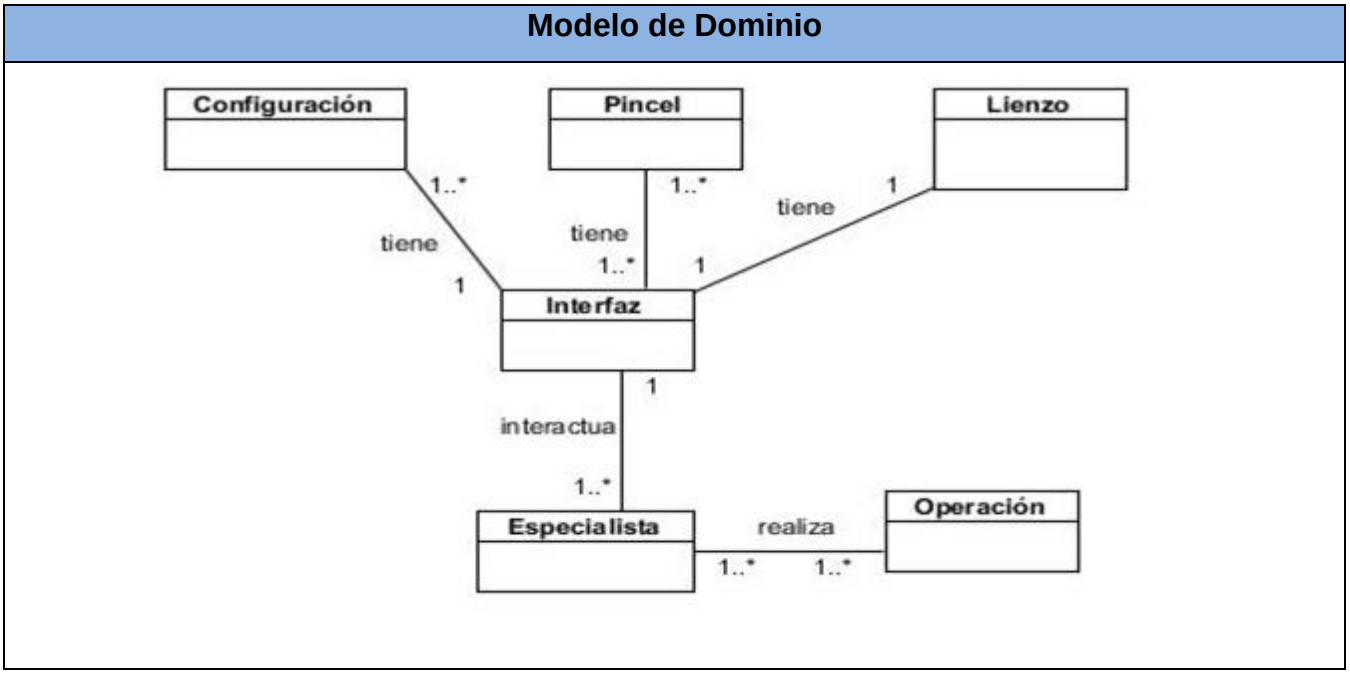

Figura 1: Diagrama Modelo de Dominio

2.3 Propuesta de solución

La solución propuesta estará compuesta por una interfaz principal, la cual contará con una configuración formada por un conjunto de pinceles. Los pinceles van a ser usados sobre una imagen cargada en un lienzo mediante operaciones predefinidas. Las operaciones serán el conjunto de cambios o procedimientos que se van a aplicar sobre la imagen, dígase de forma general Representar una lesión o Representar un tratamiento. Las operaciones antes mencionadas van a ser

llevadas a cabo por un especialista autenticado en el sistema y con la posibilidad de acceder al componente.

Dicho componente permitirá al especialista crear sus propias configuraciones, las mismas van a poseer un nombre, una descripción, una especialidad y el nombre del especialista que la crea. La solución brindará la posibilidad de seleccionar la cantidad de pinceles con los que va a trabajar sobre la imagen. La selección de los pinceles es a conveniencia, lo cual proveerá al especialista de cierta libertad a la hora de crear su propia configuración.

La configuración escogida para trabajar va a ser generada dinámicamente en el área de trabajo, esto permite combinar los pinceles asociados a la misma con los pinceles básicos del componente. La combinación de estos grupos de pinceles permite una representación más detallada de las lesiones y logra de cierta forma un estándar de trabajo para todos los especialistas.

Además de poder combinar pinceles y funciones para la presentación, el componente será capaz de generar varias imágenes de varias especialidades, haciendo su uso más extensible, a diferencia de la mayoría de los sistemas mencionados anteriormente que limitan su uso a una sola especialidad. Es necesario decir, que sin importar la cantidad de especialistas que usen el componente simultáneamente, el mismo será capaz de enviar una imagen resultante a cada uno y sin confundir las peticiones de envíos.

La solución propuesta será diseñada para retornar la imagen resultante en forma de texto en base64, esto permite que cualquier módulo en el sistema que establezca comunicación con el mismo obtenga una cadena de *string* resultante y la convierta en una imagen para ser mostrada. Una vez que la transacción haya sido completada, el uso del resultado ya no depende del componente, sino del propósito con que fue creada la imagen para cada caso.

2.3.1 Especificación de los requisitos de software

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, conformando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional (28).

2.3.2 Requisitos funcionales del componente

RF1 Crear configuración

- Permite crear una configuración.

RF2 Ver detalles de la configuración

- Permite conocer los datos de una configuración luego de haber sido creada.

RF3 Modificar configuración

- Permite modificar una configuración seleccionada.

RF4 Eliminar configuración

- Permite eliminar una configuración ya existente.

RF5 Buscar configuración

- Permite buscar una configuración en el componente, en caso de que sea necesario conocer a través de un criterio, si existe una configuración.

RF6 Seleccionar pinceles

- Permite seleccionar los pinceles de una cantidad existente.

RF7 Ver datos de configuración

- Permite conocer los datos de la configuración seleccionada.

RF8 Crear pincel

- Permite crear un nuevo pincel no existente en el componente.

RF9 Ver detalles del pincel

- Permite conocer los datos de un pincel luego de ser creado

RF10 Eliminar pincel

- Permite eliminar un pincel ya existente.

RF11 Buscar pincel.

- Permite buscar un pincel en el componente, en caso de que sea necesario conocer a través de un criterio, si existe un pincel.

RF12 Modificar pincel

- Permite modificar un pincel seleccionado.

RF13 Cargar imagen

- Permite seleccionar una imagen de un directorio físico.

RF14 Exportar imagen

- Permite exportar la imagen a un directorio físico.

RF15 Dibujar lesión.

- Permite representar las lesiones identificadas.

RF16 Dibujar tratamiento

- Permite representar un posible tratamiento en el área de la región donde se identificó la lesión.

2.3.3 Requisitos no funcionales del componente

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos son fundamentales para el éxito del mismo. Normalmente están vinculados a los requisitos funcionales, es decir, una vez se conozca lo que el sistema debe hacer se puede determinar cómo debe comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Son importantes para que clientes y usuarios tengan un criterio común en cuanto a la valoración de las características no funcionales del producto, pues si se tiene conocimiento de que cumple con todas las funcionalidades requeridas, estas propiedades pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación (28).

En el departamento de Sistemas de Gestión Hospitalaria se han definido requisitos no funcionales para el desarrollo de componentes para el HIS del CESIM. En la presente investigación se han adoptado los siguientes requisitos no funcionales siguientes:

Usabilidad

El componente estará diseñado de manera que los usuarios adquieran las habilidades necesarias para poder utilizarlo en un tiempo reducido, si este ha interactuado anteriormente con el HIS del CESIM el tiempo que necesita para aprender a usar el componente no debe pasar los tres días. En el caso de que el usuario nunca haya interactuado con el sistema su capacitación para interactuar con el sistema no debe sobrepasar los 5 días.

Seguridad

La seguridad es un tema de gran importancia para cualquier Sistema de Información y toma mayor relevancia cuando se gestiona información médica. La responsabilidad de mantener la seguridad cae en quien use el componente, este debe controlar quien puede acceder o no a las funcionalidades que este brinda. También debe comprobar quién puede consultar las imágenes resultantes después de la realización de la representación lesiones o un posible tratamiento.

Hardware

Estaciones de trabajo.

En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria, las que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y siga los estándares web. Se escogieron estaciones de trabajo de 512 MB de memoria RAM y un microprocesador de 2.0 Hz con Sistema operativo GNU/Linux.

Servidores

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

- Servidores de Base de Datos**:** servidor con cualquier tecnología o sistema operativo que soporte a PostgreSQL Server 8.4 o superior en los servidores de base de datos de cada hospital, y Oracle 10g o superior para los servidores de base de datos del Centro de Datos, 2GB de memoria y 1 TB de disco duro**.**

- Servidores de Aplicaciones: 16 GB de memoria y 1 TB de disco duro con tecnología sistema operativo que soporte el Java Runtime Environment (JRE) 1.6.0_24 o superior y al JBoss AS 4.2.2.GA.

Software

El componente debe correr en los siguientes Sistemas Operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java virtual machine, JBoss AS y PostgreSql). Los clientes deberán disponer de un navegador web, estos pueden ser IE 7 o superior, Opera 9 superior, Google Chrome 1 o superior y Firefox 4.X o superior y tener habilitado el javascript.

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 los 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 *(Java Persistance API)* y particularmente en la implementación del motor de persistencia *Hibernate*.

Interfaz de usuario

Las ventanas del componente tendrán una interfaz amigable capaz de permitir la interpretación correcta de la información. La interfaz contará con accesos directos y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada e informada al usuario. Todos los textos y mensajes en pantalla aparecerán en idioma español.

Portabilidad

El producto podrá ser utilizado bajo los sistemas operativos Linux o Windows ya que son los sistemas operativos más comunes. La aplicación podrá ser desplegada sobre Sistemas Operativos Linux (Linux 5.5+, 6.x, Ubuntu 8.04 LTS (Long Term Support) Desktop Edition, Ubuntu Linux 10.04 y superior, entre otras versiones) y Windows (2000, XP, Server 2003, Server 2008, Server 2012, Vista, 7, 8).

2.3.4 Modelo de casos de uso del componente

El modelo de casos de uso del componente documenta el comportamiento del mismo desde el punto de vista del usuario, permitiendo representar las funciones que se desean en el sistema (casos de uso), el entorno del sistema (actores), y las relaciones entre ellos. Aunque la parte más visible de dicho modelo son los diagramas de casos de uso, suele ir acompañado de una especificación textual de cada uno de los casos de uso (29).

Los actores del componente no forman parte del mismo, sino que representan elementos que interactúan con él. Estos elementos son nombrados roles que pueden ser desempeñados por una o varias personas, un equipo o un sistema automatizado. Un actor puede introducir o recibir información del sistema (29).

Actor	Descripción
Especialista	El especialista es el encargado de realizar las operaciones sobre las imágenes de modelos de exámenes físicos, así como gestionar las configuraciones existentes en el componente.

