Universidad de las Ciencias Informáticas

Facultad 6 *Bioinformática*

Título: “Sistema Informático para la gestión de la información resultante del trabajo con las Imágenes de Resonancia Magnética en el proyecto Mapeo Cerebral Humano Cubano”

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

Autores: Linet Aquino Martínez
Yenisel Ávila Vázquez

Tutores: Ing. Maikel Laurencio Giralt
Ing. Yoamel Acosta Morales
Lic. Félix Argelio Martínez Nariño

Junio 2009

PENSAMIENTO

"La pasión de saber hace que el hombre aprenda más rápidamente y aprenda en menos tiempo; la pasión de saber, la conciencia de la necesidad de saber, hace que los conocimientos se adquieran más rápidamente y, sobre todo, la vida, el trabajo práctico, los problemas diarios, constantemente nos estarán enseñando la necesidad de cada conocimiento."

Fidel Castro Ruz

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 ____ días del mes de _____ del año _____.

_____ _____

Autor: Linet Aquino Martínez ***Autor:*** Yenisel Ávila Vázquez

_____ _____

Tutor: Ing. Yoamel Acosta Morales ***Tutor:*** Ing. Maikel Laurencio Giralt

DATOS DE CONTACTO

Ing. Maikel Laurencio Giralt
Universidad de las Ciencias Informáticas
Email: mlaurencio@uci.cu

Ing. Yoamel Acosta Morales
Universidad de las Ciencias Informáticas
Email: yamorales@uci.cu

Lic. Félix Argelio Martínez Nariño
Universidad de las Ciencias Informáticas
Email: famartinez@uci.cu

AGRADECIMIENTOS

El presente trabajo no hubiese sido posible sin el apoyo de muchas personas durante estos 5 años. Que nos han brindado su apoyo y sus conocimientos en cada una de las dudas aparecidas en algunos de los temas estudiados, a todas ellas muchas gracias. En especial agradecerles:

A nuestras familias *por el cariño y la confianza que nos han brindado en todo momento, pero sobre todo por estar, cada uno a su manera, brindándonos su apoyo para alcanzar nuestro objetivo.*

A nuestros tutores*, Ing. Yoamel Acosta Morales, Ing. Maikel Laurencio Giralt, Lic. Félix Argelio Martínez Nariño por cada una de las revisiones y recomendaciones realizadas al trabajo de diploma, por estar siempre presentes y disponibles.*

A todos los profesores *que han contribuido con nuestra educación y formación profesional a lo largo de nuestra vida estudiantil.*

A los profesores y estudiantes del proyecto Mapeo Cerebral Humano Cubano*, Alex Tablada Álvarez, Dayana De La Mella Reus, Félix Roberto Aballi Morell, Francisco Muñoz Tamayo, Maikel Laurencio Giralt, Marleysi Lopez Duque, Raidel Oceguera Ravelo, Rodolfo Ruiz Padilla; Yixander Yero Tarancon, Yoamel Acosta Morales, Raudelis Figueira Jimenez y Mirielys Avila Leyva y en especial a Alejandro Claudio Perez Romeu y Yosbel Rodriguez Rodríguez.*

A: *Geniver y Carlos Manuel Guilarte Nápolez, Daisy Nápolez, Bernardo Guilarte, Felipe Áreas Rodríguez, Rachid Alí, Yumis Figueroa, Mercedes Castañet Izquierdo, Geordanis Carrillo Ávila, Berta Iris, Niurka Martínez Durán (oponente), Arsenio Llamo, Ismel Sarduy Sánchez, Reinier Tan Matamoros.*

```
for ( int i = 0; i< ∞; i++){

System.out.println(" Muchas Gracias a Todos");

}
```

DEDICATORIA

A:

La Revolución…por Todo.

Mis Padres *…por estar a mi lado en el momento justo… por su Amor, por enseñarme a crecer, por su ejemplo de la honradez, sacrificio y sencillez, por sus castigos formadores, por enseñarme a dar sin nada esperar, por sus consejos , por enseñarme como es la vida…. por Todo.*

Mis Hermanos *…por su alegría, por estar siempre unidos, a Yosdanis por su bondad y sus locuras, a Yosbanis por seguirme los pasos al precios de los mismos sacrificios, por su Título de Oro en el IPI de Informática, por estar orgulloso de ti, a mi hermana Yaimara, por su Amor para todos, por ser única entre nosotros tres.*

Zoila Bárbara Guerra Maceo (tía) *… por su ejemplo de dedicación, de sacrificio constante, de voluntad, por su espíritus progresista, por confiar en mí, por su sensibilidad humana, por ser el espejo que siempre miré para llegar a ser otro Ingeniero en la familia.*

Ania Guilarte Nápolez (novia) *…. por su Amor, por su Apoyo, por ser mi Alegría, mi Amiga, mi Fantasía y Verdad, por ser mi Camino y mi Felicidad.*

Yenisel Avila Vázquez.

DEDICATORIA

A:

Mis Padres por estar siempre a mi lado, por sus consejos, por enseñarme a ser cada día mejor, por exigirme siempre que diera lo mejor de mí, por su dedicación y sobre todas las cosas por su amor.

Mi hermana por quererme tanto, por sus locuras y por su alegría.

Mi Abuelo Guzmán que aunque no está hoy para verme graduada con su ejemplo siempre me exhortó a ser una buena profesional.

Mi Abuela Margot por su bondad, su paciencia y su dulzura.

Mis Abuelos Emelina y Berto por el cariño que me tienen y la sencillez que los caracteriza.

Mis Amigas Liusmila, Yoslainy y Dayana por haberme acompañado incondicionalmente en estos años de universidad y hacerlos inolvidables.

Especialmente a Fidel y a la Revolución que me dieron la oportunidad de graduarme en esta Universidad.

Linet Aquino Martínez

RESUMEN

Actualmente la Informática constituye una ciencia muy importante para el desarrollo económico, social y científico de todos los países. Como respuesta a los avances de las ciencias computacionales se ha notado una creciente complejidad de las aplicaciones informáticas, dichas aplicaciones están destinadas a todas las ramas científicas. En la actualidad la informática está iniciándose en el campo neurocientífico, a pesar de existir diferentes herramientas para el trabajo con imágenes cerebrales, son pocas para el estudio de lo que se conoce como la parte más compleja del cuerpo humano, el Cerebro.

El Centro de Neurociencia de Cuba lleva a cabo el desarrollo del proyecto Mapeo Cerebral Humano Cubano (MCHC) en conjunto con la Universidad de las Ciencias Informáticas (UCI) con el objetivo de crear herramientas informáticas que sean capaces de facilitarle la investigación a los neurocientíficos.

La herramienta informática propuesta consiste en una Aplicación Web desarrollada con tecnologías libres, entre las más importantes podemos citar: Java como lenguaje de programación, Java Server Fase (JSP), los framework de desarrollo Spring 2.1 e Hibernate 3.0 y PostgresSQL 8.2 como gestor de base datos. La aplicación unifica las herramientas utilizadas en la investigación para que los especialistas puedan gestionar toda la información generada del procesamiento de las Imágenes de Resonancia Magnética de un modo más fácil, hasta el momento con los estudios iniciales dicha información sobrepasa los 2 terabytes. Permite que los investigadores puedan realizar búsquedas avanzadas de un tipo de imagen cerebral (*Fase, Magnitud, Anatómicas, Difusión, Atlas, Segmentación*, entre otras) de acuerdo a los parámetros de coincidencia que decidan en los sujetos (*Rasa, Manualidad, Sexo, Rango de edad,* por citar algunos), visualizar además las imágenes cerebrales que deseen , permite conocer cuántos estudios se han realizado hasta el momento y en qué fecha, y además gestiona la cantidad de sujetos a los cuales se les han realizado los estudios (MRI) permitiendo conocer de los mismo, todos los datos relacionados a un paciente.

ÍNDICE

Índice

Índice

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En nuestros días, estamos inmersos en la era de las neuroimágenes y resulta sorprendente cómo diversos equipos son capaces de mostrarle al ojo humano el interior de nuestros incluyendo el más complejos de todos, el cerebro. Una opción para estudiar con profundidad el cerebro humano propone el mapeo cerebral, herramienta que estudia la neuroanatomía funcional del órgano a través de técnicas cuantitativas de neuroimágenes y de un análisis computacional sofisticado.

El Centro de Neurociencias de Cuba (CNEURO) está coordinando el proyecto Mapeo Cerebral Humano Cubano (MCHC). El mismo tiene el objetivo de crear un ATLAS de la estructura y funcionamiento cerebral aportando conocimientos sobre los sustratos neurales tanto de la función normal como sus alteraciones en las enfermedades neuropsiquiátricas. Otro propósito es la creación de herramientas cuantitativas para la pesquisa activa y la evaluación de tratamientos.

Notable avance ha manifestado CNEURO en esta compleja materia. En los últimos años, creó una nueva técnica de neuroimágenes producto de la fusión de la Tomografía Eléctrica y la Resonancia Magnética. Esta variante permite recrear en tres dimensiones una distribución de las corrientes que produce el cerebro a partir de la actividad eléctrica y magnética sobre el cuero cabelludo, reproduciendo retratos de la corriente eléctrica que circula por las neuronas, lo que facilita evaluar el funcionamiento cerebral e investigar las posibles enfermedades que el cerebro puede padecer.

Actualmente las disfunciones cerebrales (enfermedades psiquiátricas, neurológicas y del desarrollo infantil) acaparan más del 34% de los años de vida útil que pierde la humanidad. Sin embargo su diagnóstico se hace difícil por la extrema complejidad y variabilidad que tienen los cerebros de distintas personas aún siendo estos normales. Por ello los especialistas se plantean auxiliarse de herramientas informáticas que permitan detectar cambios sutiles que sustenten la detección precoz de estas patologías.

A esta unión de la Informática y las Neurociencias se le conoce como “Neuroinformática”.

El proyecto de Neurociencias de más relevancia a nivel mundial es el Proyecto Internacional de Mapeo Cerebral Humano. Es un análogo en el campo de las neurociencias del Proyecto Genoma Humano y está destinado a la obtención de un Mapa o Atlas Cerebral Patrón. Éste proyecto ya cosecha resultados importantes y ha recibido un grado creciente de atención de los medios informativos internacionales y entidades de países principalmente del mundo desarrollado entre los que figuran: Japón, Australia, Estados Unidos, Canadá, Finlandia, Holanda, Inglaterra, Francia.

Introducción

Cuba se incorpora al Proyecto Internacional de Mapeo Cerebral Humano con el empeño de crear herramientas diagnósticas novedosas de las enfermedades neuropsiquiátricas. El proyecto MCHC ha sido el primero a nivel internacional en incorporar el mapeo de la actividad eléctrica cerebral, a través del Electroencefalograma (EEG). En 1993 se desarrolló la Tomografía Eléctrica Cerebral (TEC), que consiste en recrear una distribución en tres dimensiones de las corrientes que producen las neuronas a partir de la actividad eléctrica magnética sobre la superficie del cuero cabelludo por EEG. Aún no se incluyó la anatomía cerebral de cada individuo por no haber en el país en ese momento ningún equipo de Resonancia Magnética (RM) de alto campo.

Como la TEC tiene una alta resolución temporal y la RM una alta resolución espacial (esta última mide las zonas que se activan en el cerebro, pero no captan imágenes en poco tiempo), los neurocientíficos cubanos decidieron entonces aprovechar lo positivo de ambas y fusionarlas. La nueva Tomografía Eléctrica y Magnética Cerebral Cubana permite conocer con una precisión máxima qué sitios y en qué orden se activan dentro del cerebro, ante un determinado estímulo a estudiar. La aplicación de esta técnica le daría características únicas al proyecto cubano: tener un carácter anatómico-funcional y ser la primera vez que se analizan personas de América Latina.

Los estudios realizados a una muestra de sujetos (1574 sujetos) obtenida al azar y con la aprobación de los mismo para formar parte del proyecto, ha permitido obtener los resultados del EEG e Imágenes de Resonancia Magnética (MRI) de alto campo, generando información que sobrepasa los 2 terabyte de datos almacenados.

Una parte importante del proyecto MCHC busca crear un atlas cuantitativo del desarrollo cerebral normal del cubano entre otras, que le permita realizar estudios tempranos y preventivos contra posibles desórdenes cerebrales. Esto sería básicamente con todo el proceso de gestión del trabajo con las MRI que le permita realizar los análisis correspondientes.

En el proceso de trabajo con las Imágenes de Resonancia Magnética en el proyecto Mapeo Cerebral Humano Cubano se utilizan diversas herramientas informáticas, algunas desarrolladas por los propios científicos de CNEURO o por terceros. El proceso consta de varios flujos que se identifican genéricamente como “Pipeline”. Cada Pipeline pasa por diferentes estados que generan grandes volúmenes de información (2 terabytes de capacidad hasta el momento), que le dificulta la investigación a los neurocientíficos al no poder realizar búsquedas avanzadas de diferentes tipos de imágenes que corresponden a un sujeto o a varios sujetos coincidentes según los parámetros de búsquedas deseados por los investigadores, así como visualizar dichas imágenes.

Ante tales dificultades los autores se plantean resolver el siguiente **Problema Científico**:

¿Cómo gestionar la información resultante del procesamiento de Imágenes de Resonancia Magnética en el proyecto Mapeo Cerebral Humano Cubano?

El **Objeto de Estudio** de la investigación comprende:

El proceso de trabajo con Imágenes de Resonancia Magnética

Los esfuerzos están encaminados al siguiente **Campo de Acción**:

La gestión de la información resultante del procesamiento de Imágenes de Resonancia Magnética a través de Aplicaciones Web.

Se plantea el siguiente **Objetivo General**:

Desarrollar una Aplicación Web que permita la gestión de la información resultante del procesamiento de Imágenes de Resonancia Magnéticas del proyecto Mapeo Cerebral Humano Cubano.

Para lograr el Objetivo General se han trazado los siguientes **Objetivos Específicos**:

- Definir las funcionalidades que tendrá el sistema informático.
- Diseñar el sistema informático.
- Implementar el sistema informático.
- Someter la solución informática que se brinda a una serie de pruebas de caja negra que validen que la aplicación cumple con las funcionalidades especificadas.

Marco contextual

La aplicación permitirá gestionar la información resultante del procesamiento de las imágenes de cada sujeto con mayor rapidez y eficiencia, para así poder obtener los resultados de los análisis en menor tiempo y poder determinar un mejor tratamiento en caso de que el sujeto padezca alguna de las enfermedades neurológicas, además de facilitar el trabajo del personal médico en el centro de neurociencias.

Marco teórico

El módulo de Imágenes de Resonancia Magnética (MRI) como parte de dicho sistema informático debe cumplir con ciertas características establecidas para su confección para que sea posible su integración

a la red de salud cubana, para llevar a cabo este propósito se utilizará PostGreSQL como gestor de base de datos, JSP y Java como lenguajes de programación, Apache Tomcat como servidor de aplicaciones, se utilizarán los frameworks Spring en el modelado del negocio e Hibernate para generar la capa de acceso a datos, utilizando además el DBDesigner Fork para el diseño de la base de datos . Se eligió la metodología: OpenUP, debido a que reduce la documentación y los roles existentes, además está preparada para desarrollar grandes y complejos proyectos y utiliza el lenguaje unificado de modelado (UML) para representar todos los esquemas de un sistema de software.

Variables involucradas en la investigación

Variable independiente: El Sistema Informático desde una aplicación Web

Variable dependiente: La gestión eficaz del proceso de trabajo con Imágenes de Resonancia Magnética del proyecto Mapeo Cerebral Humano Cubano.

Variable ajena: Soporte tecnológico

Tareas para complementar los objetivos del trabajo:

- Realización de entrevistas con el cliente.
- Investigación de las tendencias y tecnologías actuales para darle cumplimiento al problema planteado.
- Análisis y selección de las tecnologías y herramientas para desarrollar la aplicación.
- Desarrollo de las actividades del flujo de trabajo de requerimientos
- Modelación de las actividades correspondientes al análisis y diseño de la aplicación.
- Implementación de los requerimientos de la aplicación
- Realización de pruebas de caja negra para validar la solución propuesta.

Con el desarrollo de este sistema la meta principal es que todos los neurocientíficos de CNEURO tengan en sus manos una herramienta que les permita gestionar toda la información resultante del procesamiento de las Imágenes de Resonancia Magnéticas.

El contenido de este trabajo está distribuido de la siguiente manera: una introducción, cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos.

Capítulo 1: Fundamentación teórica.

En este capítulo se hace referencia al estado del arte del tema tratado, una breve explicación de la situación problémica que se quiere solucionar. Se describe la justificación de las tecnologías, herramientas y metodologías que se utilizarán para el desarrollo de la aplicación web que dará solución al problema existente.

Capítulo 2: Características del sistema.

En este capítulo se define el objeto de estudio del problema tratado. Se describen los procesos candidatos a automatizar. Se hace una descripción general de cómo debe funcionar el sistema. Se especifican los requisitos funcionales y no funcionales, se identifican los casos de uso y las relaciones con los actores y sus respectivas descripciones textuales.

Capítulo 3: Diseño del sistema.

En este capítulo se muestran todos los elementos básicos del diseño como son los diagramas de clases del diseño, los patrones de diseño utilizados, se explica la arquitectura utilizada, se define el diagrama de despliegue para ver la distribución física de los nodos de cómputo que necesita la aplicación. También se muestran el diagrama de clases persistentes y el modelo de datos.

Capítulo 4: Implementación.

En este capítulo se muestran los artefactos correspondientes al flujo de trabajo de implementación como son el diagrama de componentes y el modelo de despliegue. Se exponen las pruebas (caja negra) que se le realizaron al sistema para comprobar la presencia de errores en el sistema en caso de tenerlos, se describen algunos fragmentos de los códigos fuentes de la aplicación así como algunas de las principales interfaces funcionales de la misma.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza una caracterización del objeto de estudio. Se hace un estudio de las tendencias actuales de la utilización de aplicaciones similares a nivel nacional e internacional. Además se describen las herramientas, tecnologías y metodologías utilizadas para dar solución al problema con el fin de justificar su empleo en el desarrollo de todo el proyecto.

1.2 Sistemas de Gestión de Información

Se entiende por gestión de la información como un proceso que incluye operaciones como: extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma. [1]

La gestión de la información hoy en día se ha convertido en un aspecto fundamental en todas las organizaciones que se quieran vincular a los nuevos avances de las tecnologías de información y comunicación.

El desarrollo acelerado de las tecnologías, acompañado de la renovadora industria del software y la incorporación de sistemas coherentes para la gestión de información y conocimiento; proponen soluciones novedosas para potenciar valores a los denominados recursos intangibles, mejorar estrategias de administración y elevar niveles de eficiencia y eficacia. [2]

La información que actualmente se maneja en las instituciones de diferentes ramas de la sociedad ya no es factible gestionarla de forma manual en enormes volúmenes de documentos, producto a esta situación surgieron numerosos gestores de información que permiten hacer todo esto proceso con mayor rapidez y exactitud.

Durante la fase de inicio del proyecto Mapeo Cerebral Humano Cubano no existía en el mundo un gestor de la información resultante del trabajo con las Imágenes de Resonancia Magnética (MRI). Por esta razón CNEURO solicita la confección de un software que gestione toda la información de las MRI obtenidas en todos los estudios que se realicen a los sujetos.

Una vez realizada la solicitud de CNUERO a la UCI, ambos centros comienzan a trabajar conjuntamente con el objetivo de crear las herramientas necesarias para el desarrollo del proyecto Mapeo Cerebral Humano Cubano. Al realizar un estudio de las posibles variantes de tecnologías con las cuales se podían implementar las herramientas necesitadas por CNEURO, se concluyó que las mismas serían desarrollas sobre tecnologías web por la amplia gama de ventajas que estas nos ofrecen, a continuación se muestran algunas de las ventajas de la tecnología seleccionada respecto a tecnologías de escritorios.

Ventajas de las Aplicaciones Web en comparación con las Aplicaciones de Escritorio.

- **Compatibilidad multiplataforma**. Las aplicaciones web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software descargables. Varias tecnologías incluyendo PHP, Java, Flash, ASP y Ajax permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.
- **Actualización.** Las aplicaciones basadas en web están siempre actualizadas con el último lanzamiento sin requerir que el usuario tome acciones pro-activas, y sin necesitar llamar la atención del usuario o interferir con sus hábitos de trabajo con la esperanza de que va a iniciar nuevas descargas y procedimientos de instalación (algunas veces imposible cuando usted está trabajando dentro de grandes organizaciones).
- **Inmediatez de acceso**. Las aplicaciones basadas en web no necesitan ser descargadas, instaladas y configuradas. Usted accede a su cuenta online y están listas para trabajar sin importar cuál es su configuración o su hardware.
- **Menos requerimientos de memoria**. Las aplicaciones basadas en web tienen muchas más razonables demandas de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del proveedor, a esas aplicaciones basadas en web usa en muchos casos la memoria de las computadoras que ellos corren, dejando más espacio para correr múltiples aplicaciones al mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento.
- **Menos Errores (Bugs).** Las aplicaciones basadas en web deberían ser menos propensas a colgarse y crear problemas técnicos debido a software o conflictos de hardware con otras aplicaciones existentes, protocolos o software personal interno. Con aplicaciones basadas en web, todos utilizan la misma versión, y todos los bugs pueden ser corregidos tan pronto como

son descubiertos. Esta es la razón por la cual las aplicaciones basadas en web deberían tener mucho menos bugs que el software de escritorio descargable tradicional.

- **Precio.** Las aplicaciones basadas en web no requieren la infraestructura de distribución, soporte técnico y marketing requerido por el software descargable tradicional. Esto permite que las aplicaciones online cuesten una fracción de sus contrapartes, mientras que ofrecen componentes adicionales y servicios premium como una opción.
- **Los datos también van online.** Por supuesto con el desplazamiento de las aplicaciones locales a aquellas basadas en web también los datos que creamos y accedemos van a necesitar experimentar profundos cambios. A nadie le gusta no poder acceder a su propio e-mail cuando está de viaje, o poder recuperar un documento particular. Con las aplicaciones web este problema se resuelve ya que se puede acceder a la información desde cualquier parte del mundo.
- **Múltiples usuarios concurrentes**. Las aplicaciones basadas en web pueden realmente ser utilizada por múltiples usuarios al mismo tiempo. No hay más necesidad de compartir pantallas o enviar instantáneas cuando múltiples usuarios pueden ver e incluso editar el mismo documento de manera conjunta. Las compañías de conferencia web y colaboración online están involucradas algunas transformaciones claves y los usuarios necesitan explorar que significa realmente trabajar efectivamente y co-editar documentos juntos.
- **Los datos son más seguros.** Si bien la ruptura de discos no va a desaparecer, es probable que los usuarios escuchen mucho menos del tema. A medida que las compañías se hagan cargo del almacenamiento de los datos del usuario, granjas de almacenamiento de datos redundantes, altamente fiables, será la norma más que la excepción, y los usuarios van a tener mucho menos riesgo de perder sus datos debido a una ruptura de disco impredecible o a un virus de la computadora. Las compañías que provee aplicaciones basadas en web van a brindar amplios servicios de resguardo de datos ya sea como una parte integral del servicio básico o como una opción paga.

Durante el período de desarrollo de las herramientas informáticas propuestas para el proyecto Mapeo Cerebral Humano Cubano, surgió un software Open Source bajo una licencia de estilo BSD con derechos de la Universidad de Harvard llamado XNAT (Extensible Neuroimaging Archive Toolkit), es una plataforma de software diseñada para facilitar el manejo, conservación y consulta de neuroimágenes e información asociada.

El objetivo de los desarrolladores de este software es facilitar el manejo y la conservación de los estudios de neuroimágenes, así como su posterior capacidad para compartir esta información. Un software para un fin específico, pero con el propósito de que sea empleado ampliamente por la comunidad de neurocientíficos sin estar enfocado a un cliente específico. El XNAT es un software orientado al trabajo con neuroimágenes específicamente. No incluye capacidades para el trabajo con otros tipos de estudios neurológicos. Los tipos de imagen que soporta son DICOMS y Analyze.

El sistema propuesto está enfocado a la gestión de la información que maneja CNEURO específicamente, además de los tipos de imágenes que soporta el XNAT incluye el formato Nifti.

1.3 Imágenes por Resonancia Magnética (MRI)

La Resonancia Magnética (MRI) es un procedimiento de diagnóstico que utiliza una combinación de imanes grandes, radiofrecuencias y una computadora para producir imágenes detalladas de los órganos y las estructuras internas del cuerpo.

El estudio de MRI es el más sensitivo para la detección de tumores cerebrales, infartos y ciertos desórdenes crónicos del sistema nervioso como lo es la esclerosis múltiple. También se puede utilizar para documentar problemas mentales en pacientes con demencia y para diagnosticar pacientes con problemas en la glándula pituitaria. El estudio de MRI también puede detectar anormalidades relacionadas a los ojos o los oídos.

La MRI se utiliza a menudo:

- Para examinar el corazón, el cerebro, el hígado, el páncreas, los órganos reproductores femeninos o masculinos, y otros tejidos blandos.
- Para evaluar el flujo sanguíneo.
- Para detectar tumores y diagnosticar muchas formas de cáncer.
- Para evaluar infecciones.
- Para evaluar lesiones en huesos y articulaciones.
- El contraste intravenoso utilizado en *MRI* causa mucho menos reacciones alérgicas que los utilizados en *CT* o rayos x convencional
- No hay exposición a radiación ionizante (rayos-x). [3]

Desventajas:

- Implantes metálicos no identificados pueden ser afectados por los campos magnéticos.

- MRI no se debe utilizar durante las primeras 12 semanas de embarazo**.**
  Ver Anexo 1

1.4 Metodologías de Desarrollo de Software

El desarrollo de software no es sin dudas una tarea fácil. Como resultado a este problema ha surgido una alternativa desde hace mucho: las metodologías.

Las metodologías de desarrollo de software son un marco de trabajo para estructurar, planificar y controlar el proceso de desarrollo de un software. Estas promueven iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Con el empleo de una metodología de desarrollo se garantiza que el proceso de desarrollo del software sea más predecible y eficiente.

1.4.1 OpenUP

Para el desarrollo de este software se utilizó la metodologías OpenUP debido a que este proyecto forma parte del polo de Bioinformática y esta es la metodología por la que el mismo se rige, además de que la misma es abierta como su nombre lo indica, adaptable a las necesidades del software que se está desarrollando y a que este proyecto es de corta duración.

Este proceso de desarrollo unificado está basado en **Rational Unified Process (RUP)** o **Proceso Racional unificado (PUR)** desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad, basándose en los principios de Adaptación, Importancia a los involucrados e interesados en los resultados del proyecto; Colaboración, Valor a la iteración; y Calidad Continua.

OpenUP/Basic es un framework de procesos de desarrollo de software de código abierto, que con el tiempo espera cubrir un amplio conjunto de necesidades en el campo del desarrollo de software. OpenUP/Basic permite un abordaje ágil al proceso de desarrollo de software, con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles, y tareas. OpenUP/Basic es un proceso interactivo de desarrollo de software simplificado, completo y extensible. Es un proceso para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias. OpenUP está caracterizado por cuatro principios básicos interrelacionados, a saber:

- **Colaboración** para unificar intereses y compartir conocimientos.

- **Equilibrio** de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- Enfoque en la **articulación** de la arquitectura.
- **Desarrollo continuo** para obtener realimentación y realizar las mejoras respectivas.

Utiliza el lenguaje unificado de modelado (UML) para representar todos los esquemas de un sistema de software.

OpenUP divide en 4 fases el desarrollo del software:

- **Inicio:** El objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración:** En esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción:** En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transición:** El objetivo es llegar a obtener la liberación del proyecto.

1.5 Lenguajes de Programación

Un lenguaje de programación es un conjunto de símbolos y reglas tanto semánticas como sintácticas que definen una estructura. Estos son utilizados para controlar el comportamiento físico y lógico de una máquina.

Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

Para la confección de este sistema se utilizaron lenguajes de programación como Java y JSP.

1.5.1 Java Server Page (JSP)

Java Server Pages TM (JSP) es un conjunto de tecnologías que permiten la generación dinámica de páginas web combinando código Java (scriptlets) con un lenguaje de marcas como HTML ó XML, para generar el contenido de la página.

Como parte de la familia de la tecnología Java, con JSP se pueden desarrollar aplicaciones web independientes de la plataforma. Una característica importante es que permite separar la interfaz del usuario de la generación del contenido dinámico, dando lugar a procesos de desarrollo más rápidos y eficientes.

Adicionalmente, pueden acceder directamente a componentes Java Beans ó Enterprise Java Beans (EJB), instanciándolos y estableciendo sus propiedades e invocando sus métodos directamente desde la página JSP. Esto permite desarrollar aplicaciones n-capas donde se separan en lo posible los datos, la lógica del negocio y la lógica de presentación, encapsulando, generalmente, en Beans el acceso a los datos.

La tecnología JSP es una extensión de la tecnología Servlets, los cuales son aplicaciones 100% Java que corren en el servidor: Se crea e inicia un Servlet, se procesan las peticiones recibidas y por último se destruye. Este diseño explica por qué un Servlet remplaza perfectamente a un CGI, ya que el servlet se carga una sola vez y está residente en memoria mientras se procesan las peticiones recibidas y se generan las respuestas a los usuarios.

Cada vez que un cliente solicita al servidor web una página JSP, este pasa la petición al motor de JSP el cual verifica si la página no se ha ejecutado antes ó fue modificada después de la última compilación, tras lo cual la compila, convirtiéndola en Servlet, la ejecuta y devuelve los resultados al cliente en formato HTML.

La especificación JSP es el producto de una colaboración amplia de varias de las industrias líderes en el desarrollo de software, liderados por Sun Microsystems. Lo importante fue que Sun hizo la especificación de JSP disponible libremente para la comunidad de desarrollo de software, con la idea de que todos los servidores web soporten JSP, compartiendo la característica de la tecnología Java "Write Once, Run Anywhere" (Escríbelo una vez, córrelo donde quieras). Es conveniente resaltar, que la tecnología JSP es un componente clave de la plataforma Java 2 Enterprise Edition (J2EE) propuesta por Sun Microsystems.

En resumen, las tecnologías JSP y Servlets son una alternativa importante para la programación de web de contenido dinámico que nos permiten:

- Independencia de la plataforma
- Rendimiento mejorado
- Separación de la lógica de la aplicación de la presentación de los datos
- Uso de componentes (Java Beans)
- Facilidad de administración y uso
- El respaldo importante de la tecnología sólida Java TM. [4]

1.5.2 Java

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes:

- Es un lenguaje que se compila, generando ficheros de clases compiladas, pero estas clases compiladas, son en realidad interpretadas por la máquina virtual de java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- Gracias al API de java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo Windows, ... [5]

Java es un lenguaje robusto que verifica su código al mismo tiempo que lo escribe, y una vez más antes de ejecutarse, de manera que se consigue un alto margen de codificación sin errores. Se realiza un descubrimiento de la mayor parte de los errores durante el tiempo de compilación, ya que Java es estricto en cuanto a tipos y declaraciones, y así lo que es rigidez y falta de flexibilidad se convierte en eficacia. Respecto a la gestión de memoria, Java libera al programador del compromiso de tener que controlar especialmente la asignación que de ésta hace a sus necesidades específicas. Este lenguaje posee una gestión avanzada de memoria llamada gestión de basura, y un manejo de excepciones orientado a objetos integrados. Estos elementos realizarán muchas tareas antes tediosas a la vez que obligadas para el programador. [6]

1.6 Servidor de Aplicaciones

1.6.1 Apache Tomcat

Tomcat es el contenedor de servlets, mediante el cual puede "correr" Servlets de Java y JSP que son los lenguajes que se utilizarán para la confección de la aplicación. Tomcat es gratis, es fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java. Puede descargarse, instalarse y probarse en el iSeries en menos de una hora. Tomcat ocupa muy poco

espacio, teniendo su código binario (todo clases de Java) un tamaño total de apenas un megabyte, de modo que no es raro que se ejecute tan deprisa.

Otra ventaja de Tomcat es que es muy fiable. Innumerables empresas utilizan Tomcat (aunque es imposible saber cuántas debido a la falta de licencia comercial). Citando a Linux Torvalds acerca del código libre, "…*dado un número suficiente de ojos, todos los errores son irrelevantes*…". Lo que significa es que si el número de usuarios es lo bastante grande, siempre habrá alguien capaz de arreglar lo que los demás pueden pensar que es un error muy complejo. Dicho de otra forma, la solidez de Tomcat se basa en que miles de desarrolladores contribuyen con código. Tomcat pone a disposición de todo el mundo las últimas actualizaciones de Java. Además, como Tomcat puede ejecutarse utilizando la JVM que se quiera, puede utilizarse JDK 1.4 (por cierto que Java 1.4 ofrece aserciones, expresiones regulares, mejor rendimiento que Java 1.3 y una infraestructura de registro cronológico estándar). Tomcat ahora tiene seguridad de nivel de aplicación, una aplicación de administración intuitiva basada en web, expresiones regulares compatibles con JDK 1.2 y 1.3 y, como se mencionaba antes, mejor escalabilidad y rendimiento. [7]

1.7 Gestor de Bases de Datos

Una base de datos (BD) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Diferentes programas y diferentes usuarios pueden utilizar estos datos. Proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado. Se convierte más útil a medida que la cantidad de datos almacenados crece. La ventaja principal de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo.

Rápidamente surgió la necesidad de contar con un sistema de administración para controlar tanto los datos como los usuarios. La administración de bases de datos se realiza con un sistema llamado DBMS (Database Management System). El DBMS es un conjunto de servicios (aplicaciones de software) para administrar bases de datos, que permite:

- Un fácil acceso a los datos
- El acceso a la información por parte de múltiples usuarios
- La manipulación de los datos encontrados en la base de datos (insertar, eliminar, editar) [8]

Existen distintos gestores de base de datos como por ejemplo:

- Oracle
- MySQL
- PostgreSQL

1.7.1 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

1.7.2 Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

Ha sido criticada por algunos especialistas la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

1.7.3 PostGreSQL

- PostGreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado de varias formas desde la década de 1980.
- El proyecto PostGreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.
- PostGreSQL es ampliamente considerado como una de las alternativas de sistema de bases de datos de código abierto.

Ventajas de PostgreSQL

- **Instalación Ilimitada**

Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.

- **Soporte**

Además de las ofertas de soporte, se cuenta con una comunidad importante de profesionales y entusiastas de PostgreSQL de los que su compañía puede obtener beneficios.

- **Ahorros considerables en costos de operación**

PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.

- **Estabilidad y Confiabilidad Legendarias**

  Es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.

- **Extensible**

El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto

es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.

- **Multiplataforma**

PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows.

- **Diseñado para ambientes de alto volumen**

PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones. [9]

En este proyecto es necesario manejar una gran cantidad de información por lo que es indispensable un manejador de base de datos que sea capaz de trabajar con grandes volúmenes de datos. PostgreSQL administra grandes cantidades de datos, el cual es considerado como una de las bases de datos de código abierto (Open Source) más avanzada del mundo. Con el empleo de este gestor se garantiza integridad de los datos y la velocidad de acceso y consultas a la base de datos. Son estas las características que nos han encaminado a utilizarlo para este proyecto.

1.8 FrameWork de Aplicación

El framework puede ser definido como un conjunto de clases (algunas de ellas generalmente abstractas), y las colaboraciones que se establecen entre ellas, para proporcionar un diseño abstracto para las soluciones de un conjunto de problemas.

El framework captura las decisiones de diseño comunes a un tipo de aplicación, estableciendo un modelo común a todas ellas, asignando responsabilidades y estableciendo colaboraciones entre las clases que forman el modelo.

Frameworks de Aplicación

Un framework de aplicación encapsula una capa de funcionalidad horizontal que puede ser aplicada en la construcción de una gran variedad de programas.

Frameworks de Dominio

Un framework de dominio implementa una capa de funcionalidad vertical, correspondiéndose con un dominio de aplicación o una línea de producto. Su evolución deberá ser también la más rápida, pues deben adaptarse a las áreas de negocio para las que están diseñados.

Ventajas en la utilización de Frameworks

Se logra buen nivel de reutilización (no solo a nivel de código), lo que implica:

Reducción en el tiempo de desarrollo de nuevos aplicativos.

Reducción del costo de mantenimiento.

Mayor nivel de confiabilidad (comparado con escribir código nuevo), en la medida que hay reutilización y el framework se estabiliza.

Estandarización y Consistencia

Es posible encapsular estándares y mejores prácticas de la compañía en un framework, logrando consistencia y aseguramiento de uso de dichos estándares y mejores prácticas

Dificultades con los Frameworks

Los frameworks no son reutilizables por si solos, y cuando se diseña e implementan soluciones con la reutilización en mente, el tiempo y los costos en general, son mayores.

Esto debe ser visto como una inversión, ya que en la medida que el framework se reutilice, aparecerán los beneficios.