Tabla 1: Definición de los actores del componente

2.3.4.1 Definición de los actores del componente

Diagrama de actores


Figura 2: Diagrama de actores

2.3.4.2 Diagrama de Caso de Uso del Componente

Diagrama de Caso de Uso del Componente

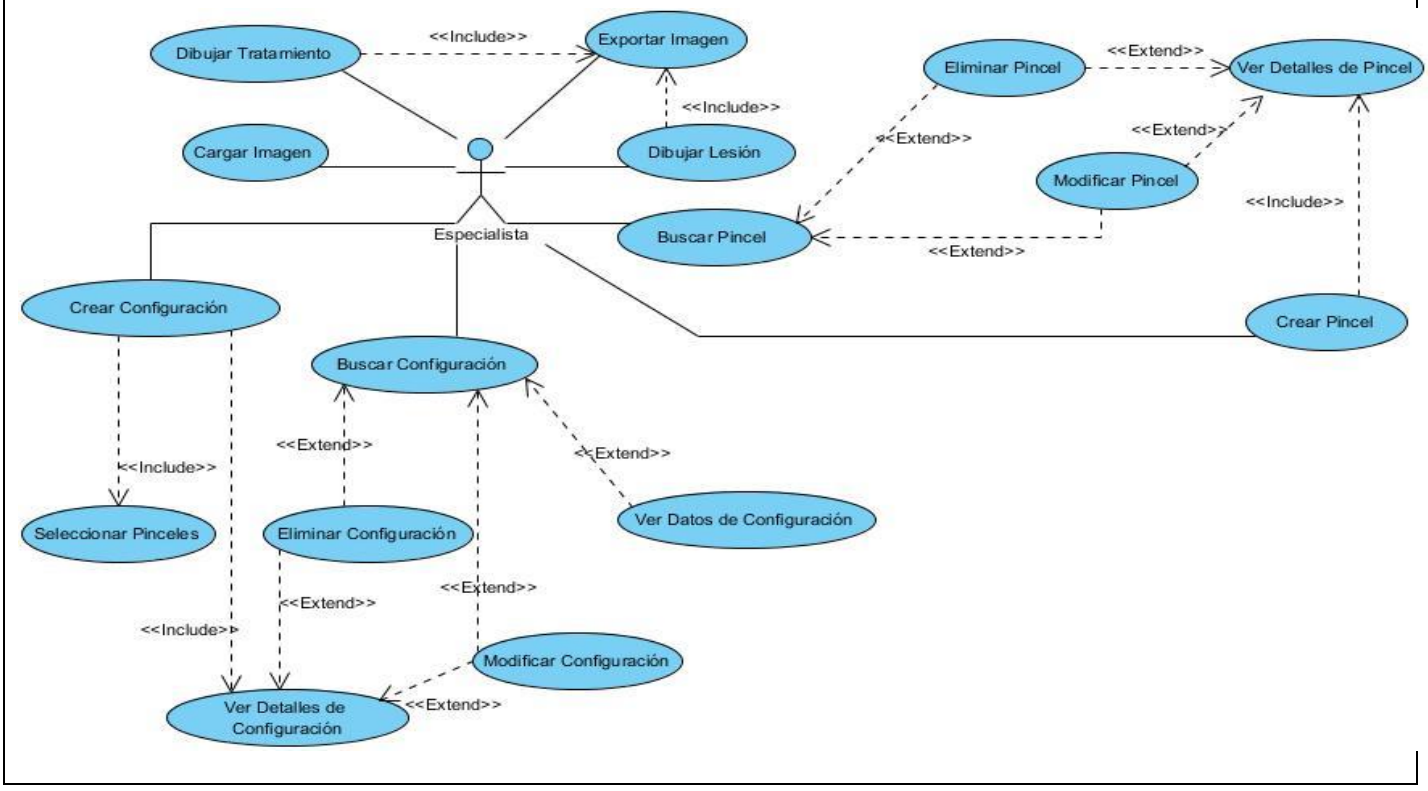

Figura 3: Diagrama de caso de uso del componente

2.3.4.3 Descripción textual de los casos de uso

CASO DE USO:	Crear Configuración
Resumen:	El caso de uso inicia cuando el actor accede a la opción Crear Configuración. El componente brinda la posibilidad de introducir los datos para crear la Configuración. El actor introduce los datos. El componente crea la nueva Configuración. El caso de uso termina.
Complejidad:	Media

Prioridad:	Media
Precondiciones:	No tiene
Especialista	Especialista
RF1	**RF1**
Configuración	**Configuración**

Tabla 2: CU Crear Configuración

CASO DE USO:	Ver detalles de la Configuración.
Resumen:	El caso de uso inicia después que el actor crea una Configuración. El componente muestra los datos de la Configuración creada. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media
Precondiciones:	Para ver los detalles de una configuración, esta debe estar creada.
Actores:	**Especialista**
Requisitos:	**RF2**
Entidades:	**Configuración**

Tabla 3: CU Ver detalles de la Configuración

CASO DE USO:	Modificar Configuración
Resumen:	El caso de uso inicia cuando el actor accede a la pantalla de configuraciones. Selecciona la opción Modificar Configuración. El componente muestra los datos de la configuración y brinda la posibilidad de cambiarlos. El componente actualiza los datos de la configuración. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media
Precondiciones:	Para modificar los datos de una configuración esta debe existir en el componente.
Actores:	**Especialista**
Requisitos:	**RF3**
Entidades:	**Configuración**

Tabla 4: CU Modificar Configuración

CASO DE USO:	Eliminar Configuración
Resumen:	El caso de uso inicia cuando el actor selecciona la opción Eliminar Configuración. El componente elimina la Configuración. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media

Precondiciones:	Tiene que estar creada una configuración en el componente.
Actores:	**Especialista**
Requisitos:	**RF4**
Entidades:	**Configuración**

Tabla 5: CU Eliminar Configuración

CASO DE USO:	Buscar Configuración
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar Configuración. El componente brinda la posibilidad de introducir criterios de búsqueda para localizar la Configuración que concuerde con el criterio. El actor introduce los datos para realizar la búsqueda. El componente busca y lista las configuraciones que cumplan con los criterios de búsqueda introducidos. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media
Precondiciones:	No tiene
Actores:	**Especialista**
Requisitos:	**RF5**
Entidades:	**Configuración**

Tabla 6: CU Buscar Configuración

CASO DE USO:	Ver Datos de la Configuración
Resumen:	El caso de uso inicia cuando el actor selecciona Ver Datos de la Configuración. El sistema muestra los datos del Pincel. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media
Precondiciones:	Para ver los datos de la configuración tiene que seleccionarla.
Actores:	Especialista
Requisitos:	**RF7**
Entidades:	**Configuración**

Tabla 6: CU Ver Datos de la Configuración

CASO DE USO:	Ver Detalles de Pincel
Resumen:	El caso de uso inicia después que el actor Crea un Pincel. El componente muestra los datos del pincel creado. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media
Precondiciones:	Para ver los detalles de un pincel, este debe ser creado.
Actores:	Especialista
Requisitos:	**RF9**

Entidades:	**Pincel**

Tabla 8: CU Ver Detalles de Pincel

CASO DE USO:	Eliminar Pincel
Resumen:	El caso de uso inicia cuando el actor selecciona un Pincel y accede a la opción Eliminar Pincel. El componente elimina el Pincel. El caso de uso termina.
Complejidad:	Media
Prioridad:	Media
Precondiciones:	Para eliminar un Pincel, este debe haber sido seleccionado.
Actores:	Especialista
Requisitos:	**RF10**
Entidades:	Pincel

Tabla 9: CU Eliminar Pincel

CASO DE USO:	Buscar Pincel
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar Pincel, el componente brinda la posibilidad de introducir criterios de búsqueda para localizar el Pincel. El actor introduce los datos que considera como criterios para realizar una búsqueda. El sistema busca y muestra los Pinceles que cumplen con los criterios de búsqueda. El caso de uso termina.
Complejidad:	Media

Prioridad:	Media
Precondiciones:	No tiene
Actores:	**Especialista**
Requisitos:	**RF11**
Entidades:	**Pincel**

Tabla 10: CU Buscar Pincel

CASO DE USO:	Cargar Imagen
Resumen:	El caso de uso inicia cuando el actor accede a la opción Cargar Imagen. El componente brinda la posibilidad de que pueda buscar la imagen que desea y cargarla al sistema. El caso de uso termina.
Complejidad:	Alta
Prioridad:	Alta
Precondiciones:	No tiene
Actores:	**Especialista**
Requisitos:	**RF13**
Entidades:	**No tiene**

Tabla 11: CU Cargar Imagen

CASO DE USO:	Dibujar Lesión
Resumen:	El caso de uso inicia cuando el actor accede a la opción Dibujar Lesión. Luego de activarla el componente brinda la posibilidad de que pueda representar las lesiones identificadas en el examen físico. Termina el caso de uso.
Complejidad:	**Alta**
Prioridad:	**Alta**
Precondiciones:	Tiene que estar cargada una imagen sobre el lienzo.
Actores:	**Especialista**
Requisitos:	**RF15**
Entidades:	**No tiene**

Tabla 12: CU Dibujar Lesión

CASO DE USO:	Dibujar Tratamiento
Resumen:	El caso de uso inicia cuando el actor accede a la opción Dibujar Tratamiento, luego de activarla el componente brinda la posibilidad de que pueda representar en el área de la lesión un posible tratamiento. Termina el caso de uso.
Complejidad:	**Alta**
Prioridad:	**Alta**
Precondiciones:	Tiene que estar cargada una imagen sobre el lienzo.

Actores:	**Especialista**
Requisitos:	**RF16**
Entidades:	**No tiene**

Tabla 13: CU Dibujar Tratamiento

Conclusiones del Capítulo

En el presente capítulo se identificó el actor que interviene y se realizó el diseño del diagrama de casos de uso del sistema, logrando una representación detallada de cada proceso. A partir de los requerimientos funcionales fueron definidas las funcionalidades a implementar, posibilitando un mejor entendimiento de la solución propuesta.

Mediante el Proceso Unificado de Desarrollo, se obtuvieron los artefactos correspondientes al modelo de domino para representar los flujos existentes y diagrama de casos de uso del sistema para describir el comportamiento del sistema desde el punto de vista del usuario y representar las funcionalidades con que contará el componente y que sirva de guía a los desarrolladores.

Capítulo 3 Diseño de la solución propuesta

En este capítulo se realiza una descripción detallada de la solución propuesta y se explica la arquitectura de la misma. También se describe el uso de los patrones de diseño, se transforman los requisitos funcionales en un diseño de clases, donde la relación que se establece entre estas clases describe el funcionamiento de los casos de uso del sistema.

3.1 Descripción de la arquitectura

La arquitectura de software es un conjunto de patrones que sirven de guía para la elaboración de un sistema. 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.

El componente propuesto presenta como parte de la línea base de su arquitectura el patrón Modelo Vista Controlador, el cual separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos: el modelo para la administración de los datos; la vista que muestra las interfaces de usuario y el controlador para la gestión de las acciones del usuario, invocando cambios tanto en la vista como en el modelo.

En el patrón arquitectónico utilizado, el MVC, JSF ofrece una clara separación entre el comportamiento y la presentación. Une los familiares componentes UI con los conceptos de la capa-Web sin limitarnos a una tecnología de script o lenguaje de marcas particular.