Requieren de programadores más experimentados para su desarrollo, que una aplicación común

Requieren de buena documentación y de entrenamiento para los desarrolladores que lo utilizan. [10]

1.8.1 Spring

Spring es un framework de aplicación desarrollado por la compañía Interface 21, para aplicaciones escritas en el lenguaje de programación Java. Fue creado gracias a la colaboración de grandes programadores, entre ellos se encuentran como principales partícipes y líderes de este proyecto Rod Johnson y Jürgen Höller. Estos dos desarrolladores, además de otros colaboradores que juntando toda sus experiencias en el desarrollo de aplicaciones J2EE (Java 2 Enterprise Editions), incluyendo

EJB(Enterprise JavaBeans), Servlets y JSP, lograron combinar dichas herramientas y otras más en un solo paquete para brindar una estructura más sólida y un mejor soporte para este tipo de aplicaciones.

Además se considera a Spring un framework lightweight, es decir liviano o ligero, ya que no es una aplicación que requiere de muchos recursos para su ejecución, además el framework completo puede ser distribuido en un archivo de alrededor de 1MB, lo cual representa muy poco espacio, y para la cantidad de servicios que ofrece es relativamente insignificantes su tamaño.

Este framework se encuentra actualmente en su versión 1.2.5, aunque es una versión temprana, está adquiriendo gran auge y una gran popularidad. Una de las características que ayuda a este éxito, es que es una aplicación open source, lo cual implica que no tiene ningún costo, ni se necesita una licencia para utilizarlo, por lo tanto da la libertad a muchas empresas y desarrolladores a incursionar en la utilización de esta aplicación.

Spring fue creado basado en los siguientes principios:

- El buen diseño es más importante que la tecnología subyacente.
- Los JavaBeans ligados de una manera más libre entre interfaces es un buen modelo
- El código debe ser fácil de probar

Spring, además, permite configurar las clases en un archivo XML y definir en él las dependencias. De esta forma la aplicación se vuelve muy modular y a la vez no adquiere dependencias con Spring. Cuenta con plantillas de utilidades para Hibernate, Ibatis y JDBC, así como la integración con Struts, JSF y otros frameworks.

Es uno de los proyectos más sorprendentes en el panorama actual de Java, en el grado en que ayuda a que los diferentes componentes que forman una aplicación trabajen entre sí, pero no establece apenas dependencias consigo mismo. Esta es la primera característica de este framework. Sería posible retirarlo sin prácticamente cambiar líneas de código. Lo único que sería necesario es, lógicamente, añadir la funcionalidad que provee, ya sea con otro framework principal o con el código del propio desarrollador. A nivel de soporte de la comunidad, Spring es uno de los proyectos con mayor actividad, con desarrollo dentro y fuera del framework.

Anteriormente se dijo que Spring era un conjunto de librerías o módulos. Pues bien, entrando un poco dentro de Spring se puede decir que es un framework compuesto por varios frameworks más pequeños, los cuales son:

- Contenedor de inversión de control: la configuración de los componentes de la aplicación y el ciclo de vida es manipulado básicamente por objetos Java comunes.
- Programación orientada a aspectos: el trabajo con las funcionalidades que no pueden ser implementadas mediante la programación orientada a objetos sin hacer sacrificios de rendimiento y estabilidad.
- Framework de acceso a datos: trabaja con sistemas de administración objeto-relacionales que proveen soluciones para los desafíos técnicos reusables en multitudes de ambientes basados en Java.
- Manipulación de transacciones: armonización de varios APIs manipuladores de transacciones orquestados por objetos Java.
- Modelo-Vista-Controlador: framework basado en HTTP y Servlets que provee extensiones y personalizaciones.
- Framework de acceso remoto: soporte para aplicaciones basadas en el protocolo HTTP, tales como RMI, CORBA y Servicios Web o Web Services (SOAP).[11]

Básicamente, la estructura de los módulos de Spring queda como sigue:

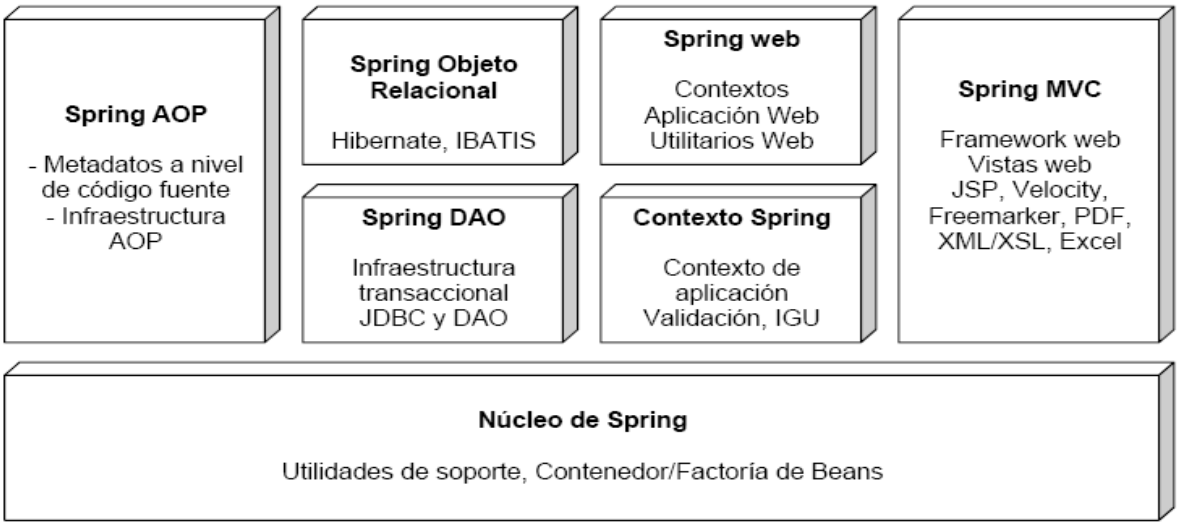


Figura 1

1.8.2 Hibernate

Hibernate es una herramienta de mapeo objeto/relacional para ambientes Java. El término "mapeo objeto/relacional" (ORM por sus siglas en inglés) se refiere a esta técnica de "mapear" la representación de los datos desde un modelo de objetos hacia un modelo de datos relacional, con un esquema de base de datos basado en SQL.

Hibernate no sólo se hace cargo del mapeo de clases Java a las tablas de una base de datos (y de los tipos Java a los tipos de la base de datos), sino que también provee utilidades para consulta y captura de datos, y puede reducir considerablemente el tiempo que, de otra manera, habría que invertir con el manejo manual de datos mediante SQL y JDBC.

La meta de Hibernate es aliviar al programador del 95% de las tareas más comunes relacionadas con persistencia. Probablemente, Hibernate no sea la mejor solución para aplicaciones data-céntricas que tengan casi toda su lógica de negocios en procedimientos almacenados (stored procedures) en la base de datos; es más útil con modelos orientados a objetos cuya lógica de negocio reside en la capa intermedia. Sin embargo, Hibernate puede ayudarlo a encapsular o eliminar código SQL que sea específico de un proveedor de BD, y ayudará en la tarea usual de traducir desde una representación tabular a un gráfico de objetos.

Las principales características de Hibernate se pueden resumir como sigue:

- Ocultamiento de objetos: la sesión a nivel de transacción que oculta los objetos persistentes.
- La ejecución de sentencias se realiza solo cuando son necesarias, de manera que si ocurre una excepción y es necesario abortar una transacción muchas declaraciones nunca habrían sido ejecutadas. Además, esto hace que los tiempos de acceso a las bases de datos sea el mínimo posible.
- La no actualización de objetos no modificados. Es muy común ver, en aplicaciones que utilizan código a mano JDBC, los estados persistentes de los objetos actualizados, por ejemplo, si se presiona el botón “guardar” pero no se han hecho modificaciones. Hibernate es capaz de saber en todo momento si el estado de un objeto ha cambiado, quitando, de esta manera viajes innecesarios al servidor de bases de datos.
- En consecuencia con el punto anterior, Hibernate solo manipulará las colecciones de datos si estas realmente han cambiado.

- Hibernate puede enrollar dos actualizaciones aparentemente distintas en una del mismo objeto en la misma declaración de actualización.
- Hibernate implementa un muy eficiente algoritmo de búsqueda.
- Inicialización de colecciones perezosas.
- Inicialización de objetos perezosos.
- Hibernate puede ser utilizado tanto en la web como en aplicaciones convencionales. Esto no solo brinda la inversión de tiempo en Hibernate de cara al futuro, sino que, de ser útil, hace a los desarrolladores totalmente independientes del mismo.

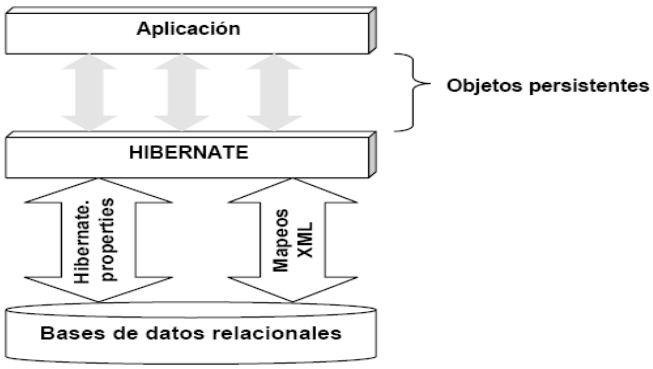


Figura 2

Arquitectura base de Hibernate. [12]

1.9 Herramientas CASE

CASE es una sigla, que corresponde a las iníciales de: Computer Aided Software Engineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación.

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. [13]

Las herramientas CASE aseguran que se alcanzan las tres C´s:

- Consistencia

- Complitud
- Cumplimiento de los estándares.

Mejora de la productividad y de la calidad, mediante un entorno interactivo.

Aceleración del proceso de desarrollo, favoreciendo el proceso de prototípico (espiral)

Automatizar e integrar las tareas de las distintas etapas del ciclo de vida.

Asistencia en la gestión de proyectos software.

Mejora de la calidad del software (automatización comprobación de errores).

Automatizar la generación de documentación.

Existen herramientas Case de trabajos visuales como DBDesigner 4, Rational Rose, Embarcadero ER/Studio, GNU Ferret, MagicDraw, Umbrello, ArgoUML y Visual Paradigm que permiten realizar el modelado del desarrollo de los proyectos. A continuación se explican algunas de las herramientas CASE más utilizadas. [14]

1.9.1 Visual Paradigm for UML (Unified Modeling Language)

Es una herramienta CASE que utiliza “UML”: como lenguaje de modelaje. Se integra con las siguientes herramientas Java:

Eclipse/IBM WebSphere

JBuilder

NetBeans IDE

Oracle JDeveloper

BEA Weblogic

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una construcción de aplicaciones de calidad más rápida,

mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. [15]

El equipo de desarrollo utiliza para la realización del trabajo la plataforma GNU/Linux, es por esta razón que se utiliza como Herramienta CASE el Visual Paradigm y además es un software del la universidad posee licencia.

1.9.2 DBDesigner Fork

Es un sistema de diseño de base de datos disponible libre que integra el diseño de base de datos, el modelado, la creación y el mantenimiento en un ambiente simple y perfecto.

Esta herramienta está disponible o soporta los sistemas operativos Windows XP y GNU/Linux, se puede acceder a la descarga gratis bajo la licencia GNU GPL. Para quienes utilizan MySQL como motor de base de datos, DBDesigner 4 es la mejor opción ya que ha sido desarrollado y optimizado para MySQL.

Esta herramienta soporta multitud de opciones, entre las que se encuentran:

- Modo de diseño o consulta.
- Ingeniería inversa de bases de datos MySQL, Oracle, MSSQL y cualquiera con driver ODBC.
- Control de versiones.
- Constructor de queries.
- Soporte especial para MySQL.

Es un producto gratuito bajo licencia GPL.

1.10 Conclusiones

En este capítulo se hace un estudio de las tendencias actuales y se proponen las soluciones que más se pueden ajustar a el desarrollo de este sistema. Se determinan las tecnologías y herramientas que se usarán. Se llega a la conclusión de que la metodología usada será OpenUP con UML como lenguaje de modelado. Como gestor de Bases de Datos PostgreSQL para almacenar adecuadamente la información generada. Se utilizará Eclipse para la implementación de la aplicación, DBDesigner Fork para el diseño de la Base de Datos del sistema y Visual Paradigms for UML como herramienta de modelado.

Capítulo 2: Características del Sistema

2.1 Introducción

Con el desarrollo de este capítulo conocerá como quedó definido el modelo de dominio asimismo se describe cómo quedará la solución propuesta a la situación problémica planteada. Se plantean las características y funcionalidades que el sistema debe tener a través de requisitos funcionales y no funcionales, los casos de uso correspondientes y sus respectivas descripciones así como el diagrama de casos usos anteriormente mencionados.

2.2 Modelo del dominio

En CNEURO los procesos no están bien identificados y el flujo de acciones es muy sencillo, por esta razón, para comprender la estructura y la dinámica de esta organización en la cual se va a implantar el sistema así como asegurar que los clientes y desarrolladores tengan un entendimiento común del funcionamiento de la misma se realizó un modelo de dominio que captura los tipos de objetos y conceptos más importantes de este entorno.

2.2.1 Descripción textual del modelo de dominio

En el procesamiento de las imágenes participan una serie de elementos como son el equipo de MRI, las imágenes, la computadora personal del investigador y las herramientas con las cuales se procesan las imágenes.

El sujeto va y se realiza la prueba en el equipo de Resonancia Magnética donde quedan las imágenes correspondientes.

El investigador se dirige a la entidad donde se encuentra el equipo de MRI y recoge las imágenes de resonancia magnética en una computadora personal. Luego se dirige al centro de Neurociencia y allí utilizando herramientas como DICOMTool, IBASPM se realiza el procesamiento y se visualizan los resultados finales con el MRICROM.

2.2.2 Modelo de Dominio

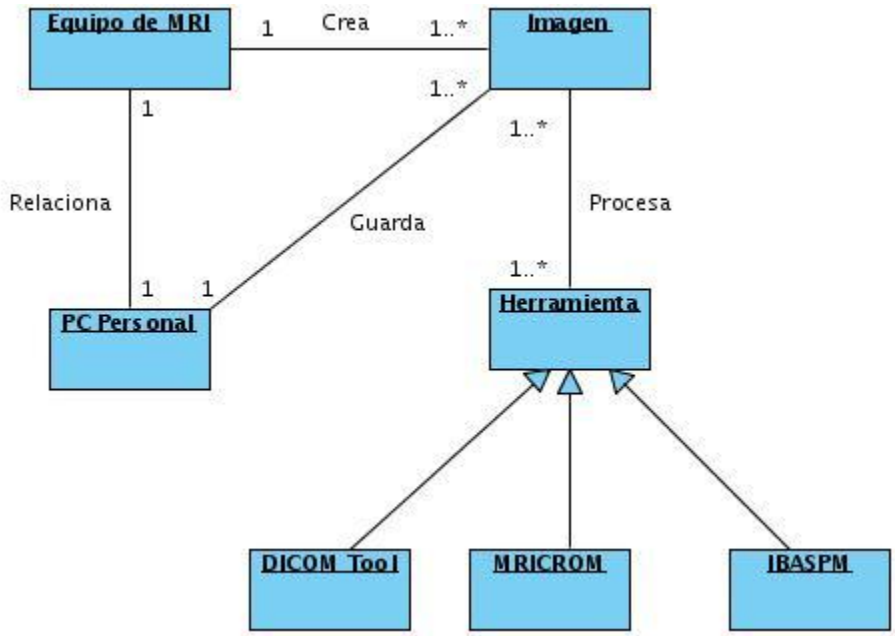


Figura 3

2.3 Especificación de los Requerimientos de la aplicación.

2.3.1 Requerimientos funcionales

Estos requerimientos no son más que capacidades o condiciones que el sistema debe cumplir para satisfacer las necesidades del cliente.

1. Manipular Información de Imágenes

1.1 Mostrar listado de imágenes

1.2 Visualizar imagen

2. Manejar estudios

2.1 Visualizar cantidad de estudios realizados

2.2 Mostrar listado de estudios realizados

3. Autenticar Usuario.

4. Manejar información de sujeto.

4.1 Buscar sujeto

4.2 Mostrar listado de sujetos

4.3 Mostrar expediente del sujeto

5. Generar Reporte de direcciones físicas de las imágenes

6. Realizar Búsqueda Avanzada de imágenes.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son aquellos que consisten en propiedades o cualidades que debe tener el producto final. Estos son los que hacen el software más agradable, atractivo, usable y eficiente en cuanto rendimiento y seguridad a la vista del usuario final.

Apariencia e Interfaz:

El sistema informático contará con un diseño de interfaz sencillo, de fácil entendimiento. Cada una de las páginas de la aplicación estará poco cargada, sólo con la información requerida para el usuario.

Seguridad:

La seguridad del la aplicación está sujeta a la seguridad del sistema generar del proyecto debido a que el sistema propuesto es un módulo del mismo. Independientemente de la seguridad del sistema general, la aplicación cuenta con un sistema de seguridad propio basado en un solo nivel de acceso, además la aplicación web no permite modificar ni borrar la información gestionada.

Usabilidad:

El sistema permitirá un acceso fácil y rápido. Toda aquella persona que cuente con conocimientos básicos del manejo de una computadora y un navegador Web podrá usarlo.

Confiabilidad:

El sistema solo será usado por el personal del Centro de Neurociencia y este validará todos los datos que se manejen para evitar entrada inadecuada de los mismos.

Soporte:

Cuando el sistema sea instalado en el Centro de Neurociencia se impartirá un curso de adiestramiento y familiarización con el mismo. También se le dará seguimiento al funcionamiento de la aplicación.

Legales:

Se estarán usando herramientas de software libre, licencia GNU/GPL y herramientas de las que se posea licencia.

Portabilidad:

El sistema es adaptable a diferentes entornos como GNU/Linux o Windows sin necesidad de requerir herramientas adicionales. Además es compatible con las diferentes versiones de los navegadores existentes en los entornos anteriores.

Hardware:

Se requiere de una PC cliente de al menos 256 MB de RAM. Se deberá contar con impresora en las computadoras clientes que interactúen con la aplicación. El servidor debe tener las siguientes características: capacidad de disco duro superior a 80.0 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

Software:

Se deberá disponer, para instalar la aplicación, del Sistema Operativo Windows 98 o superior, o cualquier distribución de Linux. Las computadoras clientes de los usuarios accederán al sistema usando uno de los siguientes navegadores: Internet Explorer o Mozilla Firefox. Para el servidor de la aplicación el sistema operativo recomendado es GNU/Linux. Se debe instalar un servidor Web Apache Tomcat. Para la Base de Datos del sistema se usará PostgreSQL.

Restricciones en el diseño y la implementación:

Se debe usar Java y JSP como lenguajes de programación. Como entornos integrados de desarrollo (IDE por sus siglas en inglés) se utilizarán Eclipse Europe 3.3, Framework Spring 2.5, Framework Hibernate; UML como lenguaje de modelado y como herramienta CASE para el modelado de los artefactos Visual Paradigm. Se empleará para el trabajo con la base de datos DBDesigner Fork para el diseño y PostgreSQL como gestor. Además como servidor de aplicaciones se utilizará el Apache Tomcat 5.5 o una versión superior.