Los *beans* a los que se vinculan los formularios JSF son el modelo. También contienen acciones, que son una extensión de la capa del controlador y delegan las peticiones del usuario a la capa de la lógica de negocio. Las páginas *Facelets* con etiquetas JSF personalizadas son la capa de la vista. El *Servlet Faces* proporciona la funcionalidad del controlador.

Se definen en el Modelo las clases entidades que se manejan en la solución, aquí se pueden encontrar las clases Pincel y Configuración. Para el trabajo en el modelo se utiliza *Hibernate* como herramienta de Mapeo objeto-relacional (ORM), el cual es la implementación para EJB y el API para la persistencia de java o JPA.

En el caso de la Vista se encuentran las interfaces para interactuar con el usuario final, en ella se utilizó JSF porque permite la creación de interfaces de usuarios para aplicaciones web, se usan componentes *Seam* de interfaz de usuario y las bibliotecas de componentes *Richfaces* y *Ajax4jsf*. También se utilizó *JavaScript* para el trabajo en las interfaces con las imágenes y eventos.

Para el controlador se utiliza *JBossSeam* como *Framework* de integración entre los componentes visuales y los de acceso a datos.

3.2 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados

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 o en sistemas externos que incluyan la solución que se desea desarrollar. Esto reduce los costes de diseño, codificación y comprobación (30).

El Sistema de Información Hospitalaria del CESIM, 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 realizado a las funcionalidades encargadas de la representación de lesiones en imágenes de modelos de examen físico.

Se identificó que en módulo de consulta externa existe una funcionalidad llamada Representar afecciones identificadas en examen proctológico, en la que se dibujan las afecciones identificadas en este tipo de examen. Dicha funcionalidad se reutilizará para realizar la representación de lesiones y posibles tratamientos en las imágenes de dicho modelo.

3.3 Estrategias de integración

Con el propósito de lograr una integración completa del componente a todos los módulos del Sistema de Información Hospitalaria del CESIM se utilizan componentes que son de uso común para cualquier módulo que lo integran, entre los que se encuentran: las clases *ActiveModule* con el cual se puede conocer en que módulo está trabajando el usuario que interactúa con el sistema y la clase

Especialista para conocer las especialidades del usuario que está trabajando en el sistema y así mostrarle si tiene alguna configuración creada o las que coincidan con su especialidad.

El código que se genera para la realización de las funcionalidades fue pensado y adaptado a la estructura que hoy en día tienen las interfaces del sistema, para que al integrarse el componente no se alteran las formas en que se visualiza y se maneja la información. Las interfaces de componente utilizan ficheros JavaScript correspondientes a las funcionalidades asociadas al dibujo haciendo una referencia a la ubicación de estos.

3.4 Modelo de diseño

El Diseño es imprescindible para materializar con exactitud las exigencias del cliente, lo que conlleva que se realice con buena calidad el proyecto. El proceso de diseño es una secuencia de pasos que permiten al diseñador describir todos los aspectos del componente a construir. Esta fase es la que debe suministrar completamente la idea de lo que es el software, enfilando los dominios de datos y el comportamiento desde el punto de vista de la Implementación (31).

Por otra parte se encarga de modelar el sistema para que sustente todos los requisitos funcionales y no funcionales. En este modelo, los casos de uso son ejecutados por las clases del diseño, mediante los cuales se estructura el diagrama de clases del diseño. Este expone un conjunto de interfaces, colaboraciones y relaciones entre las clases definidas para dar solución a cada funcionalidad identificada, además detalla la estructura de clases del sistema y las relaciones entre las mismas (31).

3.4.1 Patrones de diseño

Los patrones de diseño representan un esquema de las soluciones a problemas en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos comunes. Se debe tener presente los siguientes elementos de un patrón: nombre, problema (cuando aplicar un patrón), solución (descripción abstracta del problema) y consecuencias (costos y beneficios) (32).

En el diseño del componente se utilizaron los siguientes patrones para dar solución a las diferentes problemáticas.

3.4.1.1 Patrón Experto o Experto en información

Problema: las responsabilidades deben ser acordes a la información que tiene cada clase.

Solución: para esto se aplica el patrón GRASP, Experto.

Aplicación: la clase AreaTrabajo es la experta en la información, porque contiene los datos para construir el área de trabajo y la lista de pinceles para crear la paleta de trabajo.

3.4.1.2 Patrón Creador

Problema: crea instancias solamente de objetos que contiene.

Solución: para esto se aplica el patrón GRASP, Creador.

Aplicación: la clase Configuración, tiene la responsabilidad de crear objetos de tipo Pincel porque contiene y realiza operaciones sobre los mismos.

3.4.1.3 Patrón Bajo acoplamiento

Problema: disminuir la dependía entres las clases.

Solución: para esto se aplica el patrón GRASP, Bajo acoplamiento

Aplicación: relación directa entre las clases del componente, el ejemplo de esto es la relación entre las clases Pincel y Configuración.

3.4.1.4 Patrón Alta cohesión

Problema: disminuir la dependencia entres las clases.

Solución: para esto se aplica el patrón GRASP, Bajo acoplamiento

Aplicación: relación directa entre las clases del componente, el ejemplo de esto es la relación entre las clases Pincel y Configuración.

3.4.2 Diagrama de Paquetes

Diagrama de paquetes: ofrece un nivel de abstracción que permite apreciar cómo se divide el sistema en ciertas agrupaciones de elementos que se les puede denominar paquetes. Este diagrama puede ayudar a repartir entre los desarrolladores la implementación por paquetes (28).

Para la elaboración del modelo de diseño, se define una estructura de paquetes que permite dividir el sistema en fragmentos manejables para su futura implementación. Se emplea el criterio de empaquetamiento por proceso, siguiendo la estructura de procesos definidos en el sistema. Los paquetes son graficados mostrando la relación que guardan entre sí. Estos utilizan el paquete repositorio de clases para su funcionamiento.

Un paquete referente a procesos, está conformado por subpaquetes que responden a las realizaciones de casos de uso, donde cada una de ellas contiene un diagrama de clases del diseño y los respectivos diagramas de secuencia.

El paquete repositorio de clases está compuesto por 3 subpaquetes, entidades, sesiones y vistas. El subpaquete de las entidades contiene las clases autogeneradas especificadas en el diseño, en dependencia de las tecnologías que serán usadas en la implementación y las personalizadas. Estas clases se autogeneran desde la base de datos utilizando el ORM Hibernate (generación de objetos java desde la base de datos). Las clases personalizadas son aquellas que se modifican para una mejor gestión de la información, por lo que pueden heredar de las autogeneradas. El de las sesiones por su parte estará conformado por las clases controladoras autogeneradas por el entorno de desarrollo, por las clases controladoras personalizadas y por las controladoras del proceso.

El paquete de las vistas en cambio estará compuesto por contenidos web referentes a las páginas clientes y los formularios que las componen, además de contener las vistas que interactúan con el usuario. Asociados a este paquete están 2 paquetes, uno es el de los ficheros JavaScript y el otro el de los ficheros CSS necesarios para el trabajo en el mismo.

El paquete gestionar pinceles contiene varios subpaquetes que responden a la realización de los casos de uso asociados a la gestión de los pinceles en el componente. El paquete gestionar configuración contiene un conjunto de subpaquetes que responden a la realización de los casos de uso asociados a la gestión de las configuraciones en el componente. Existen también dentro del diagrama de paquetes cuatro paquetes que responden a los casos de uso para el trabajo con las imágenes de modelos de examen físico en el componente, ellas son: salvar imagen, exportar imagen, dibujar lesión y dibujar tratamiento.

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

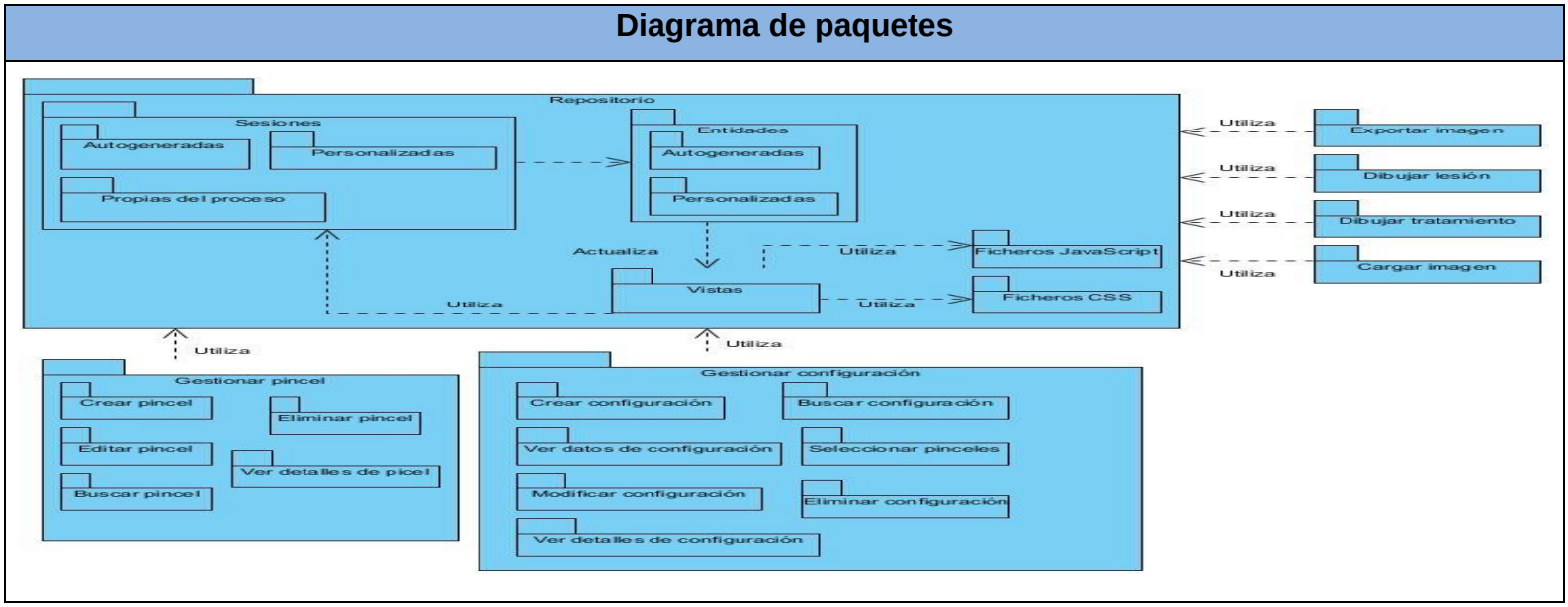

Figura 4: Diagrama de paquetes

3.4.3 Diagramas de clases de diseño

Diagrama de clases del diseño**:** constituyen un elemento fundamental en la concepción de la aplicación que se propone, ya que servirán de guía a los desarrolladores al constituir una aproximación del sistema que se desea implementar, contribuyendo de esta forma a la calidad del producto final (31).

Los Diagramas de Clases del Diseño se realizan mediante estereotipos Web que son una representación gráfica de los componentes a los cuales se hace referencia, se emplean para ello relaciones entre páginas y clases. En general muestran el flujo como una página servidora que construye una página cliente que a su vez contiene un formulario, el cual puede actualizar directamente a las entidades o enviar las peticiones a la página servidora, estas invocan métodos y responsabilidades en las clases controladoras, las cuales a su vez pueden modificar las entidades.