Políticos y Culturales:

La aplicación utilizará el español como idioma. Algún cambio que se quiera hacer, será a través de la dirección del proyecto que desarrolla esta investigación.

2.4 Modelo de Casos de Usos del Sistema

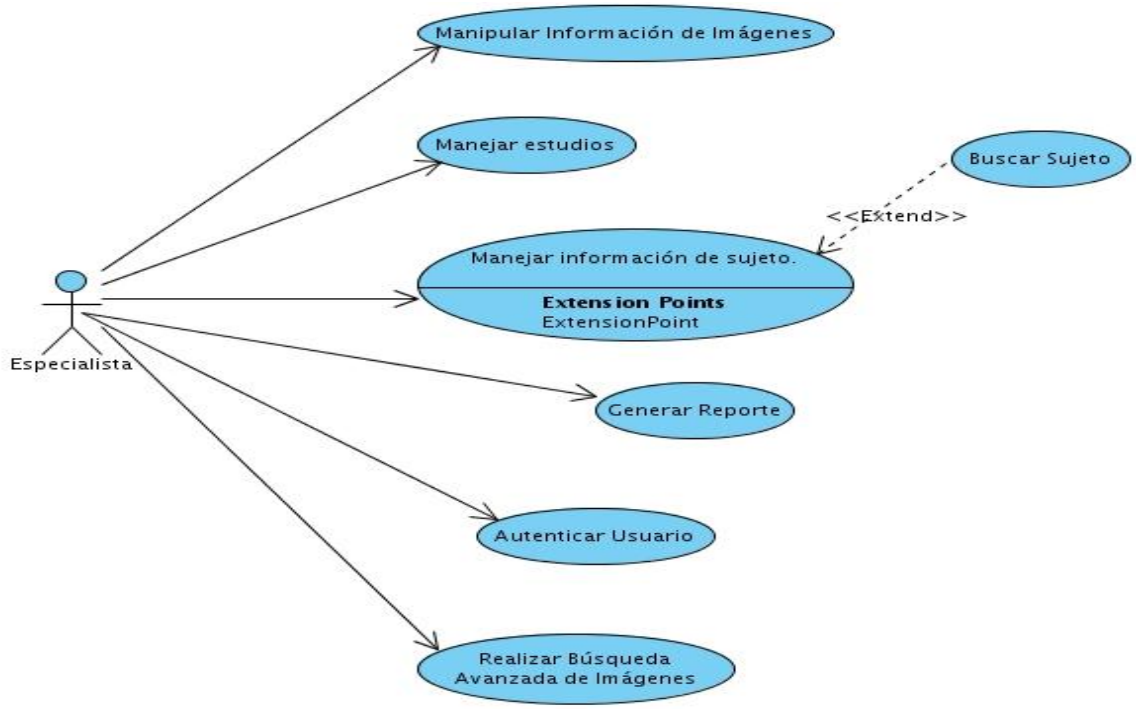


Figura 4

2.5 Descripción textual de los casos de usos del sistema

Descripción textual caso de uso Autenticar Usuario

Nombre del Caso de Uso	Autenticar Usuario
Actores	Especialista
Propósito	Entrar al sistema para realizar cierta operación.
Resumen	El Caso de Uso se inicia cuando el usuario introduce los datos que se le piden para acceder a la aplicación, estos se verifican y finaliza dándole los permisos según los roles y habilitándole la entrada. El caso de uso termina cuando el usuario entra al sistema
Referencias	3
Precondiciones	
Poscondiciones	Se habilitan las funcionalidades según lo privilegios.
Requerimientos especiales	-
Prototipo	Consultar Expediente de Proyecto

Descripción textual caso de uso Generar Reportes

Nombre del Caso de Uso	Generar Reportes
Actores	Especialista
Propósito	Muestra todos los reportes acerca de las direcciones físicas de las imágenes
Resumen	El caso de uso se inicia cuando el especialista va a realizar alguna de las siguiente operaciones:

	Generar Reporte. El sistema le muestra la interfaz correspondiente según su solicitud con todos los reportes de las direcciones físicas de las imágenes.
Referencias	5
Precondiciones	Que el especialista esté autenticado en la aplicación.
Poscondiciones	
Requerimientos especiales	-
Prototipo	Consultar Expediente de Proyecto

Descripción textual caso de uso Manipular Información de Imágenes

Nombre del Caso de Uso	Manipular Información de Imágenes
Actores	Especialista
Propósito	Llevar a cabo una serie de operaciones con las imágenes obtenidas en el estudio como: Visualizar Imagen, Buscar Imagen, etc.
Resumen	El caso de uso se inicia cuando el especialista va a realizar alguna de las siguiente operaciones: 1. Mostrar listado de imágenes. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Referencias	1.1, 1.2
Precondiciones	Que el especialista esté autenticado en la aplicación.
Poscondiciones	El sistema muestra la imagen o el listado de imágenes solicitado
Requerimientos especiales	-
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. El especialista, quiere realizar la siguiente operación: Mostrar listado de imágenes	2. El sistema, en dependencia de la operación que solicita realizar, hace lo siguiente: • Si decide Mostrar listado de imágenes, ir a la sección: “Mostrar listado”.
Sección “Mostrar listado”	
Acción del actor	Respuesta del Sistema
	1. El sistema muestra la interfaz correspondiente a Mostrar listado.

2. El especialista selecciona el tipo de imagen a listar que pueden ser: • Anatómica • Difusión • Fase • Magnitud	3. El sistema muestra el listado del tipo de imagen seleccionado dándole al especialista la posibilidad de visualizarla.
4. El especialista selecciona la siguiente opción. • Visualizar Imagen	5. El sistema visualiza la imagen seleccionada.
Flujo Alterno de la Sección “Mostrar listado ”	
	3.1 Si la imagen no existe el sistema muestra un mensaje de notificación. Ir al paso 2 de la sección “Mostrar listado”
4.1 El técnico no realiza ninguna opción y sale de la sección.	
Prototipo	Consultar Expediente de Proyecto

Descripción textual caso de uso Manejar Estudios

Nombre del Caso de Uso	Manejar estudios
Actores	Especialista
Propósito	Llevar a cabo una serie de operaciones con los estudios realizados como: Mostrar cantidad de estudios, Mostrar listado de estudios realizados, etc.
Resumen	El caso de uso se inicia cuando el especialista selecciona: • Mostrar listado de estudios realizados El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	2.1, 2.2
Precondiciones	Que el especialista esté autenticado en la aplicación.
Poscondiciones	El sistema muestra la cantidad de estudios realizados o el listado de los estudios.
Requerimientos especiales	-

Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. El especialista, quiere realizar la siguiente operación: • Mostrar listado de estudios realizados	2. El sistema, en dependencia de la operación que solicita realizar, hace lo siguiente: • Si decide Mostrar listado de estudios realizados, ir a la sección: “Mostrar listado”.

Sección “Mostrar listado”	
Acción del actor	Respuesta del Sistema
	1. El sistema muestra la interfaz correspondiente a Mostrar listado.
2. El especialista selecciona la opción mostrar el listado de los estudios	3. El sistema muestra el listado de los estudios realizados dando la posibilidad de visualizar las imágenes correspondientes a este estudio
4. El especialista selecciona la opción de visualizar	5. El sistema le da la opción de escoger que tipo de imagen quiere ver
6. El especialista escoge el tipo de imagen que pueden ser: • Anatómica • Difusión • Fase • Magnitud	7. El sistema visualiza la imagen seleccionada.
Sección “Visualizar Cantidad”	
	1. El sistema muestra el número de estudios que se han realizados.

Flujo Alterno de la Sección “Mostrar listado”	
4.1 El especialista no selecciona ninguna opción y sale de la sección.	
Prototipo	Consultar Expediente de Proyecto

Descripción textual caso de uso Manejar información de sujeto.

Nombre del Caso de Uso	**Manejar información de sujeto.**
Actores	Especialista
Propósito	Buscar un sujeto del estudio
Resumen	El caso de uso se inicia cuando el especialista va a realizar alguna de las siguiente operaciones: 1. Mostrar Listado de Sujetos. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	4
Precondiciones	Que el especialista esté autenticado en la aplicación.
Poscondiciones	El sistema muestra el sujeto solicitado
Requerimientos especiales	-
Flujo normal de los eventos	
Sección “Mostrar listado de Sujetos”	

Acción del actor	Respuesta del Sistema
1.Elespecialista selecciona la opción • Mostrar listado de Sujetos	2. El sistema muestra el listado de sujetos dando la posibilidad de: • Buscar Sujeto • Mostrar expediente de sujeto
3. El especialista selecciona una de las siguientes opciones • Buscar Sujeto • Mostrar expediente de sujeto	4. En dependencia de la opción que se selecciona el sistema hace lo siguiente: • Si selecciona Buscar Sujeto ir a la sección “Buscar Sujeto” • Si selecciona Mostrar expediente de sujeto ir a la sección “Mostrar expediente de sujeto”
Sección “Buscar Sujeto”	
1. El especialista inserta el ID del sujeto que desea buscar.	2. El sistema muestra el sujeto solicitado con su expediente.
Flujo Alterno de la Sección “Buscar Sujeto”	
1.1 Si el especialista no desea buscar ningún sujeto sale de la sección	2.1 Si el sujeto no existe el sistema muestra un mensaje de notificación. Ir al paso 2 de la sección “Buscar Sujeto”
Sección “Mostrar expediente de sujeto”	
1. El especialista selecciona el expediente de que sujeto desea ver.	2. El sistema muestra el expediente del sujeto seleccionado dando la posibilidad de visualizar las imágenes de los estudios que se

	le han realizado.
Prototipo	Consultar Expediente de Proyecto

Descripción textual del caso de uso Realizar Búsqueda Avanzada de Imágenes.

Nombre del Caso de Uso	Realizar Búsqueda Avanzada de Imágenes.
Actores	Especialista
Propósito	Realizar la búsqueda de imágenes teniendo en cuanto determinados parámetros como: raza, sexo, edad, manualidad, etc.
Resumen	El caso de uso se inicia cuando el especialista va a realizar alguna de las siguiente operaciones: 1. Realizar Búsqueda Avanzada. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	6
Precondiciones	Que el especialista esté autenticado en la aplicación.
Poscondiciones	El sistema muestra la imagen o el listado de imágenes solicitado
Requerimientos especiales	-
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema

1. El especialista, quiere realizar la siguiente operación: Búsqueda Avanzada	2. El sistema, muestra la interfaz correspondiente a esta operación
3. El especialista llena los campos correspondientes a los parámetros de búsqueda, que pueden ser: • Sexo • Raza • Edad • Manualidad • Tipo de Imagen	4. El sistema muestra la o las imágenes que dio como resultado la búsqueda.
Flujo alterno	
3,1 El especialista no llena ninguno de los campos porque no desea realizar la búsqueda y sale de la sección.	
Prototipo	Consultar Expediente de Proyecto

2.6 Conclusiones

En este capítulo se definieron las funcionalidades principales con las que debe contar el sistema, los casos de usos y sus relaciones reflejadas en el diagrama de casos de uso, así como sus descripciones que le dan cumplimiento a los requisitos funcionales especificados.

Capítulo 3: Diseño del sistema

3.1 Introducción

En este capítulo de confeccionan una serie de artefactos correspondientes al flujo de trabajo de diseño donde se describe cómo se va a implementar el sistema entre ellos se encuentran los diagramas de clases del diseño por cada caso de uso así como los diagramas de secuencia correspondientes. También se describe la arquitectura utilizada y los principales patrones arquitectónicos que se tendrán en cuenta.

3.2 Patrones de Arquitectura y Patrones de Diseño

"Cada patrón describe un problema que ocurre una y otra vez en un entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma."

Se deben mencionar 2 principios que hay que tener en cuenta a la hora de usar patrones:

- **Los patrones son un punto de partida, no un destino.**
- **Los modelos no están bien o mal, sino que son más o menos útiles.**

"Los patrones le ayudan a construir sobre la experiencia colectiva de ingenieros de software experimentados. Estos capturan la experiencia existente y que ha demostrado ser exitosa en el desarrollo de software, y ayudan a promover las buenas prácticas de diseño. Cada patrón aborda un problema específico y recurrente en el diseño o implementación de un sistema de software." [16]

El principal objetivo que persigue la utilización de patrones no es más que lograr un lenguaje común entre los diseñadores y desarrolladores para comunicar experiencias.

Durante el diseño de este sistema se utilizó el patrón Modelo-Vista-Controlador (MVC).

Es muy frecuente que se solicite cambio a interfaz. Los cambios a interfaz deberían ser fáciles y efectuados en tiempo de ejecución. El cambio de interfaz no debería de tener consecuencias para el núcleo del código de la aplicación.

Con el empleo de este patrón el sistema se divide en tres partes: procesamiento, entradas y salidas.

Modelo – encapsula los datos y la funcionalidad de la aplicación.

Vista – despliega la información contenida en el modelo (pueden existir varias vistas).

Controlador – está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del Modelo o de Vista. El usuario interactúa con el sistema solamente vía controladores.

Entre las razones por las cuales se emplea MVC se pueden citar:

- Permitir desarrollar independientemente el modelo y las capas de interfaz para el usuario.
- Permitir conectar fácilmente otras vistas a la capa actual del dominio, sin que esto la afecte.
- Permitir vistas simultáneas y múltiples del mismo objeto modelo.
- Permitir ejecutar la capa del modelo independiente de la capa de interfaz para el usuario.
- Permitir transportar fácilmente la capa de modelo a otro esquema de interfaz para el usuario. [17]

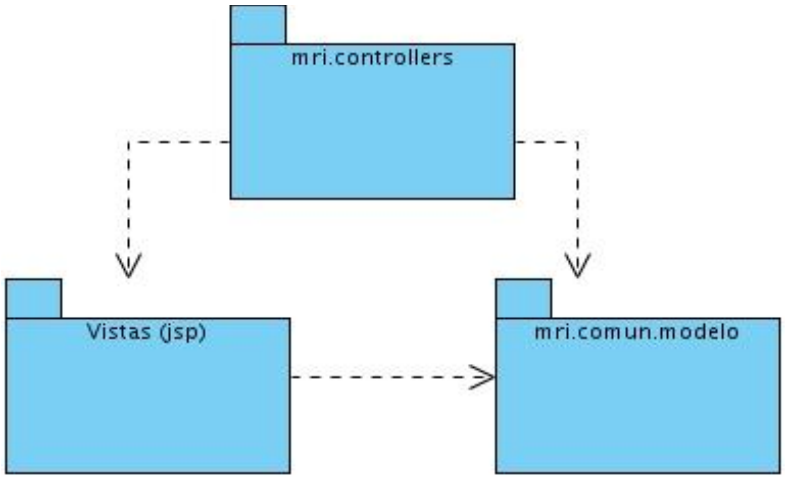


Figura 5

También se emplea el patrón Fachada o Facade como también se le conoce. Este es un patrón de software cuyo objetivo se basa en simplificar o hacer más amigable la complejidad asociada a una librería software.

Proporcionar una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

El uso de este patrón está recomendado para:

- Simplificar el uso y comprensión de una librería software.
- Encapsular un interfaz de librería poco amigable en un interfaz más coherente o mejor estructurado.
- Centralizar las dependencias externas hacia la librería en uno o pocos puntos de entrada.

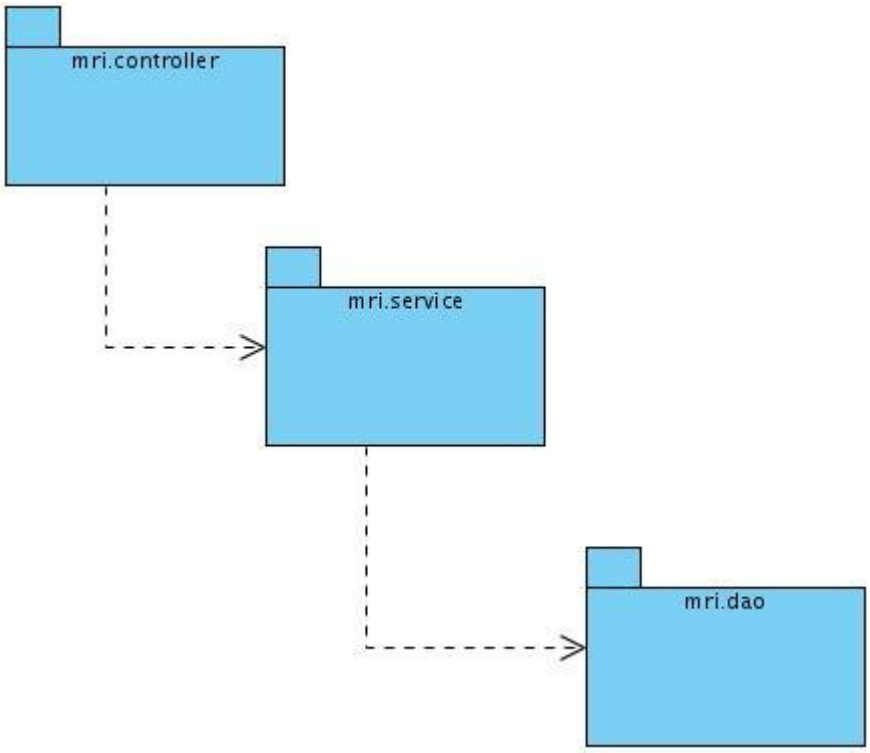


Figura 6

Otro de los patrones utilizados fue el nombrado Vistas Compuestas o Composite View**.** Este patrón hace que la representación de vistas sea más manejable ya que gestiona los diferentes elementos de una página por medio de una plantilla. Frecuentemente, las páginas webs contienen una combinación de contenido dinámico y elementos estáticos, tales como cabeceras, pies, logos, imágenes de fondo, etc. La parte dinámica es particular para cada página, pero los elementos estáticos suelen ser los mismos para todas las páginas. La plantilla de la vista compuesta captura estas características comunes. La integración debe ser dinámica, siendo el Composite View básicamente un diseño (layout, esquema) que componga dicha página.

Las aplicaciones web sofisticadas presentan contenido de numerosas fuentes de datos, usando múltiples subvistas, que conforman una única página a visualizar.

Usar vistas compuestas que están formadas por múltiples subvistas, Cada componente de la plantilla puede ser incluido dinámicamente en el total, y el esquema de la página puede ser gestionado de forma independiente al contenido.

Este escenario ocurre, por ejemplo, en portales web que incluyen numerosas e independientes subvistas.

Un requisito típico de las aplicaciones web es el que su visualización tenga una estructura similar a lo largo de todo el web. Una plantilla es un componente de presentación que compone vistas separadas en una única página con un diseño específico.