A continuación se muestran las Clases de Diseño de la aplicación propuesta:

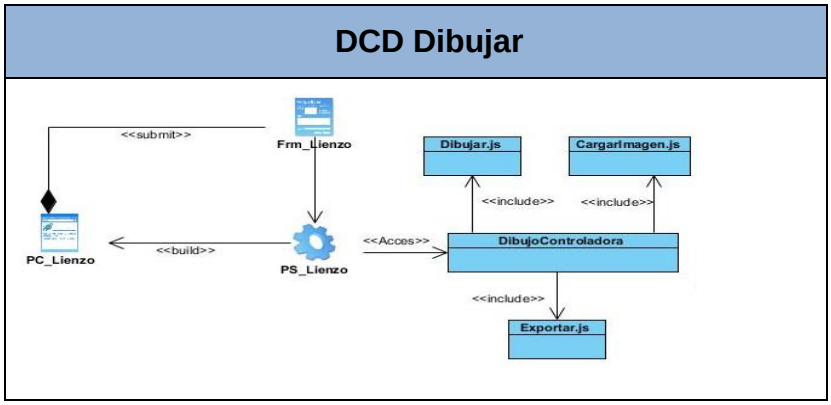

Figura 5: DCD Dibujar

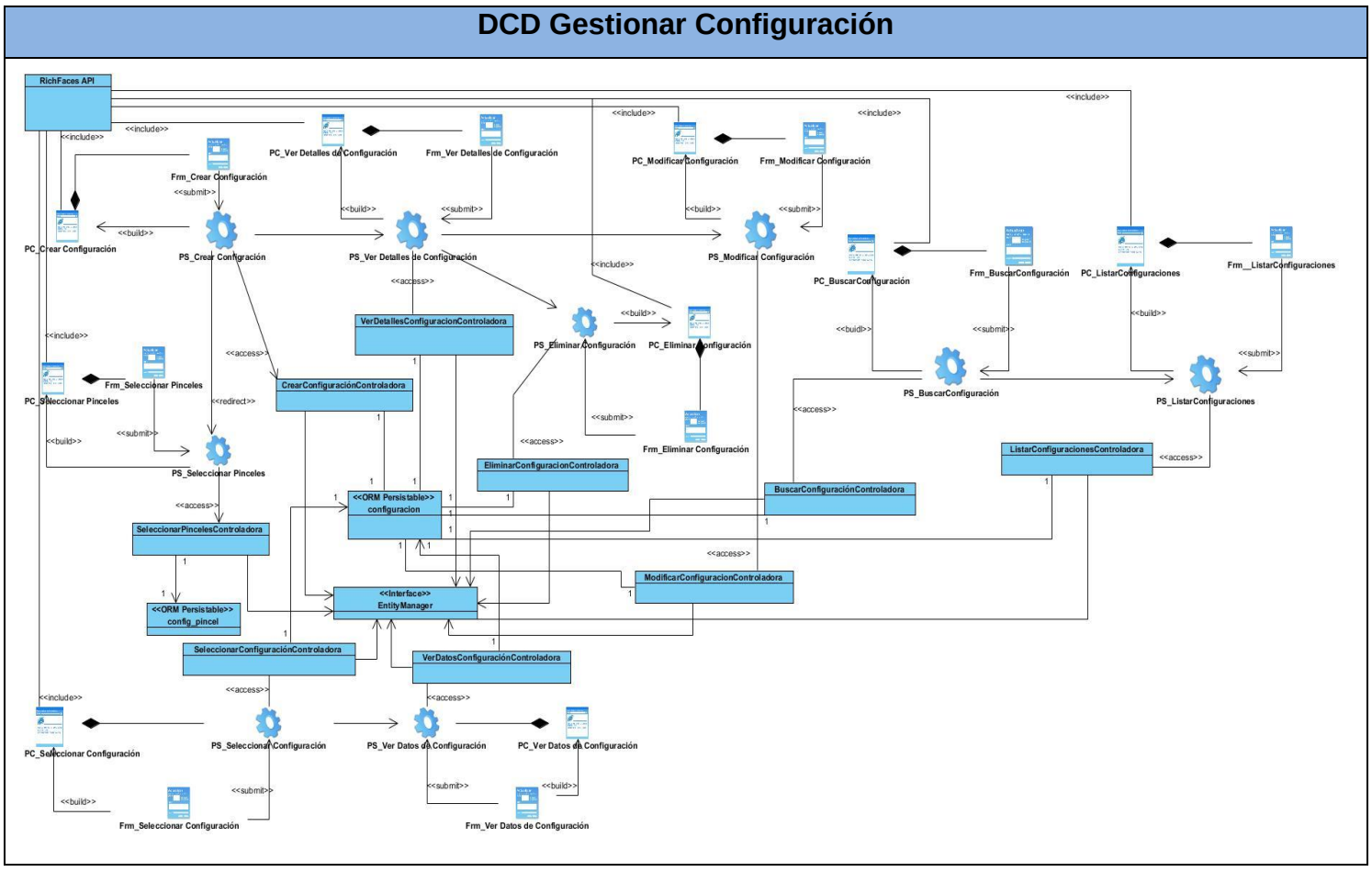

Figura 6: DCD Gestionar Configuración

DCD Gestionar Pincel

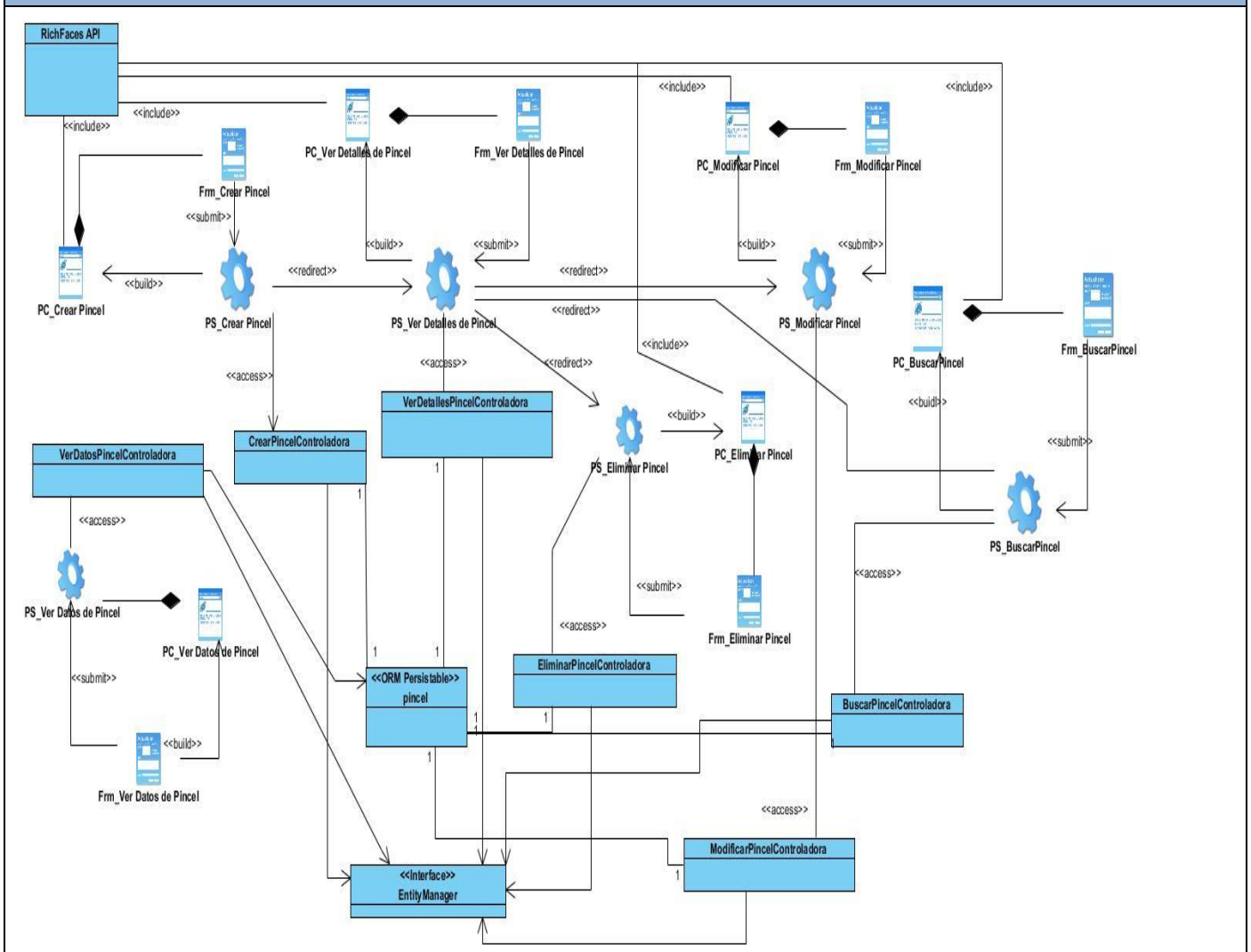

Figura 7: DCD Gestionar Pincel

Descripción de Clases

Nombre: CrearConfiguracion	
Tipo de clase: Controladora.	
Atributo	**Tipo**
nombre	String
especialidad	String
nombreEspecialista	String
descripcion	String
relacionMany	List<ConfiguracionPincel_exfisico>
pincelesSeleccionados	List<Pincel_exfisico>
confEnUso	Configuracion_exfisico
pincelSelect	HashTable<Long, Pincel_exfisico>
Funcionalidades	
Nombre:	estaSeleccionado(idParam : Long): boolean
Descripción:	Es el método encargado de recorrer la lista de pinceles seleccionados y retornar un valor booleano para notificar si este pincel ha sido seleccionado previamente o no, usando como criterio de búsqueda su id de tipo *Long*. El valor *false* en caso que no y el valor *true* en caso que sí.
Nombre:	seleccionarPincel(pincel Pincel_exfisico): void

Descripción:	Se encarga de agregar un pincel a la lista de pinceles en caso que el mismo no se encuentre en la lista de los pinceles seleccionados. Si el pincel ya se encuentra seleccionado entonces lo elimina de la lista de los pinceles seleccionados.
Nombre:	eliminarPincel(pincel Pincel_exfisico) : void
Descripción:	Se encarga de eliminar un pincel de la lista de pinceles que serán incluidos en la configuración a crear teniendo en cuenta un parámetro de tipo *Pincel_exfisico* para realizar la comparación.
Nombre:	persistirConfiguracion() : void
Descripción:	Se encarga de crear la configuración para posteriormente ser usada en el área de trabajo.