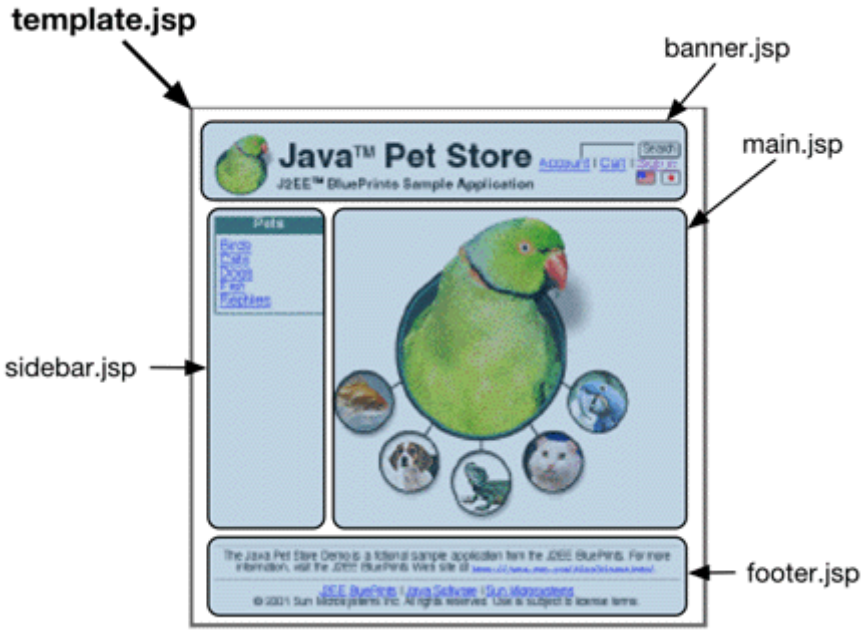


Figura 7

El diseño de clases de este patrón es el siguiente:

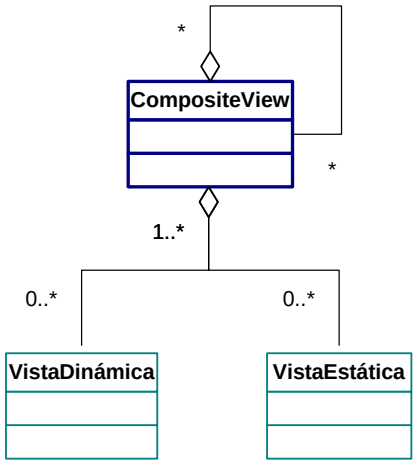


Figura 8

Con la aplicación de este patrón de diseño mejora la modularidad y la reutilización, mejora la flexibilidad, el mantenimiento, y la manejabilidad.

3.3 Patrones de asignación de responsabilidades (GRASP)

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Todo lo contrario: intentan codificar el conocimiento, las expresiones y los principios ya existentes: cuanto más trillados y generalizados, tanto mejor. En consecuencia, los patrones GRASP no introducen ideas novedosas; son una mera codificación de los principios básicos más usados. [18]

Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. [18]

GRASP es un acrónimo que significa "General Responsibility Assignment Software Patterns" (patrones generales de software para asignar responsabilidades) y la aplicación de sus principios son de vital importancia si se quiere diseñar eficazmente el software orientado a objetos. [18]

Experto:

Este patrón consiste fundamentalmente en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.

El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

En la figura se muestra la clase Anatómica evidenciándose la utilización de este patrón ya que esta es la clase que contiene los datos e información (atributos, constructor y métodos) mediante los cuales puede obtener información de la misma y modificarla.

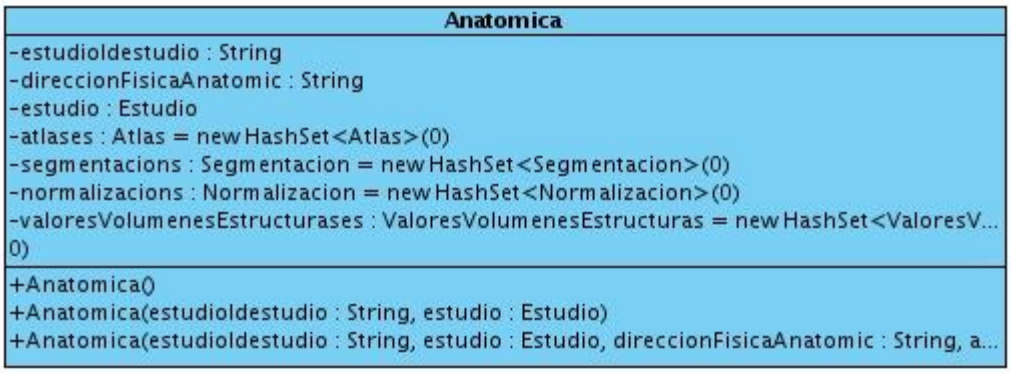


Figura 9

Creador:

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos.

En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella.

El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.

El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

Al escogerlo como creador, se da soporte al bajo acoplamiento.

Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A o
- B utiliza especialmente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.

Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

En esta figura se muestra el patrón creador en la clase SujetoDetail ya que esta crea objeto de las clases Sujeto y Estudio.

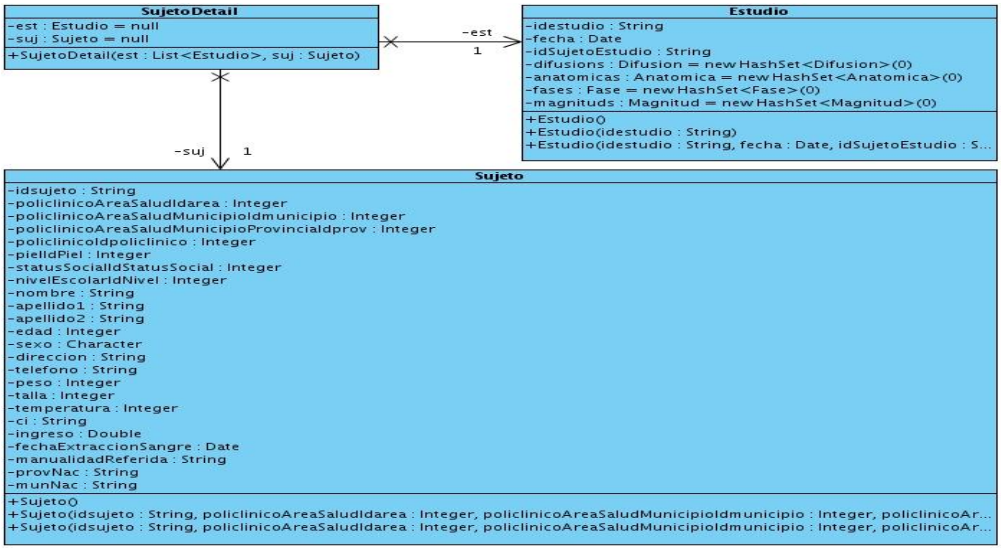


Figura 10

Controlador:

Consiste en asignar las responsabilidades del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El "sistema" global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados "manejador<NombreCasodeUso>" (controlador de casos de uso).

Beneficios:

Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la interfaz.

En la figura se evidencia el empleo de este patrón porque la clase AbstractHibernateDAO contiene todos los métodos relacionados con el acceso a la Base de Datos

Figura 11

Bajo Acoplamiento:

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

Acoplamiento bajo significa que una clase no depende de muchas clases.

Acoplamiento alto significa que una clase recurre a muchas otras clases. Esto presenta los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Difíciles de entender cuando están aisladas.
- Difíciles de reutilizar puesto que dependen de otras clases.

Este patrón consiste en asignar una responsabilidad para mantener bajo acoplamiento.

El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.

Alta Cohesión:

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. Esto presenta los siguientes problemas:

- Son difíciles de comprender
- Difíciles de reutilizar
- Difíciles de conservar

Las afectan constantemente los cambios.

Solución:

Asignar una responsabilidad de modo que la cohesión siga siendo alta.

3.4 Diagramas de Clases del Diseño.

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.). Los diagramas de clases son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño). El diagrama de clases de más alto nivel (main class diagram), será lógicamente un dibujo de los paquetes que componen el sistema. A su vez cada paquete tendrá un main class diagram que muestra las clases del paquete

Debido al uso del framework Spring se introduce para el diseño el patrón arquitectónico Modelo-Vista-Controlador (MVC) es por esta razón que los diagramas de clases del diseño correspondientes representan las interacciones entre las clases del sistema cumpliendo con el patrón arquitectónico definido. Este diseño está compuesto por tres paquetes (Modelo, Vista y Controlador) donde se encuentran las clases controladoras, las páginas clientes y las clases entidades. Las peticiones realizadas por los usuarios son atendidas por el controlador mri-servlet quien delega a otro componente de Spring el procesamiento de la solicitud; que es quien determina cuál será el controlador que recibirá la petición del usuario. El controlador definido se encarga de recuperar la información necesaria mediante comunicaciones que establece con las diferentes clases como las que prestan servicios, las clases de acceso a datos y las clases entidades contenidas en el modelo. Luego este notifica al controlador mri-servlet para que construya las páginas clientes con los datos requeridos.

Diagrama de Clases del diseño del Caso de Uso Manejar Estudios.

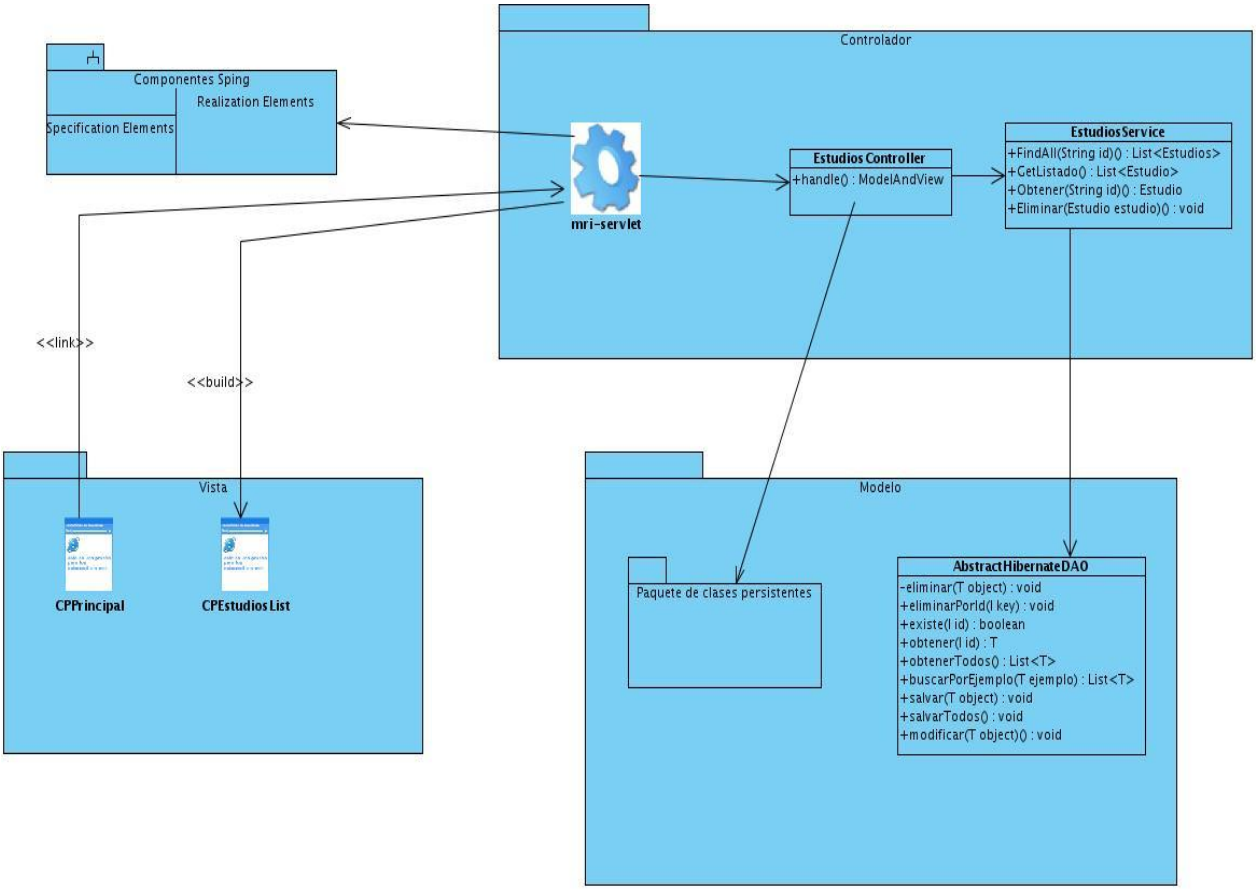


Figura 12

Diagrama de Clases del diseño del Caso de Uso Manejar Información de Sujeto.

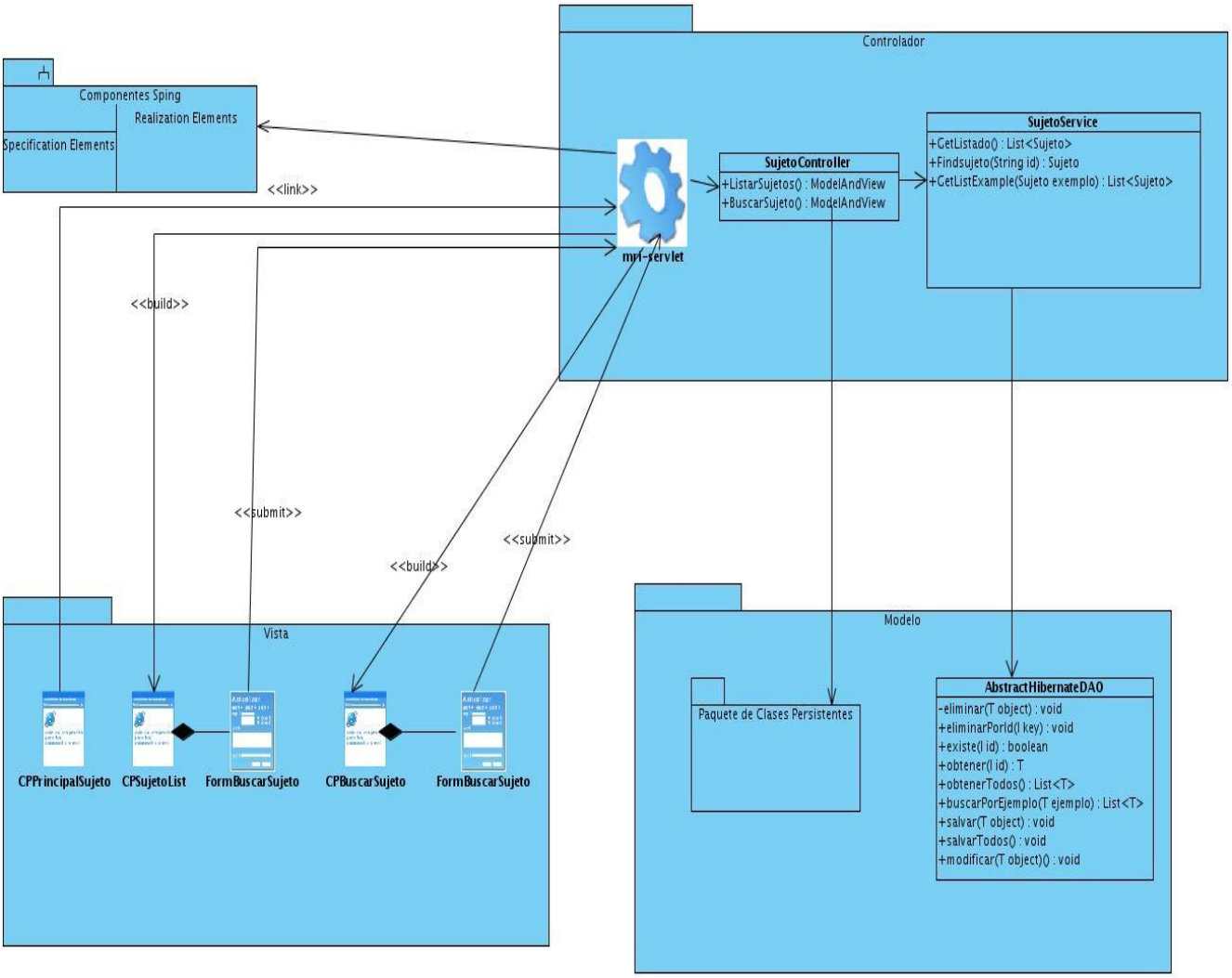


Figura 13

Diagrama de Clases del diseño del Caso de Uso Manipular Información de Imágenes.

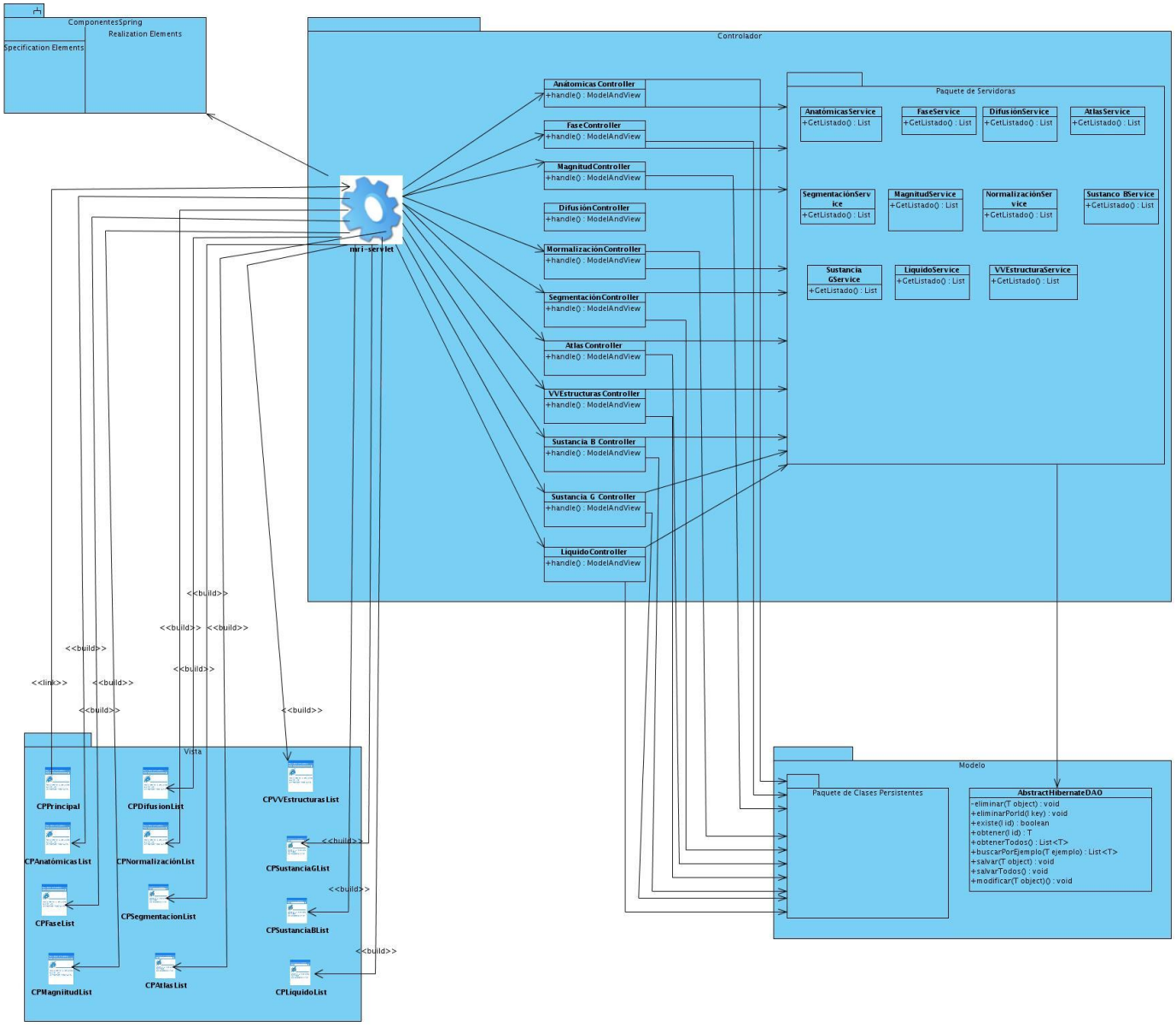


Figura 14

Diagrama de Clases Persistentes.