Tabla 14: Descripción de la clase controladora CrearConfiguración

Nombre: ModificarConfiguracion	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Nombre	String
Especialidad	String
nombreEspecialista	String
Descripción	String
relacionMany	List<ConfiguracionPincel_exfisico>

pincelesSeleccionados	List<Pincel_exfisico>
confEnUso	Configuracion_exfisico
pincelSelect	HashTable<Long, Pincel_exfisico>
loaded	Boolean
Funcionalidades	
Nombre:	estaSeleccionado(idParam : Long) : boolean
Descripción:	Es el método encargado de recorrer la lista de pinceles seleccionados y retornar un valor booleano para notificar si este pincel ha sido seleccionado previamente o no, usando como criterio de búsqueda su id de tipo *Long*. El valor *false* en caso que no y el valor *true* en caso que sí.
Nombre:	seleccionarPincel(pincel Pincel_exfisico) : void
Descripción:	Se encarga de agregar un pincel a la lista de pinceles en caso que el mismo no se encuentre en la lista de los pinceles seleccionados. Si el pincel ya se encuentra seleccionado entonces lo elimina de la lista de los pinceles seleccionados.
Nombre:	eliminarPincel(pincel Pincel_exfisico) : void
Descripción:	Se encarga de eliminar un pincel de la lista de pinceles que serán incluidos en la configuración a crear teniendo en cuenta un parámetro de tipo *Pincel_exfisico* para realizar la comparación.
Nombre:	persistirConfiguracion() : void
Descripción:	Se encarga de crear la configuración para posteriormente ser usada en el área

	de trabajo.

Tabla 15: Descripción de la clase controladora ModificarConfiguración

Nombre: componente.js	
Tipo de clase: Fichero JavaScript.	
Variables Generales:	
canvas	Esta variable es la encargada hacer referencia a la etiqueta <*canvas*> que se encuentra en la página visual, tomando como valores todos los atributos pertenecientes a este componente.
contexto	Es una variable creada con el fin de controlar y llevar a cabo todas las operaciones realizadas sobre el componente *canvas*. La importancia de esta variable radica en la selección de los enfoques dimensionales puesto que permite el renderizado 2D y 3D sobre el elemento *canvas*.
posx	Variable que contiene la coordenada x del cursor en el lugar que se encuentra en la pantalla.
posy	Variable que contiene la coordenada y del cursor en el lugar que se encuentra en la pantalla.
colorEnUso	El uso de colores en el trabajo con el componente *canvas* es esencial por lo que esta variable contendrá el color seleccionado por el usuario en todo momento. El valor de esta variable es un tipo de dato *String*, que tiene la forma hexadecimal en la que se representa un color en una página web.
toolSelected	Esta variable de tipo *String* indica en todo momento el nombre del pincel que selecciona el usuario para interactuar con el lienzo.

Funcionalidades	
Nombre:	leerPosicion()
Descripción:	Esta función permite conocer en todo momento las coordenadas [x,y] del cursor en la pantalla gracias a las variables *posx* y *posy*.
Nombre:	drawCanvas()
Descripción:	Permite seleccionar un pincel para trabajar en el lienzo. Cada pincel realiza diferentes operaciones sobre el lienzo, por lo que para conocer el pincel seleccionado solo es necesario conocer el valor de la variable *toolSelected*.
Nombre:	drawCanvas()
Descripción:	Este método se encarga de realizar operaciones en el canvas teniendo en cuenta el pincel que ha sido seleccionado. Para cada pincel seleccionado este método desencadena una serie de operaciones correspondientes a cada pincel.
Nombre:	panelColores()
Descripción:	Este método se ejecuta automáticamente una vez que la página comienza a construirse. Es el encargado crear paletas de colores que serán necesarias para seleccionar el color a usar en cada operación. Gracias a la notación RGB una página puede mostrar alrededor de 16.7 millones de colores en varias tonalidades y gamas, pero en este caso solo van a ser construidos los 216 principales.
Nombre:	drawLine(x1, y1, x2, y2)
Descripción:	Dibuja una línea en el lienzo teniendo en cuenta las coordenadas [x,y], el largo y el grosor introducidos por parámetro.
Nombre:	drawRectangle(x1, y1, x2, y2)

Descripción:	Dibuja un rectángulo en dependencia de los parámetros introducidos, que son coordenadas [x,y], largo y ancho.
Nombre:	arc(x,y,radio,anguloDeInicio,3.14*anguloFinal,sentidoHorario)
Descripción:	Dibuja un círculo en dependencia de los parámetros introducidos, que son coordenadas, radio del círculo, ángulo de inicio, ángulo final en radianes y el sentido en que va a comenzar a ser construido. El sentido horario es una variable booleana que indica hacia la derecha o hacia la izquierda. El valor *true* para la derecha y el valor *false* para la izquierda.

Tabla 16: Descripción del fichero componente.js

Conclusiones del Capítulo

Como resultado del estudio realizado en este capítulo, correspondiente al flujo de diseño, se definió la estrategia de integración del componente con el sistema HIS del CESIM, se identificaron las clases fundamentales que deben ser definidas para que el componente funcione satisfactoriamente. Asimismo fueron especificados los atributos y métodos que deben tener las clases para brindarle al desarrollador una idea clara de lo que se debe implementar. Se elaboró el Diagrama de Paquetes para dividir el componente en fragmentos manejables y los Diagramas de Clases de Diseño para cada caso de uso, obteniéndose de esta manera la estructura interna del funcionamiento de la solución propuesta.

Capítulo 4 Implementación

Luego de realizado el flujo de trabajo diseño, en este capítulo se procede a la realización del flujo de implementación, con el objetivo de especificar las principales características que contendrá el componente de propuesta de tratamiento en dicho aspecto. En este capítulo se realiza la elaboración del diagrama de componentes, así como el diagrama de despliegue con el objetivo de conocer la distribución física de la solución implementada, todo ello como parte del modelo de implementación.

4.1 Modelo de datos

Un modelo de datos es utilizado para la descripción de una base de datos. Por lo general, permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base) (33). Está compuesto por entidades, atributos y sus relaciones.

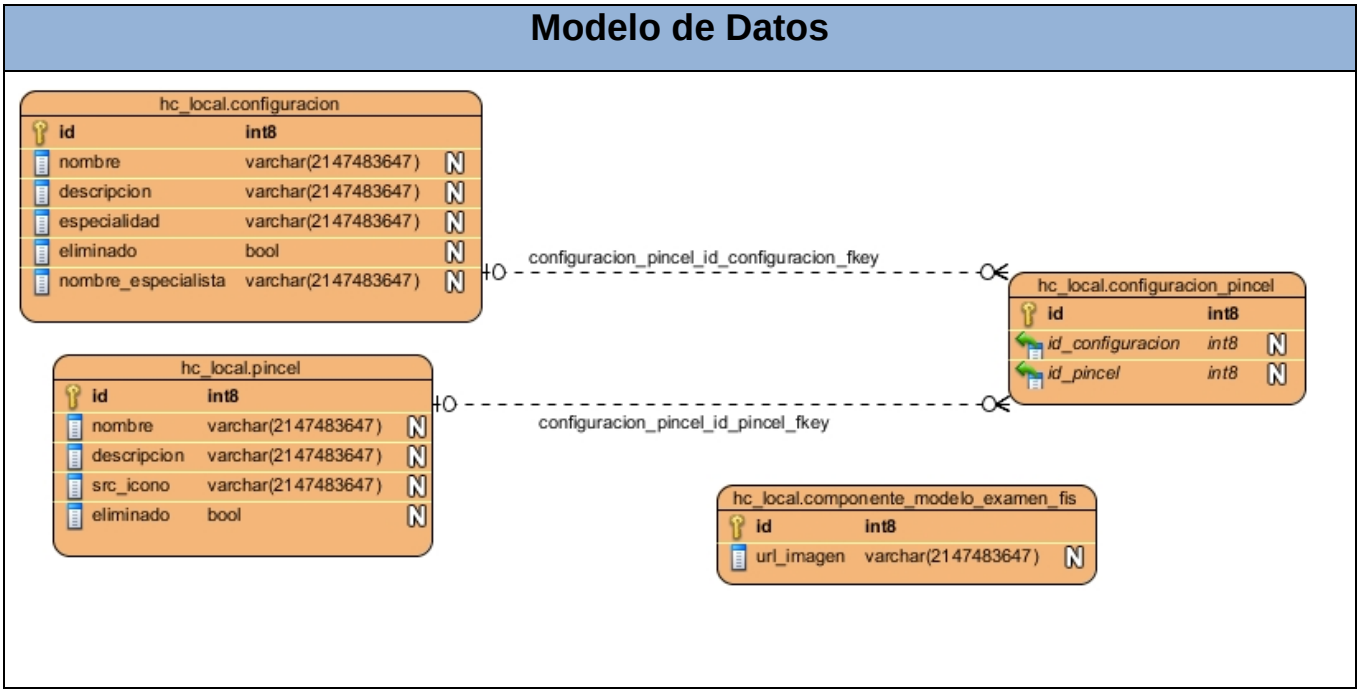

Figura 8: Modelo de datos

Descripción de las tablas de la base de datos

configuracion		
Descripción: almacena la información de todas las configuraciones existentes en el sistema.		
Atributo	**Tipo**	**Descripción**
id	bigint	Identificador (llave primaria) de esta tabla.
nombre	varchar	Nombre de la configuración.
descripcion	varchar	Una breve descripción de la configuración.
especialidad	varchar	Nombre de la especialidad de esta configuración.
eliminado	boolean	Permite la eliminación lógica con que cuenta el sistema, cuando está en verdadero indica que la entidad está eliminada.
nombre_especialista	varchar	Nombre del especialista que crea la configuración.

Tabla 17: Descripción de la tabla configuración

pincel		
Descripción: almacena la información de todos los pinceles existentes en el sistema.		
Atributo	**Tipo**	**Descripción**
id	bigint	Identificador (llave primaria) de esta tabla.
nombre	varchar	Nombre del pincel.

descripción	varchar	Una breve descripción de para qué es el pincel.
src_icono	varchar	Url de la imagen que se mostrara al dibujar con este pincel.
eliminado	varchar	Permite la eliminación lógica con que cuenta el sistema, cuando está en verdadero indica que la entidad está eliminada.

Tabla 18: Descripción de la tabla pincel

configuracion_pincel		
Descripción: almacena los identificadores de la relación de las tablas configuración y pincel.		
Atributo	**Tipo**	**Descripción**
id	bigint	Identificador (llave primaria) de esta tabla.
id_configuracion	bigint	Llave foránea, identificador de la tabla configuración.
id_pincel	bigint	Llave foránea, identificador de la tabla pincel.

Tabla19: Descripción de la tabla configuracion_pincel

componente_modelo_examen_fis		
Descripción: Almacena la url de la imagen resultante.		
Atributo	**Tipo**	**Descripción**

id	bigint	Identificador (llave primaria) de esta tabla.
url_imagen	varchar	Url de la imagen resultante después de realizar el dibujo de la lesión o el tratamiento.

Tabla 20: componente_modelo_examen_fis

4.2 Modelo de implementación

El modelo de implementación está formado por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes pueden ser encontrados datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (34).

Su propósito es definir la organización del código, planificar las integraciones del sistema e implementar las clases y subsistemas encontrados durante el diseño (34).

4.2.1 Diagrama de despliegue

El diagrama de despliegue muestra el entorno computacional donde se encuentra instalado el sistema, visualiza la configuración de los elementos de procesamiento en tiempo de ejecución con sus respectivos procesos de software (34).

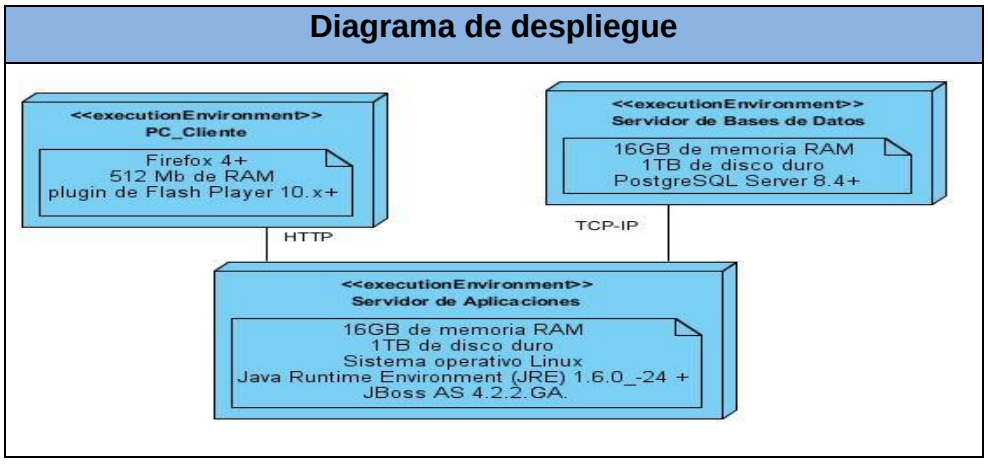

Figura 8: Diagrama de despliegue

4.2.2 Diagrama de componentes

Los diagramas de componentes describen los elementos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, entre otros. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente (34).

A continuación se muestra la distribución de componentes para el sistema propuesto:

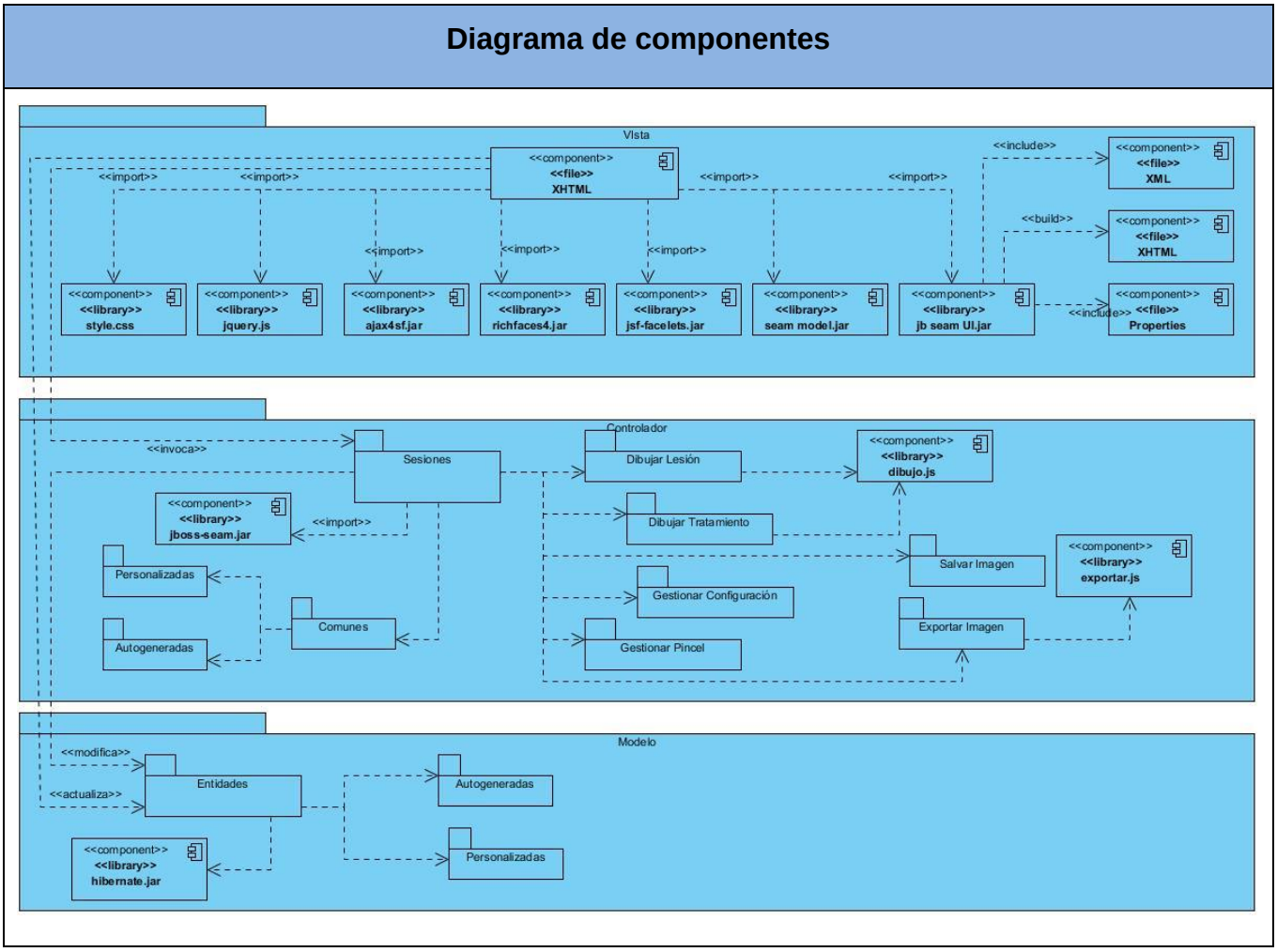

Figura 9: Diagrama de componentes

4.3 Tratamiento de errores

Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias. Las excepciones son el mecanismo recomendado para la propagación de errores que se produzcan durante la ejecución de las aplicaciones. Cuando dicho error ocurre dentro de un método Java, automáticamente se crea un objeto “Excepcion” el cual es tratado en el sistema de ejecución. Este objeto contiene información sobre la excepción, incluyendo su tipo y el estado del programa cuando ocurrió el error.

En el componente, el control de las excepciones se lleva a cabo a toda porción de código, donde pueda surgir alguna situación inesperada. También se controlan los errores que pueden surgir en la validación de datos provenientes de la interfaz de usuario, puesto que encierran una lógica compleja en cierta medida.

Para el manejo de las excepciones o errores, en las clases controladoras de procesos, se utilizará el bloque try para detectar cuando ocurra algún fallo y un bloque catch donde se manejarán dichas excepciones, mediante mensajes que se muestran en la interfaz de usuario, por las facilidades que brinda el FacesMessages, componente del framework Seam.

4.4 Seguridad

La seguridad es un tema de gran importancia para cualquier sistema de información y toma mayor relevancia cuando se gestiona información médica. La responsabilidad de mantener la seguridad cae en quien use el componente, este debe controlar quien puede acceder o no a las funcionalidades que este brinda. También debe comprobar quién puede consultar las imágenes resultantes después de la realización de la representación lesiones o un posible tratamiento.

4.5 Estrategias de Codificación. Estándares y estilos a utilizar

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación, es de gran importancia para la calidad del software. La aplicación de estándares de codificación además posibilita que el software que se obtiene sea fácil de comprender y de mantener en el tiempo (35).

4.5.1 Variables y constantes

- **Apariencia de constantes**

Todas sus letras en mayúscula: se deben declarar las constantes con todas sus letras en mayúscula.

- **Aspectos generales**

Nombres de las variables y constantes: el nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Indentación

- **Inicio y fin de bloque**

  Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.

- **Aspectos generales**

  El indentado 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 debe estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción.

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

4.5.2 Comentarios, separadores, líneas, espacios en blanco y márgenes

- **Ubicación de comentarios**

  Al inicio de cada clase o función y al final de cada bloque de código: se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.

- **Líneas en blanco**

  Se emplean antes y después de métodos, clases y estructuras: se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.

- **Espacios en blanco**

  Entre operadores lógicos y aritméticos: se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nomproducto

- **Aspectos generales**

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

4.5.3 Clases y Objetos

Apariencia de clases y objetos

Primera letra en mayúscula: los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Ejemplo: MiClase (). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.

Apariencia de atributos

Primera letra en minúscula: el nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing[1].

Declaración de parámetro 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 recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos.

Aspectos generales

Sobre las clases, los objetos, los atributos y las funciones: el nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

4.5.4 Tablas, esquemas y campos de la base de datos

- **Apariencia de las tablas**

  Todas las letras en minúscula: el nombre a emplear para las tablas debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizará guión bajo para separarlo.

- **Tablas que representen relaciones**

  Todas las letras en minúscula: el nombre a emplear para estas tablas de relación, debe tener el nombre las tablas presentes en la relación separada de guión bajo, todo en minúscula.

- **Apariencia de los campos**

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

[1] Notación CamellCasing: Especifica que la palabra de inicio del identificador comienza con minúscula. Si el identificador está compuesto por más de una palabra entonces estás deben comenzar con mayúsculas.

- **Nombre de los campos**

  En caso de identificadores: todos los campos identificadores se llamaran id.

- **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.

Conclusiones del capítulo

Con el desarrollo de las tareas correspondientes a este capítulo se logró elaborar el modelo de datos, que muestra las tablas de la base de datos que se utilizan, además se realizó el Diagrama de Despliegue y de Componentes, con la concepción de ambos quedó conformado el modelo de implementación de las funcionalidades. Durante el proceso de codificación se cumplió con los estándares y estilos definidos, lo que permitió obtener un grupo de funcionalidades entendibles para todos los programadores.

Conclusiones generales

A partir del estudio realizado y los resultados obtenidos, puede afirmarse el cumplimiento del objetivo general y por ende de las distintas tareas de la investigación.

Lo anterior se ve demostrado a través de lo siguiente:

- El análisis de las tendencias actuales de los sistemas de información hospitalaria que cuentan con visualizaciones de imágenes de modelos de examen físico permitió identificar y garantizar que se contaran con las mejores alternativas para el desarrollo del componente.
- Se estableció el uso de la etiqueta Canvas para el trabajo con imágenes en la web, como procedimiento idóneo para la implementación del componente de creación, visualización y manipulación de imágenes de modelos de examen físico.
- El retorno de la imagen resultante en un string base64 garantiza la interpretación de este resultado por cualquier sistema que decida usar el componente.
- Se desarrolló un componente capaz de incrementar la exactitud de la representación de lesiones en imágenes de modelos de examen físico para cualquier especialidad de la medicina.

Recomendaciones

Una vez concluida la investigación se recomienda lo siguiente:

- Continuar con la investigación e implementación de una nueva versión de este componente para sistemas hospitalarios, usando capas (*layers*) para tratar las formas (*shapes*) creadas y las imágenes como objetos separados del Canvas, garantizando un mayor control sobre la representaciones en las imágenes de modelos de examen físico influyendo directamente en los diagnósticos médicos.

Referencias bibliográficas

1. **Antonio, Dr. Cerritos**. Sistema de Información Hospitalaria. [En línea] [Citado el: 2013 de octubre de 25.] http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf.

2. **Vázquez, Prof Médico Endocrinólogo Richard López.** *Examen Físico Curso de Fisiopatología.* México : s.n., 2010.

3. **Medicina, Real Academia Nacional de.** *Diccionario de términos médicos.* Madrid : Panamericana, 2012.

4. **MediaActive.** *El gran libro de Photoshop CS2.* Barcelona : Díaz Tudur, 2010.

5. **GP, DDS.** DDS GP. [En línea] [Citado el: 2013 de octubre de 25.] http://www.ddsgp.com/about-us.html.