El en siguiente diagrama se muestran las clases del sistema y sus relaciones, son las clases que persistirán en la base de datos, por lo cual estas en correspondencia de 1…1 con las tablas que contiene la base datos, las cueles se muestran en el Modelo Entidad – Relación.

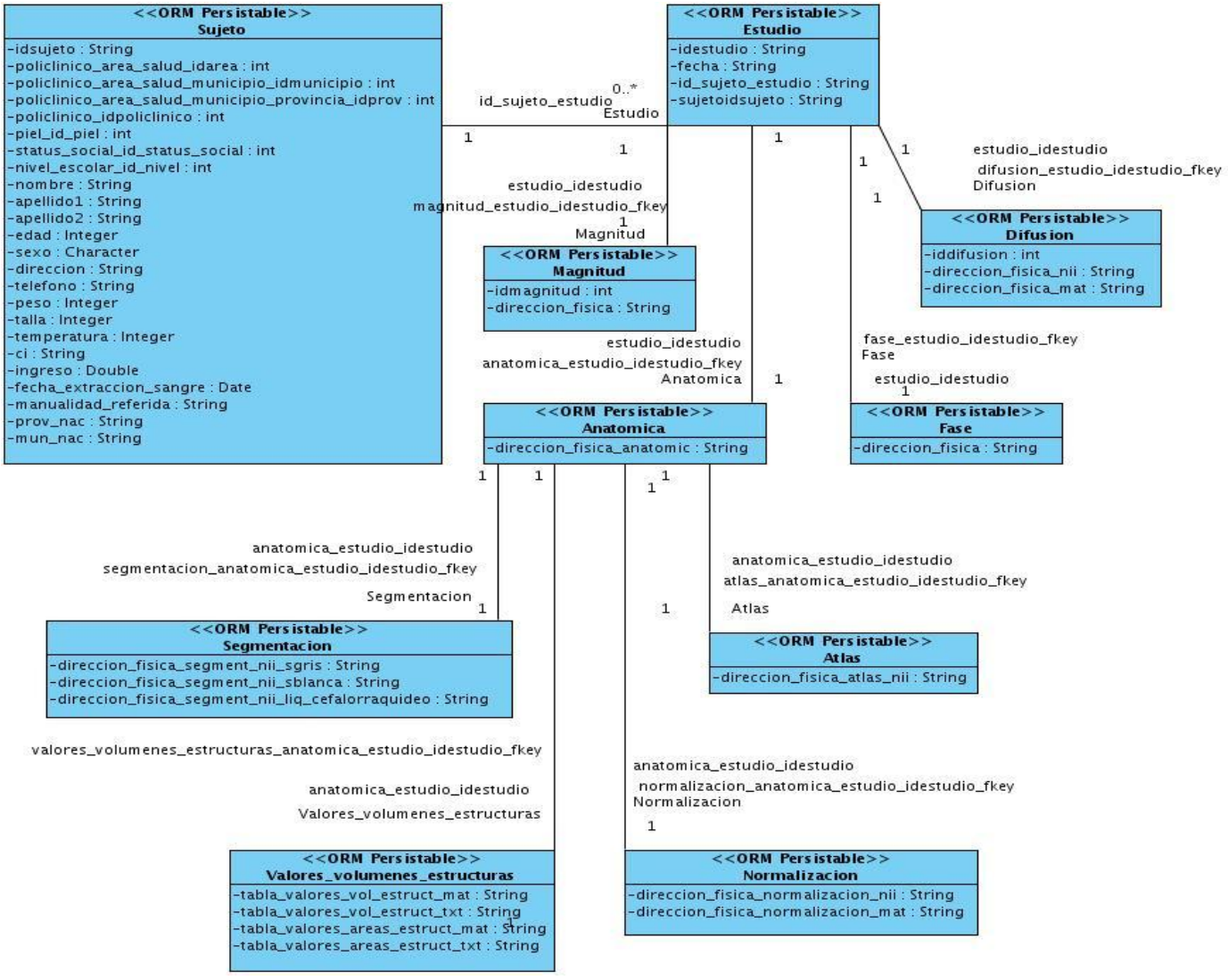


Figura 15

3.5 Diagrama de Secuencias.

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. En aplicaciones grandes además de los objetos se muestran también los componentes y casos de uso.

Los conceptos más importantes relacionados con los diagramas de secuencia son:

Línea de vida de un objeto (lifeline): La línea de vida de un objeto representa la vida del objeto durante la interacción. En un diagrama de secuencia un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal que denotan la ejecución de métodos (activación). El rectángulo de encabezado contiene el nombre del objeto y el de su clase.

Activación: Muestra el período de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto.

Mensaje: El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta. [19]

Diagrama de secuencia del Caso de Uso Manejar Estudios.

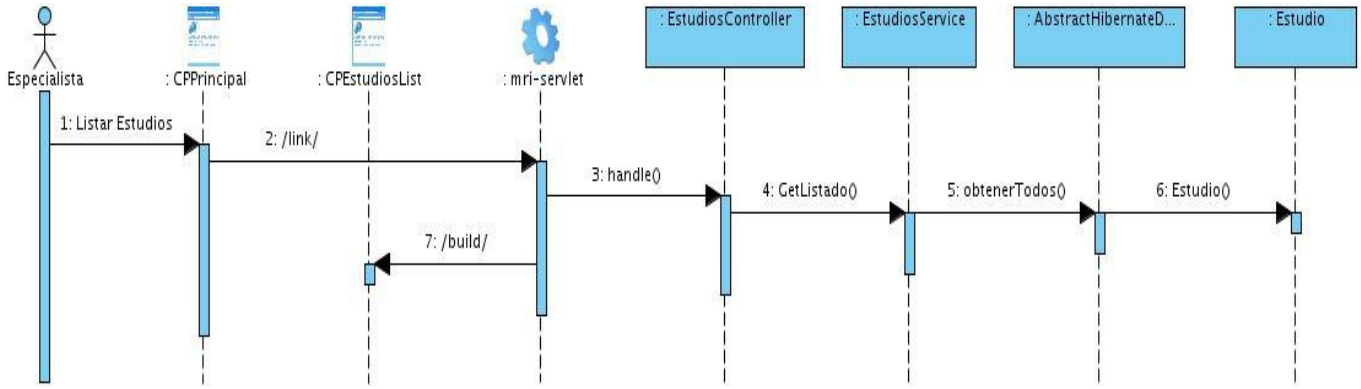


Figura 16

Diagrama de secuencia del Caso de Uso Manejar Información de Sujeto.

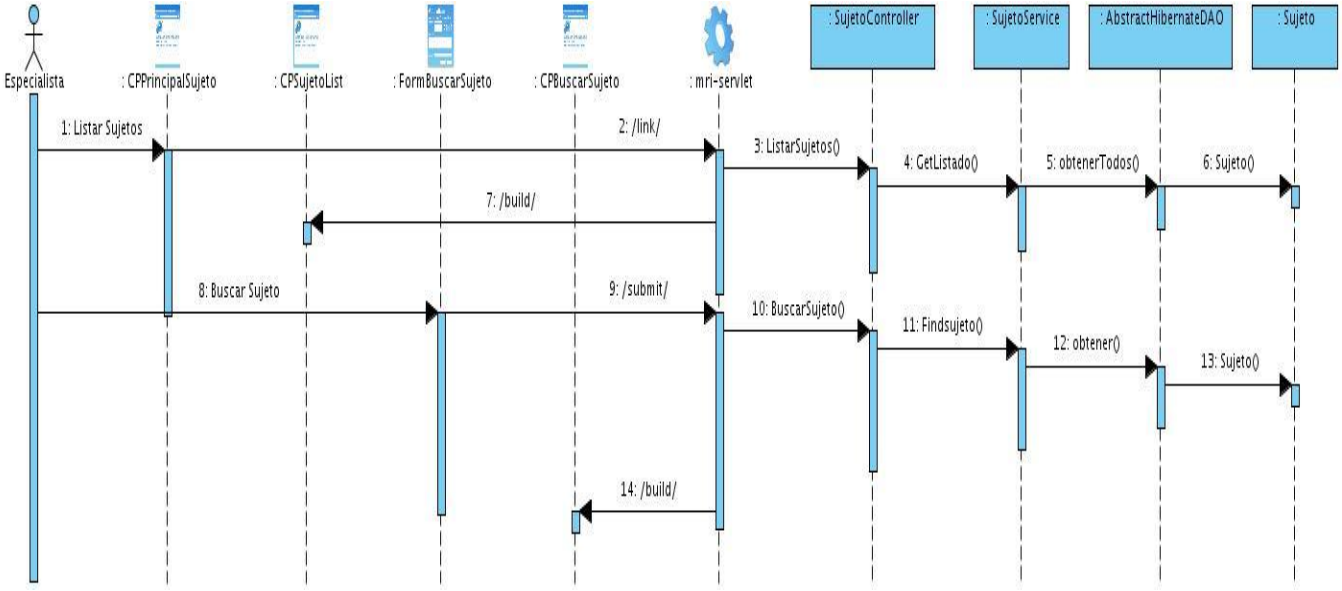


Figura 17

Diagrama de secuencia del Caso de Uso Manipular Información de Imágenes (Anatómica).

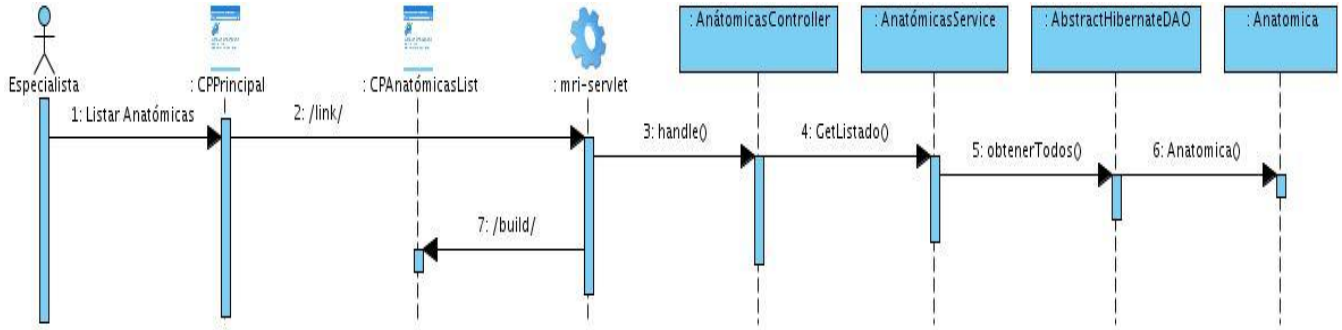


Figura 18

Diagrama de secuencia del Caso de Uso Manipular Información de Imágenes (Fase).

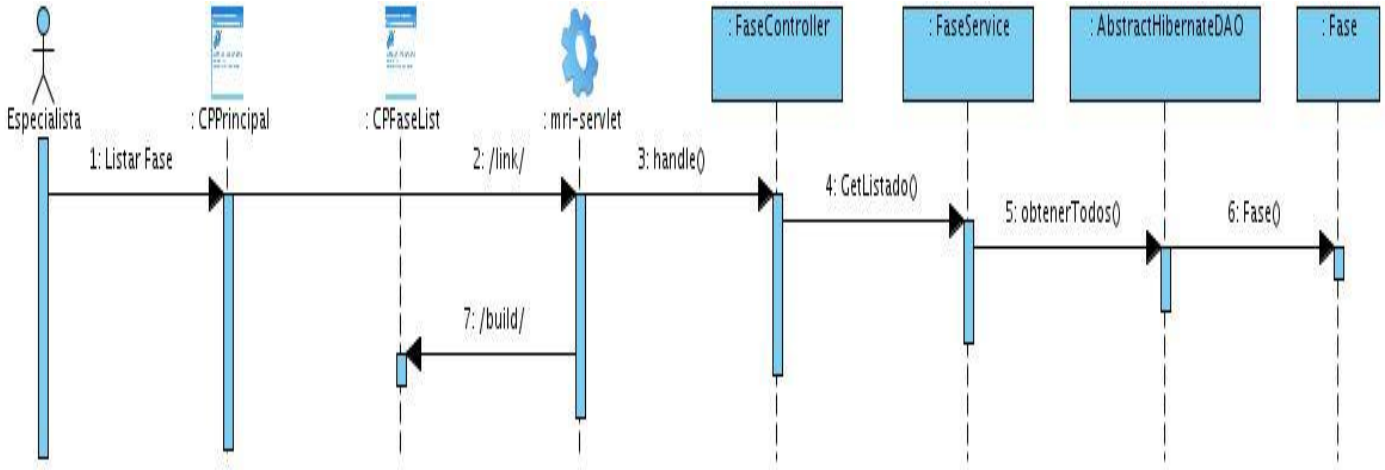


Figura 19

Diagrama de secuencia del Caso de Uso Manipular Información de Imágenes (Magnitud).

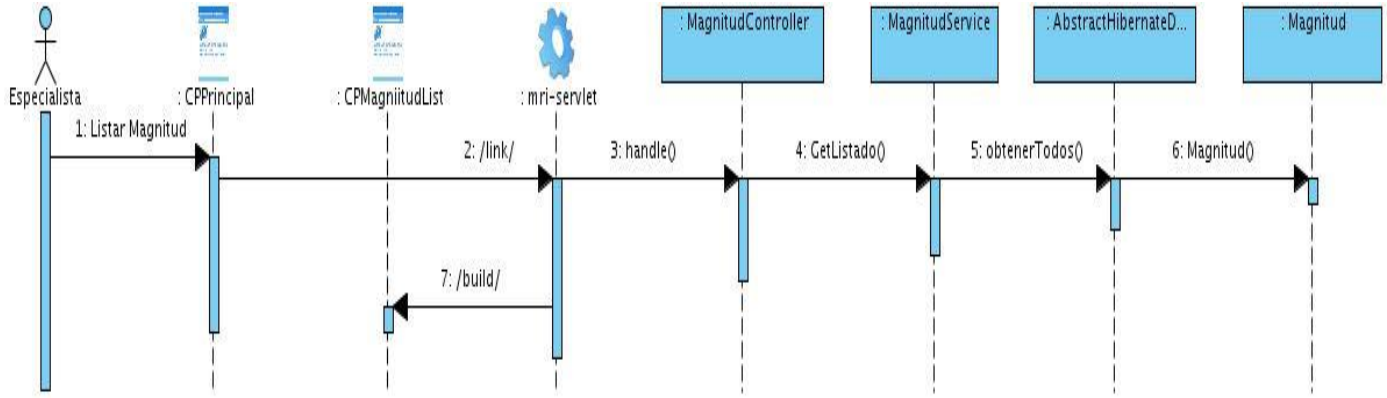


Figura 20

Diagrama de secuencia del Caso de Uso Manipular Información de Imágenes (Difusión).

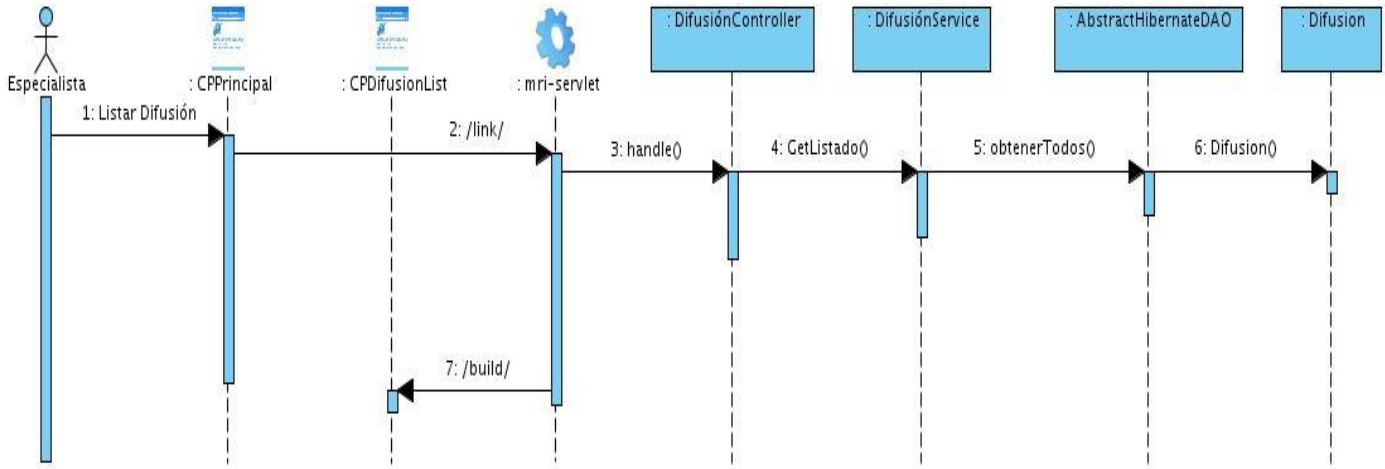


Figura 21

3.6 Modelo de Datos

Para almacenar la información generada en el proceso de las imágenes se diseñó una Base de Datos en el DBDesigner ForK. Esta herramienta permite el diseño y exportación a diferentes gestores de bases de datos tales como: PostgreSQL, MySQL, Oracle y SQL Server.