6. **AquarSoftware.** AquarSoftware. [En línea] [Citado el: 2013 de octubre de 25.] http://www.aquarsoftware.com/.

7. **Soft, Pring.** Pring Soft. [En línea] [Citado el: 2013 de octubre de 25.] http://www.pring-soft.com/catalog7.html.

8. **PlasticSurgerySimulator.** PlasticSurgerySimulator. [En línea] [Citado el: 2013 de octubre de 25.] http://www.plastic-surgery-simulator.com/es/front .

9. **León, Jeimy.** Incytde. [En línea] [Citado el: 2014 de enero de 21.] http://incytde.org/incytde/node/86..

10. **Gómez, José Jorge Márquez.** Arquitectura MVC Visión General. [En línea] [Citado el: 2014 de enero de 21.] http://jorge.queideas.com/wp-content/uploads/2011/11/Arquitectura-MVC.pdf.

11. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Estados Unidos : Addison Wesley, 2004. 9788478290369 .

12. **García de Jalón, J., Rodriguez Iñigo, M. J., Alfonso Brazález, A. I., Larzabal, A., Calleja, J., & García,J.** *Aprenda Java como si estuviera en primero.* Universidad de Navarra, España. : Escuela Superior de Ingenieros Industriales., Enero de 2000.

13. **H, Ty.** Manuales. [En línea] [Citado el: 2014 de enero de 21.] http://max-alva.webs.com/javascript.html.

14. **Ignside.** Cascade Style Sheets. [En línea] [Citado el: 21 de enero de 2014.] http://www.ignside.net/man/css/index.php.

15. **Group, Expert.** Javaserverfaces. [En línea] [Citado el: 16 de noviembre de 2013.] http://www.javaserverfaces.org/specification/expert-group.

16. **Community, JBoss.** JBoss Community. [En línea] [Citado el: 21 de enero de 2014.] http://www.jboss.org/richfaces.

17. **Ramos, Ing Juan Alonso.** Introducción a Ajax4jsf. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.

18. **Hookom, Jacob.** Facelets - JavaServer Faces View Definition Framework. [En línea] [Citado el: 2014 de enero de 21.] http://facelets.java.net/nonav/docs/dev/docbook.html#intro.

19. **Application, Web.** Plataforma J2EE. JBoss Seam Framework. [En línea] [Citado el: 21 de noviembre de 2013.]

20. **Corbillón, Javier Oliver.** *Desarrollo de funcionalidades del módulo Facturación del sistema alas HIS orientado al consumo de productos farmacéuticos.* Habana.Cuba : s.n., 2012.

21. **Lucifer, Prometeo.** Java Runtime Environment - JRE. [En línea] [Citado el: 28 de enero de 2014.] http://www.elleonplateadodeojosrojos.es/blog/java-runtime-environment-jre/ .

22. **Hernandis, John. A.** Versioncero. [En línea] [Citado el: 21 de enero de 2014.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.

23. **Release, Ganymede.** Ganymede Release. [En línea] [Citado el: 21 de enero de 2014.] https://www.eclipse.org/ganymede/.

24. **Scribd.** pstrgreSQL-Investigación. [En línea] [Citado el: 21 de enero de 2014.] http://es.scribd.com/doc/36570462/postgreSQL-investigacion.

25. Guía de Ubuntu. [En línea] [Citado el: 21 de enero de 2014.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III&printable=yes.

26. **Alférez Sánchez, José A.** *Instalación, configuración y administración del Servidor de aplicaciones JBoss.*

27. **Garcerant, Iván. Tecnología y Synergix.** Modelo de Dominio. [En línea] [Citado el: 15 de enero de 2014.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio.

28. **Pressman, Roger S.** *Ingeniería del software: un enfoque práctico.* Madrid : s.n., 1997.

29. **Maure, Amaya Álvarez Lorenzo y Mirelio Mora.** *Desarrollo de funcionalidades para la especialidad de Psicología del módulo Consulta Externa del sistema alas HIS.* La Habana : s.n., 2012.

30. **E, Flores.** Seminario Reutilización de Software. [En línea] [Citado el: 13 de marzo de 2014.] http://users.dsic.upv.es/~prosso/resources/FloresEtAl_TIMM11.pdf.

31. **Ruiz, Francisco.** Diseño de Software. [En línea] [Citado el: 15 de marzo de 2014.] http://alarcos.esi.uclm.es/per/fruiz/cur/santander/fruiz-pn.pdf.

32. **Tedeschi, Nicolás.** Developer Network. [En línea] [Citado el: 15 de marzo de 2014.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

33. Modelo de Datos. [En línea] [Citado el: 16 de marzo de 2014.] http://definicion.de/modelo-de-datos.

34. **Tedeschi, Nicolás.** Modelo de Implementación: Diagramas de Componentes y Despliegue. [En línea] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

Bibliografías

1. **Antonio, Dr. Cerritos**. Sistema de Información Hospitalaria. [En línea] [Citado el: 2013 de octubre de 25.] http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf.

2. **Vázquez, Prof Médico Endocrinólogo Richard López.** *Examen Físico Curso de Fisiopatología.* México : s.n., 2010.

3. **Medicina, Real Academia Nacional de.** *Diccionario de términos médicos.* Madrid : Panamericana, 2012.

4. **MediaActive.** *El gran libro de Photoshop CS2.* Barcelona : Díaz Tudur, 2010.

5. **GP, DDS.** DDS GP. [En línea] [Citado el: 2013 de octubre de 25.] http://www.ddsgp.com/about-us.html.

6. **AquarSoftware.** AquarSoftware. [En línea] [Citado el: 2013 de octubre de 25.] http://www.aquarsoftware.com/.

7. **Soft, Pring.** Pring Soft. [En línea] [Citado el: 2013 de octubre de 25.] http://www.pring-soft.com/catalog7.html.

8. **PlasticSurgerySimulator.** PlasticSurgerySimulator. [En línea] [Citado el: 2013 de octubre de 25.] http://www.plastic-surgery-simulator.com/es/front .

9. **León, Jeimy.** Incytde. [En línea] [Citado el: 2014 de enero de 21.] http://incytde.org/incytde/node/86..

10. **Gómez, José Jorge Márquez.** Arquitectura MVC Visión General. [En línea] [Citado el: 2014 de enero de 21.] http://jorge.queideas.com/wp-content/uploads/2011/11/Arquitectura-MVC.pdf.

11. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Estados Unidos : Addison Wesley, 2004. 9788478290369 .

12. **García de Jalón, J., Rodriguez Iñigo, M. J., Alfonso Brazález, A. I., Larzabal, A., Calleja, J., & García,J.** *Aprenda Java como si estuviera en primero.* Universidad de Navarra, España. : Escuela Superior de Ingenieros Industriales., Enero de 2000.

13. **H, Ty.** Manuales. [En línea] [Citado el: 2014 de enero de 21.] http://max-alva.webs.com/javascript.html.

14. **Ignside.** Cascade Style Sheets. [En línea] [Citado el: 21 de enero de 2014.] http://www.ignside.net/man/css/index.php.

15. **Group, Expert.** Javaserverfaces. [En línea] [Citado el: 16 de noviembre de 2013.] http://www.javaserverfaces.org/specification/expert-group.

16. **Community, JBoss.** JBoss Community. [En línea] [Citado el: 21 de enero de 2014.] http://www.jboss.org/richfaces.

17. **Ramos, Ing Juan Alonso.** Introducción a Ajax4jsf. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.

18. **Hookom, Jacob.** Facelets - JavaServer Faces View Definition Framework. [En línea] [Citado el: 2014 de enero de 21.] http://facelets.java.net/nonav/docs/dev/docbook.html#intro.

19. **Application, Web.** Plataforma J2EE. JBoss Seam Framework. [En línea] [Citado el: 21 de noviembre de 2013.]

20. **Corbillón, Javier Oliver.** *Desarrollo de funcionalidades del módulo Facturación del sistema alas HIS orientado al consumo de productos farmacéuticos.* Habana.Cuba : s.n., 2012.

21. **Lucifer, Prometeo.** Java Runtime Environment - JRE. [En línea] [Citado el: 28 de enero de 2014.] http://www.elleonplateadodeojosrojos.es/blog/java-runtime-environment-jre/ .

22. **Hernandis, John. A.** Versioncero. [En línea] [Citado el: 21 de enero de 2014.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.

23. **Release, Ganymede.** Ganymede Release. [En línea] [Citado el: 21 de enero de 2014.] https://www.eclipse.org/ganymede/.

24. **Scribd.** pstrgreSQL-Investigación. [En línea] [Citado el: 21 de enero de 2014.] http://es.scribd.com/doc/36570462/postgreSQL-investigacion.

25. Guía de Ubuntu. [En línea] [Citado el: 21 de enero de 2014.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III&printable=yes.

26. **Alférez Sánchez, José A.** *Instalación, configuración y administración del Servidor de aplicaciones JBoss.*

27. **Garcerant, Iván. Tecnología y Synergix.** Modelo de Dominio. [En línea] [Citado el: 15 de enero de 2014.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio.

28. **Pressman, Roger S.** *Ingeniería del software: un enfoque práctico.* Madrid : s.n., 1997.

29. **Maure, Amaya Álvarez Lorenzo y Mirelio Mora.** *Desarrollo de funcionalidades para la especialidad de Psicología del módulo Consulta Externa del sistema alas HIS.* La Habana : s.n., 2012.

30. **E, Flores.** Seminario Reutilización de Software. [En línea] [Citado el: 13 de marzo de 2014.] http://users.dsic.upv.es/~prosso/resources/FloresEtAl_TIMM11.pdf.

31. **Ruiz, Francisco.** Diseño de Software. [En línea] [Citado el: 15 de marzo de 2014.] http://alarcos.esi.uclm.es/per/fruiz/cur/santander/fruiz-pn.pdf.

32. **Tedeschi, Nicolás.** Developer Network. [En línea] [Citado el: 15 de marzo de 2014.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

33. Modelo de Datos. [En línea] [Citado el: 16 de marzo de 2014.] http://definicion.de/modelo-de-datos.

34. **Tedeschi, Nicolás.** Modelo de Implementación: Diagramas de Componentes y Despliegue. [En línea] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

35. **Rodríguez, Bertha de los Ángeles Vaillant Preval y Dáyron Aguila.** *Desarrollo del módulo Banco de Ojos del Sistema Información Hospitalaria alas HIS.* La Habana : s.n., 2012.

36. Capítulo 2 El lenguaje Java. [En línea] [Citado el: 2013 de diciembre de 6.] http://www.dma.fi.upm.es/mabellanas/voronoi/java/java.htm#2.1.1.

37. **Cifuentes Briceño, Wilmer de Jesús.** Modelo de datos para un sistema automatizado de control de costos de proyecto. [En línea] [Citado el: 2014 de marzo de 20.] http://tesis.luz.edu.ve/tde_busca/arquivo.php?codArquivo=2235.

38. **García, Alejandro Pérez.** Desarrolloweb.com. [En línea] [Citado el: 2014 de enero de 21.] http://www.desarrolloweb.com/articulos/2380.php.

39. **Community, JBoss.** JBoss Community. [En línea] [Citado el: 2014 de enero de 21.] http://www.jboss.org/richfaces.