La Base de Datos, compuesta por cuarenta y nueve tablas, está normalizada hasta la tercera forma normal. La tabla principal, denominada Sujeto, contiene los datos personales (nombre, apellidos, edad, sexo, policlínico, área de salud, municipio y provincia de residencia) del sujeto que pertenece al estudio.

3.6.1 Tablas de la Base de Datos.

Nombre: Sujeto		
Descripción: Contiene todos los datos referentes al sujeto al que se le realiza el estudio		
Atributo	**Tipo**	**Descripción**

idsujeto	varchar	Contiene el identificador del sujeto
policlínico_area_salud_ldarea	int	Contiene el id del área a la que corresponde el policlínico al que pertenece el sujeto
policlinico_area_salud_municipio_idmunicipio	int	Contiene el id del municipio al que pertenece el policlínico al que pertenece el sujeto
policlinico_area_salud_municipio_provincia_idprov	int	Contiene el id de la provincia a la que pertenece el policlínico al que pertenece el sujeto
policlinico_idpoliclinico	int	Contiene el id del policlínico al que pertenece el sujeto
piel_id_piel	int	Contiene el identificador del tipo de piel del sujeto
status_social_id_status_social	int	Contiene el status social del sujeto
nivel_escolar_id_nivel	int	Contiene el nivel escolar del sujeto
nombre	varchar	Contiene el nombre del sujeto
apellido1	varchar	Contiene el 1er apellido del sujeto
apellido2	varchar	Contiene el 2do apellido del sujeto

edad	int	Contiene la edad del sujeto
sexo	char	Contiene el sexo del sujeto
dirección	varchar	Contiene la dirección del sujeto
teléfono	varchar	Contiene el teléfono del sujeto
peso	int	Contiene el peso del sujeto
talla	int	Contiene la talla del sujeto
temperatura	int	Contiene la temperatura del sujeto
ci	float	Contiene el número de carnet de identidad del sujeto
ingreso	float	Contiene el salario del sujeto
fecha_extraccion_sangre	date	Contiene la fecha de la última extracción de sangre que se hizo el sujeto
manualidad_referida	varchar	Contiene la manualidad del sujeto
prov_nac	varchar	Contiene la provincia en la que nació el sujeto
mun_nac	varchar	Contiene el municipio en el que nació el sujeto

Nombre: Estudio		
Descripción: Contiene todos los datos referentes al estudio del sujeto.		
Atributo	**Tipo**	**Descripción**
idestudio	varchar	Contiene el identificador del estudio
fecha	date	Contiene la fecha en la que se realizó el estudio
Ide_sujeto_estudio	varchar	Contiene el identificador del sujeto al que se le está realizando este estudio

Nombre: Anatómica		
Descripción: Contiene los datos referentes a la imagen de tipo anatómica		
Atributo	**Tipo**	**Descripción**
dirección_física_anatómica	varchar	Contiene la dirección física de donde se encuentra la imagen

Nombre: Fase
Descripción: Contiene los datos referentes a la imagen de tipo Fase

Atributo	**Tipo**	**Descripción**
dirección_física	varchar	Contiene la dirección física de donde se encuentra la imagen

Nombre: Difusión		
Descripción: Contiene los datos referentes a la imagen de tipo Difusión		
Atributo	**Tipo**	**Descripción**
dirección_física_nii dirección_física_mat	varchar	Contiene la dirección física de donde se encuentra la imagen

Nombre: Magnitud		
Descripción: Contiene los datos referentes a la imagen de tipo Magnitud		
Atributo	**Tipo**	**Descripción**
dirección_física	varchar	Contiene la dirección física de donde se encuentra la imagen

Nombre: Segmentación
Descripción: Contiene los datos referentes a la imagen de tipo Segmentación

Atributo	**Tipo**	**Descripción**
dirección_física_segment_nii_sgris dirección_física_segment_nii_sblanca dirección_física_segment_nii_liq_cefalorraquideo	varchar	Contiene la dirección física de donde se encuentra la imagen
Nombre: Valores_Volumenes_Estructuras		
Descripción: Contiene los datos referentes a la imagen de tipo Valores_Volumenes_Estructuras		
Atributo	**Tipo**	**Descripción**
tabla_valores_vol_estruct_mat tabla_valores_vol_estruct_txt tabla_valores_areas_estruct_mat tabla_valores_areas_estruct_txt	varchar	Contiene la dirección física de donde se encuentra la imagen

Nombre: Normalización
Descripción: Contiene los datos referentes a la imagen de tipo Normalización

Atributo	**Tipo**	**Descripción**
dirección_física_normalización_nii dirección_física_normalización_mat	varchar	Contiene la dirección física de donde se encuentra la imagen

Nombre: Atlas		
Descripción: Contiene los datos referentes a la imagen de tipo Atlas		
Atributo	**Tipo**	**Descripción**
dirección_física_atlas_nii	varchar	Contiene la dirección física de donde se encuentra la imagen

3.6.2 Modelo Entidad-Relación

El modelo entidad no es más que un diagrama donde se visualizan los objetos que pertenecen a la base de datos como entidades los cuales contienen atributos y se vinculan mediante relaciones.

Entidades: Representa una “cosa” u "objeto" del mundo real con existencia independiente, es decir, se diferencia unívocamente de cualquier otro objeto o cosa, incluso siendo del mismo tipo.

Atributo: Los atributos son las propiedades que describen a cada entidad en un conjunto de entidades.

Relación: Describe cierta dependencia entre entidades o permite la asociación de las mismas.

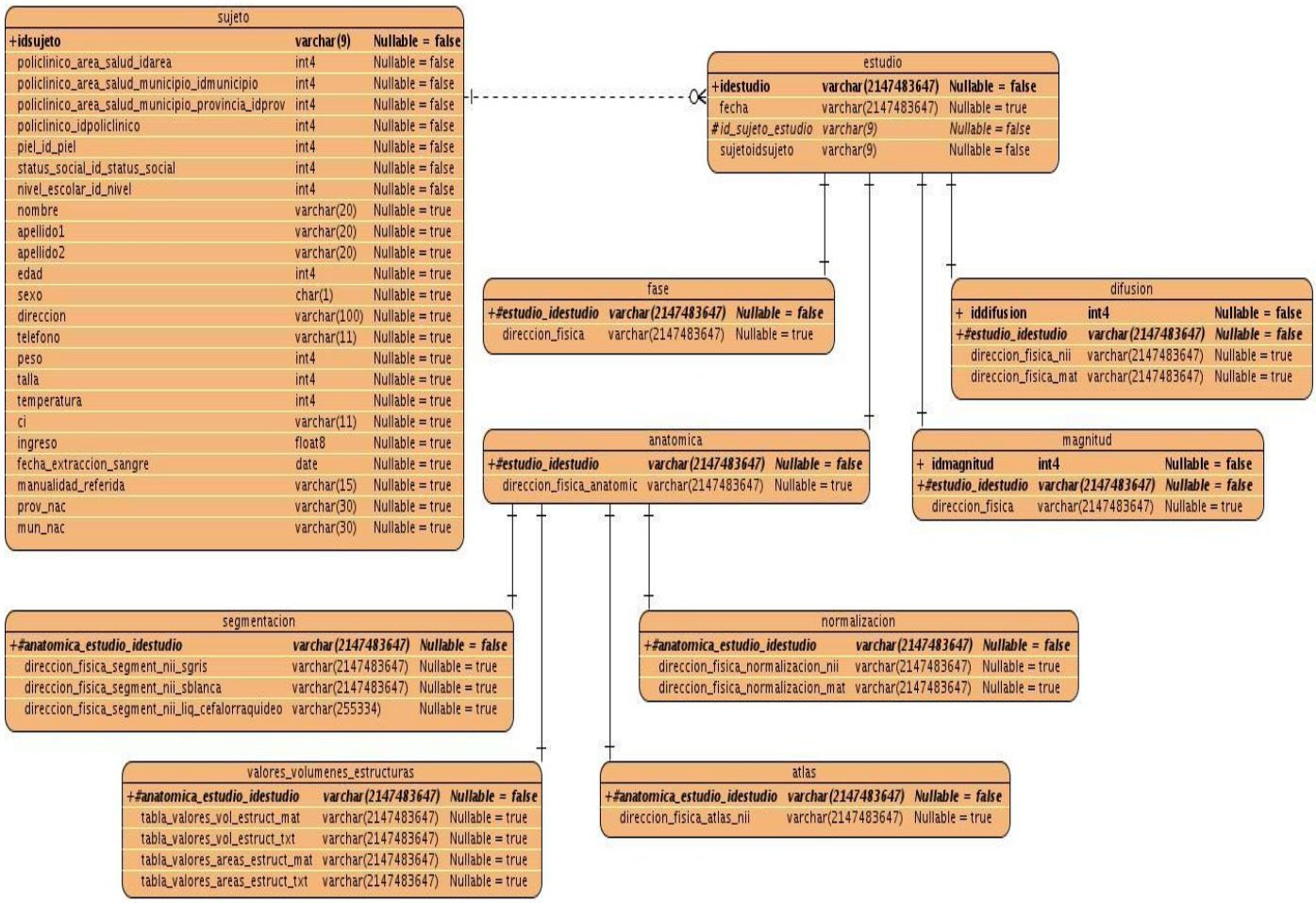


Figura 22

3.7 Diagrama de Despliegue

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

La aplicación estará distribuida en los servidores de CNEURO, de la siguiente manera: Un servidor web para publicar el sistema, otro servidor para la base de datos y una para almacenar las imágenes, como gestor de base datos se tendrá PostgreSQL y Apache Tomcat como servidor de la aplicación web, a la aplicación se tendrá acceso desde cualquier máquina cliente del centro.

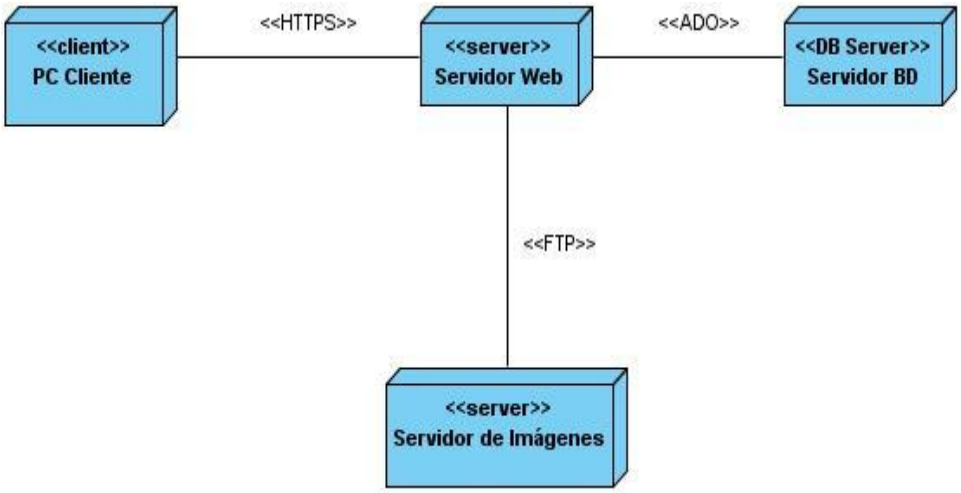


Figura 23

3.8 Mapa de Navegación

El siguiente diagrama muestra como está estructurado el contenido que gestiona la aplicación web, muestra además como el usuario puede navegar en ella, visualizando gráficamente la falibilidad que le brinda la aplicación web al usuario de acceder a todo el contenido con un número menor a 3 clics desde cualquier página en la que se encuentre.

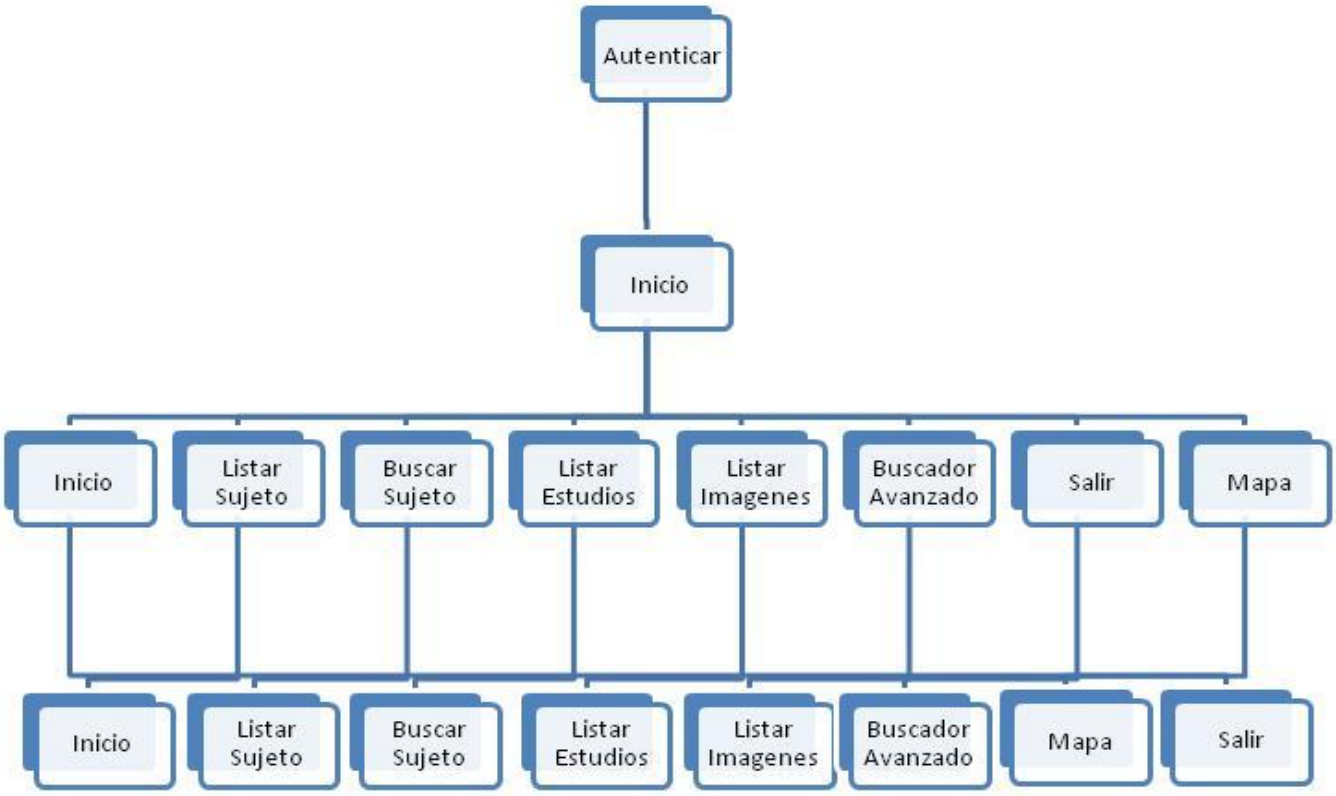


Figura 24

3.9 Conclusiones

Con este capítulo se obtuvo como resultado la representación de los diagramas de clases del diseño así como su descripción, los diagramas de secuencias de los diferentes casos de uso para luego pasar a su implementación. Además se ejemplificó el uso de los diferentes patrones de diseño y arquitectura. También se realizó una descripción del modelo datos.

Capítulo 4: Implementación y Prueba

4.1 Introducción

En este capítulo se realiza una descripción de cómo va a ser implementado el sistema a través del diagrama de componentes y una breve explicación del mismo. Además se presentan algunas de las pantallas principales que conforman la aplicación y los resultados de algunas pruebas que validen que el sistema funciona correctamente.

4.2 Diagrama de Componentes

Un diagrama de componentes muestra un conjunto de componentes y sus relaciones. Los diagramas de componentes se utilizan para describir la vista de implementación estática de un sistema. Los diagramas de componentes se relacionan con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones. [19]

Para un mejor entendimiento de la estructura de la implementación de la aplicación los componentes están divididos en las 3 capas de la arquitectura utilizada, modelo, vista y controlador conformando cada una de estas capas un paquete de componentes. En las figuras siguientes de muestran los paquetes con los componentes correspondientes y sus relaciones.

Modelo

En la siguiente imagen se representa la interrelación entre los componentes correspondiente a la capa Modelo de la arquitectura MVC utilizada en el desarrollo de la aplicación web, el diseño del diagrama está dado según la filosofía del framework Hibernate, cada una de las clases .hbm.xml del subpaquete mapeo tienen una relación de 1 a 1 con las clases .java del subpaquete modelo y su vez ambos subpaquetes están relacionados con el subsistema Hibernate y la base de datos respectivamente, conformando así el paquete de componentes Modelo.

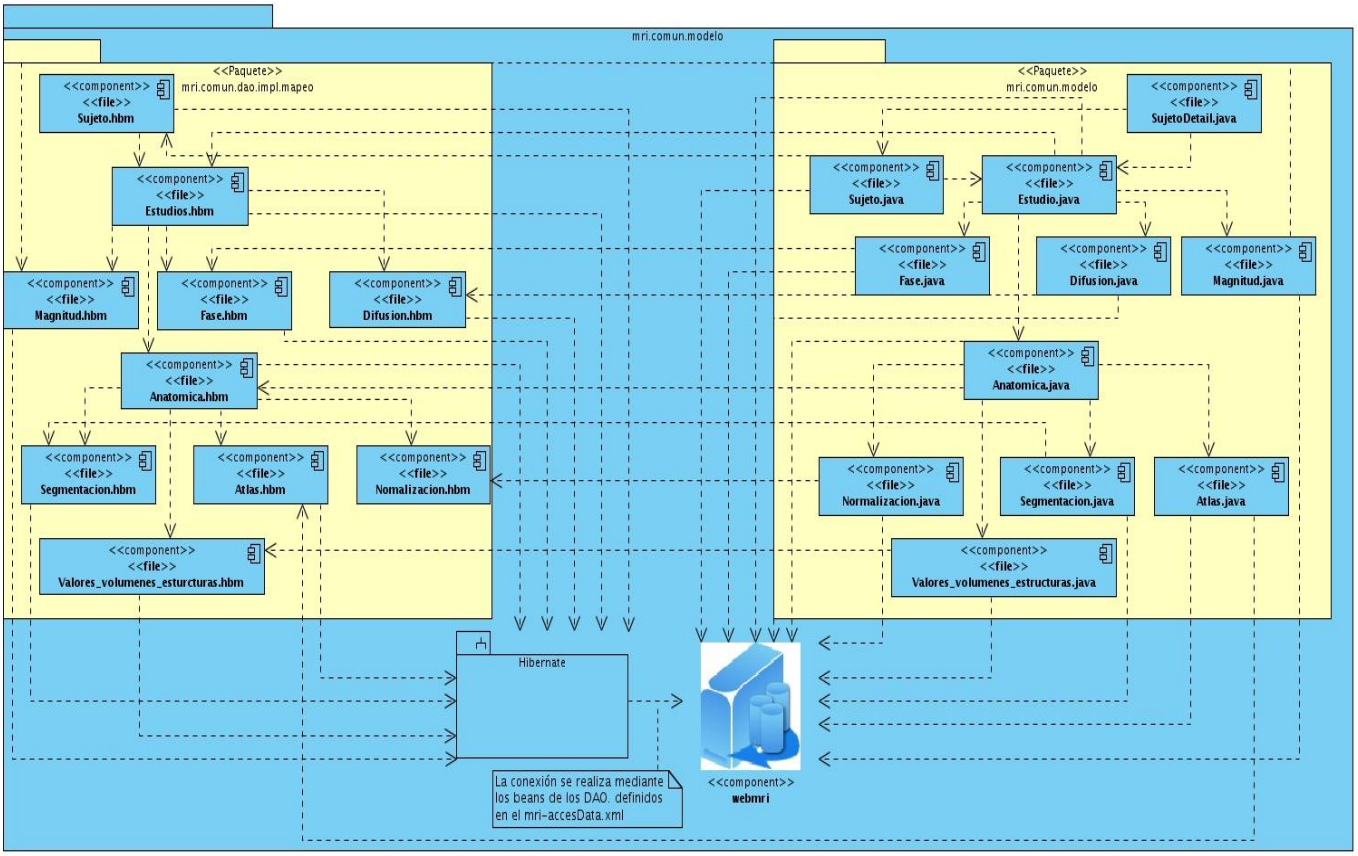


Figura 25

Vista

El siguiente diagrama de componentes representa la interacción de los mismos, en la capa vista de la arquitectura mencionada, las relaciones entre los componentes están dadas según la filosofía del framework Spring, utilizando la facilidad de los tiles del mismo, los cuales permiten utilizar un solo diseño para todas las vistas .jsp, en este caso el diseño llamado template.jsp, compuesto por las vistas menú.jsp, include.jsp e includeHeader,jsp, la estructura del diseño es debido al uso del patrón Composite View utilizado en dicha aplicación para darle un fácil mantenimiento y cambios en la interfaz siempre y cuando sea necesario o el cliente lo solicite.