40. —. JBoss Community. [En línea] [Citado el: 2014 de enero de 21.] http://www.jboss.org/jbossajax4jsf/docs/devguide/en/html/Introduction.html.

41. **Wikimedia Foundation.** Canvas del HTML 5 . [En línea] [Citado el: 21 de enero de 2014.] ttp://es.wikipedia.org/wiki/Canvas_%28HTML%29.

42. **Desarrolloweb.** Introducción a Canvas del HTML 5. [En línea] [Citado el: 21 de enero de 2014.] http://www.desarrolloweb.com/articulos/introduccion-canvas-html5.html.

43. **Campos, Beatris Prieto.** Diseño y programación de páginas web . [En línea] [Citado el: 21 de enero de 2014.] http://atc.ugr.es/~bprieto/paginas-web8/introduccionx.html.

44. **Jaramillo, Wilmer.** Software Libre de Venezuela 777, C.A. [En línea] [Citado el: 21 de enero de 2014.] http://wilmer.fedorapeople.org/files/presentations/JBoss.pdf..

45. **González Cornejo, José Enrique.** ¿Qué es UML? [En línea] [Citado el: 21 de noviembre de 2013.] http://www.docirs.cl/uml.htm..

46. Descripción de Eclipse SDK 3.3.2. [En línea] [Citado el: 21 de noviembre de 2013.] https://www.ohloh.net/p/pgadmin.

47. **Pérez, Javier Egíluz.** Librosweb.es. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.librosweb.es/xhtml/capitulo1/html_y_xhtml.html.

48. **Franky, María Consuelo.** Java EE 5. [En línea] [Citado el: 12 de noviembre de 2013.] http://www.acis.org.co/fileadmin/Conferencias/ConfConsueloFranky_Abr19.pdf.

49. **Ottinger, Joseph.** TheServerSide.com. JBoss releases JBoss Tools, Eclipse Plugins including Exadel. [En línea] [Citado el: 12 de noviembre de 2013.] http://www.theserverside.com/news/thread.tss?thread_id=45933.

50. **MeRinde.** MeRinde. [En línea] [Citado el: 16 de noviembre de 2013.] http://merinde.net/index.php?option=com_content&task=view&id=495&Itemid=291.

51. **Mora, Francisco.** UML: Lenguaje Unificado de Modelado. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.dccia.ua.es/dccia/inf/asignaturas/GPS/archivos/Uml.PDF.

52. **VP UML**. Sitio oficial visual paradigm. [En línea] [Citado el: 12 de diciembre de 2013.] http://www.visual-paradigm.com/product/vpuml/.

53. **Ochoa Reyes, Alexeis Joel y Orellana García, Arturo.** *Vista de análisis usando técnicas de agrupamiento para el sistema.* La Habana, Cuba : Universidad de las Ciencias Informática, 2012.

55. **Application, Web.** Plataforma J2EE. JBoss Seam Framework. [En línea] [Citado el: 21 de noviembre de 2013.] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework.

Glosario de términos

App: programa informático creado para llevar a cabo o facilitar una tarea en un dispositivo informático**.**

CEO: Director ejecutivo.

iOS: sistema operativo móvil de la empresa Apple .

Android: sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil .

Plataforma: es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.

Multiplataforma: término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

Framework: es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos.

XML: siglas de eXtensible Markup Language. Lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro de un documento, así como la reutilización de partes del mismo.

BSD (Berkeley Software Distribution): licencia de software otorgada principalmente para los sistemas Berkeley Software Distribution (BSD).

Java Management Extensions (JMX): tecnología que define una arquitectura de gestión, la API (Application Programming Interface), los patrones de diseño, y los servicios para la monitorización/administración de aplicaciones basadas en Java.

UI: interfaz de usuario.

API: Interfaz de Programación de Aplicaciones, cuyo acrónimo en inglés es API (Application Programming Interface), es un conjunto de funciones residentes en bibliotecas generalmente dinámicas. Permiten que una aplicación corra bajo un determinado sistema operativo.

JPA: Interfaz de Persistencia Java es el estándar de Java encargado de automatizar dentro de lo posible la persistencia de nuestros objetos en base de datos.

GRASP: Patrones de Software Asignación de Responsabilidades Generales, son patrones generales de software para asignación de responsabilidades, es el acrónimo de "GRASP (object-oriented design General Responsibility Assignment Software Patterns)". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

Anexos

Figura A1: Buscar configuración

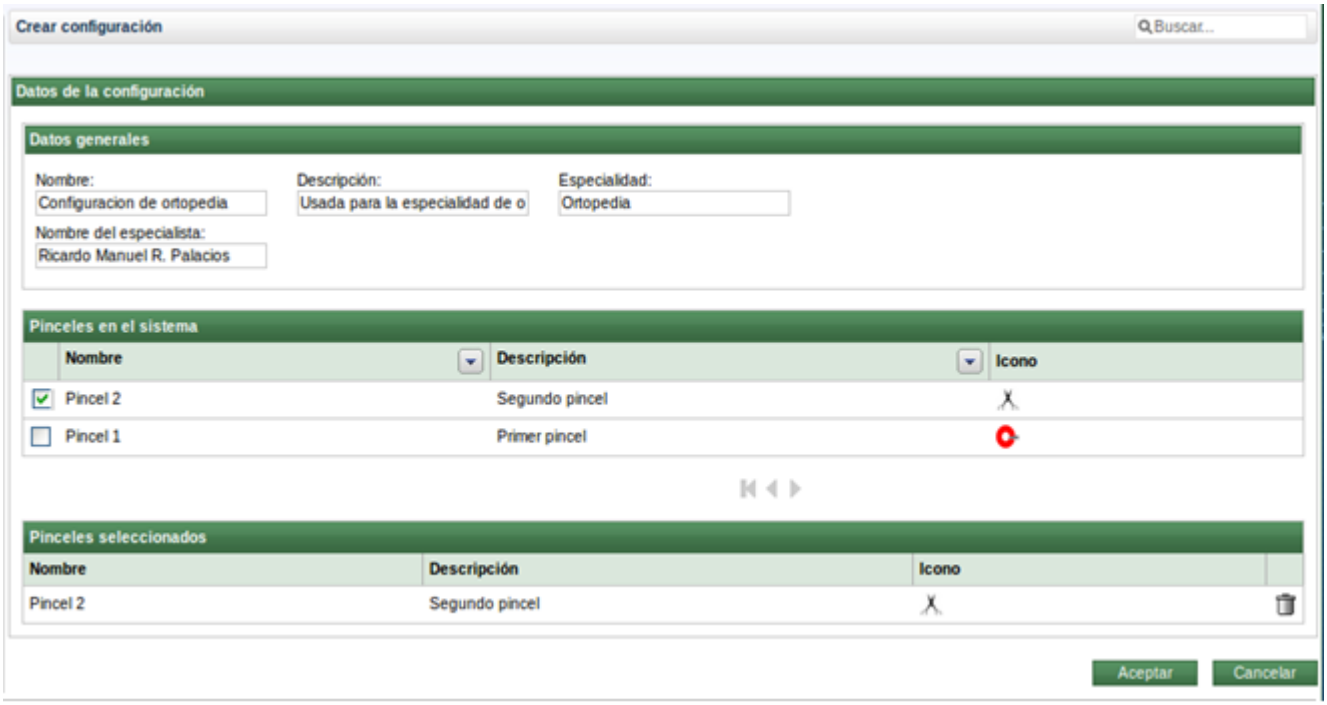

Figura A2: Crear configuración

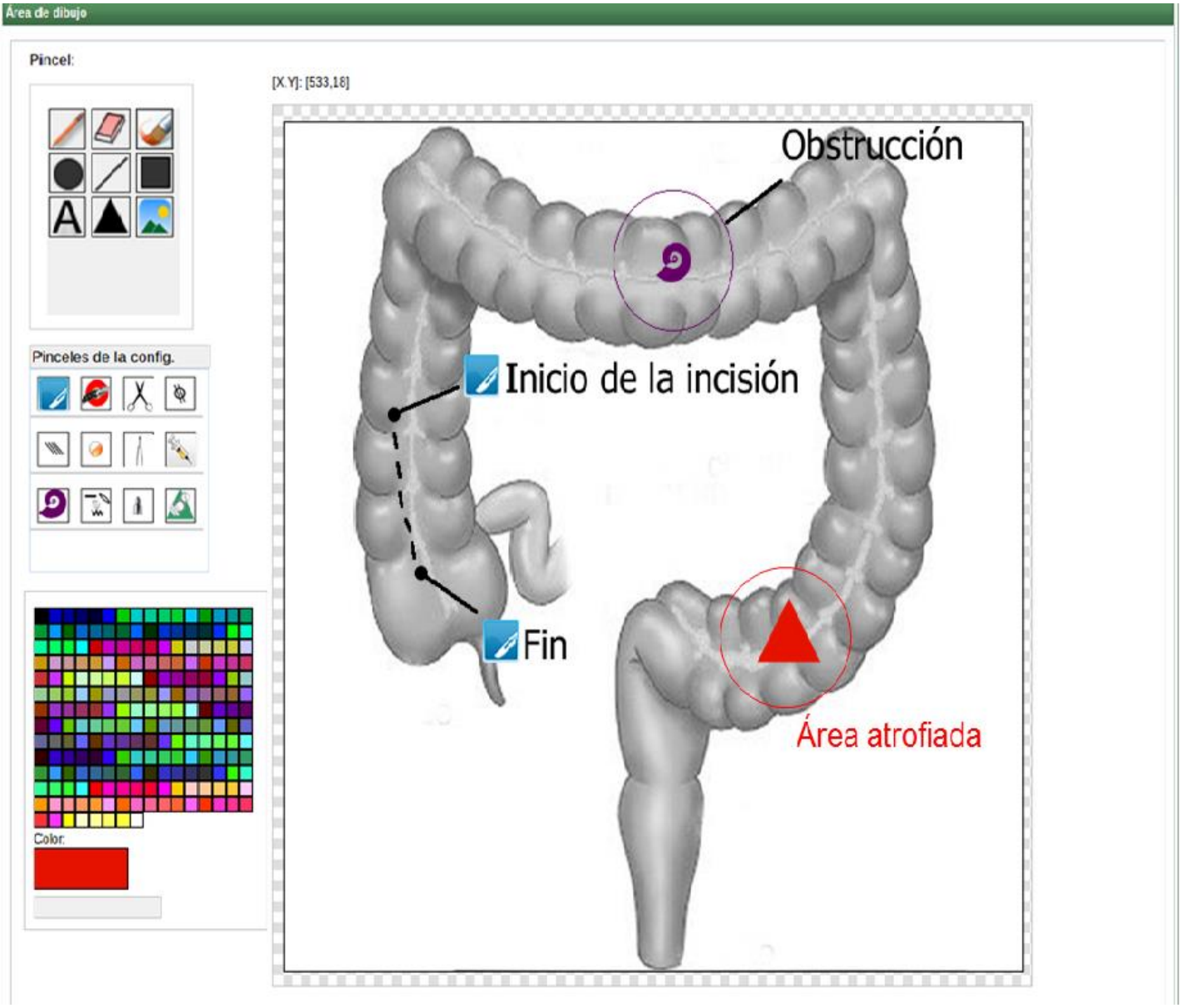

Figura A3: Dibujar lesión

Figura A4: Introducir texto