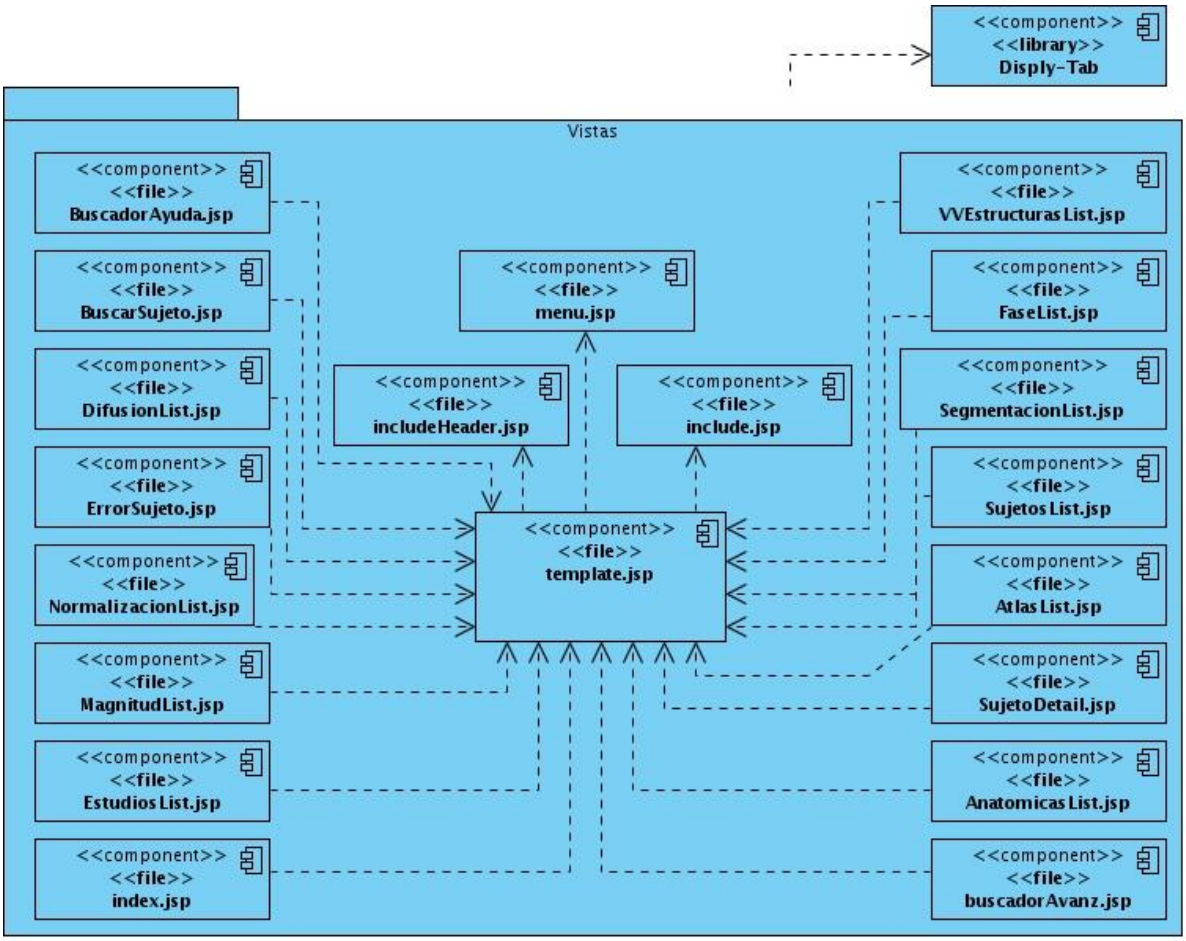


Figura 26

Controlador

La última capa de la arquitectura MVC, la capa Controladora, está representada en el siguiente diagrama compuestos por los subpaquetes controllers y services y la relación entre los componentes de los mismos, cada componente del subpaquete controllers tiene una relación de 1 a muchos con los componentes del subpaquete services. Y a su vez cada componente del subpaquete services implementa su interfaz, en el caso de los componentes ImprimirDireciones.java y BuscadorAyudaControllers.java del subpaquete controllers utilizan el subpaquete de componentes services completo.

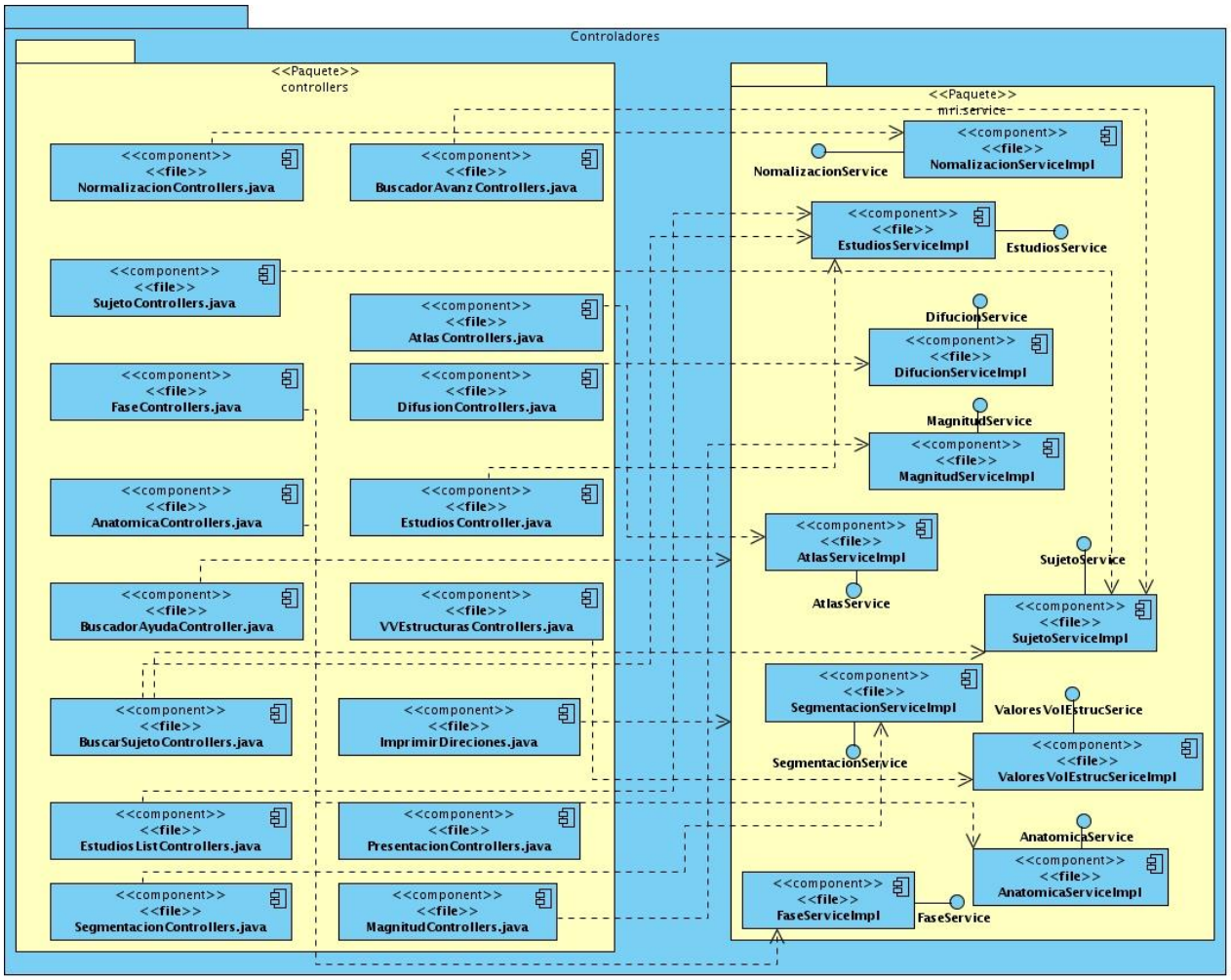


Figura 27

Diagrama de Componentes del CU Manipular Información del Sujeto

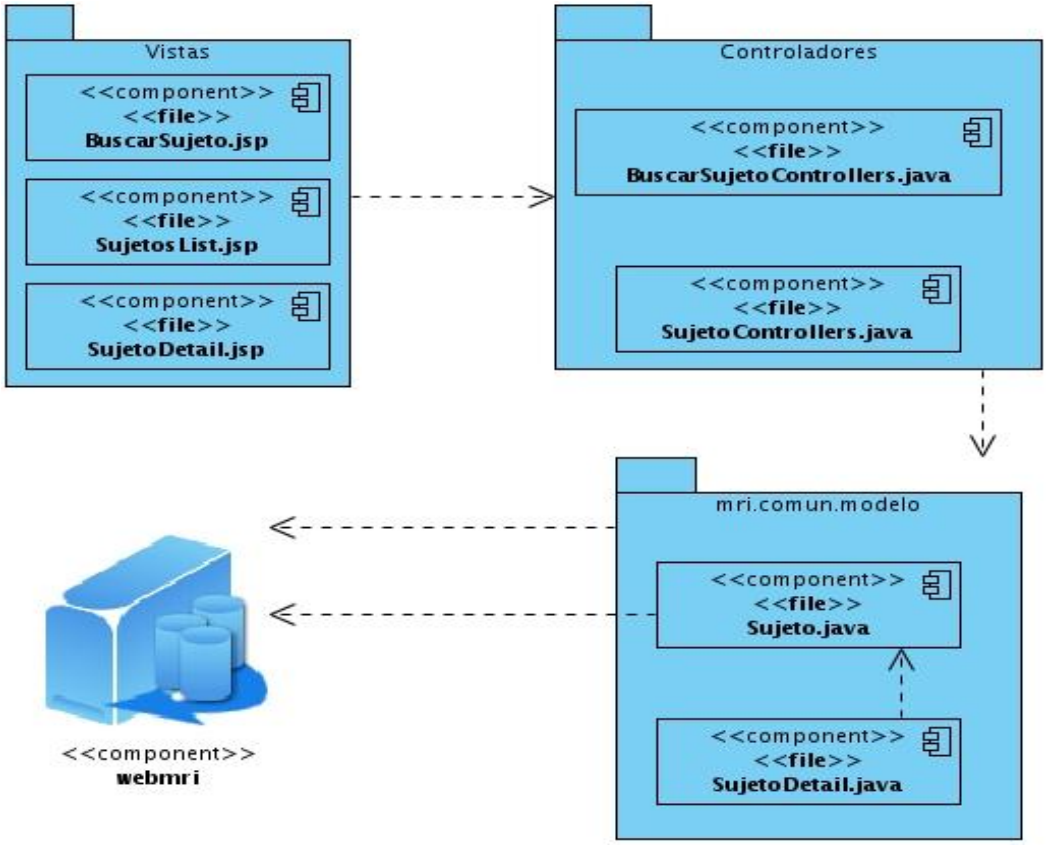


Figura 28

Diagrama de Componentes del CU Manipular Información de Imágenes

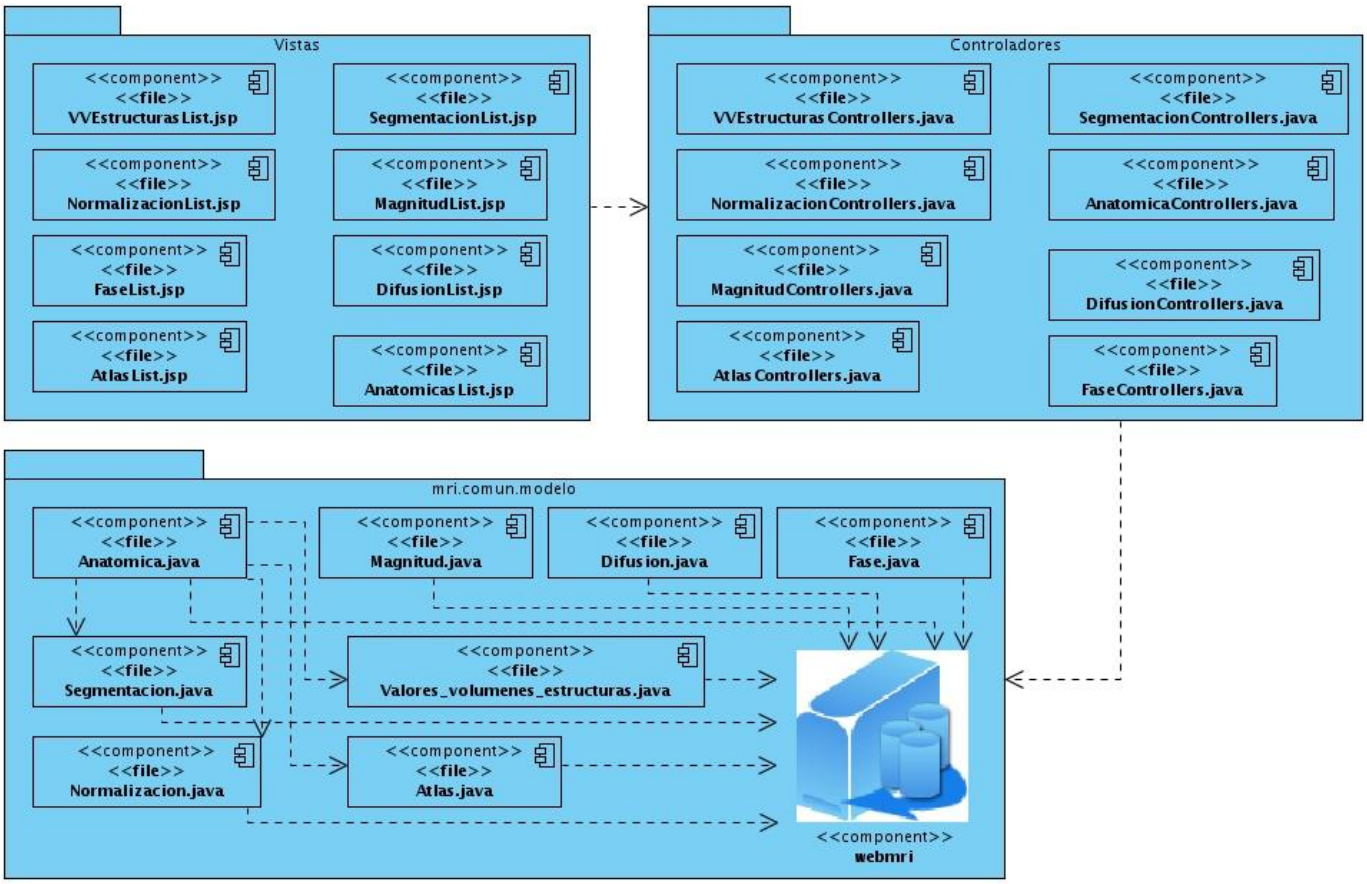


Figura 29

Diagrama de Componentes del CU Manejar Estudios

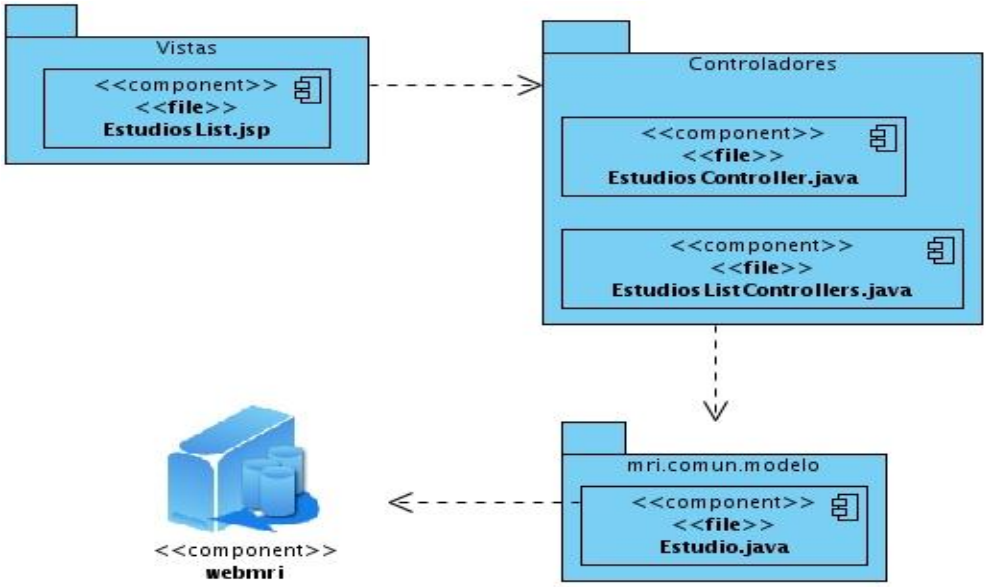


Figura 30

Diagrama de Componentes del CU Realizar Búsqueda Avanzada de Imágenes

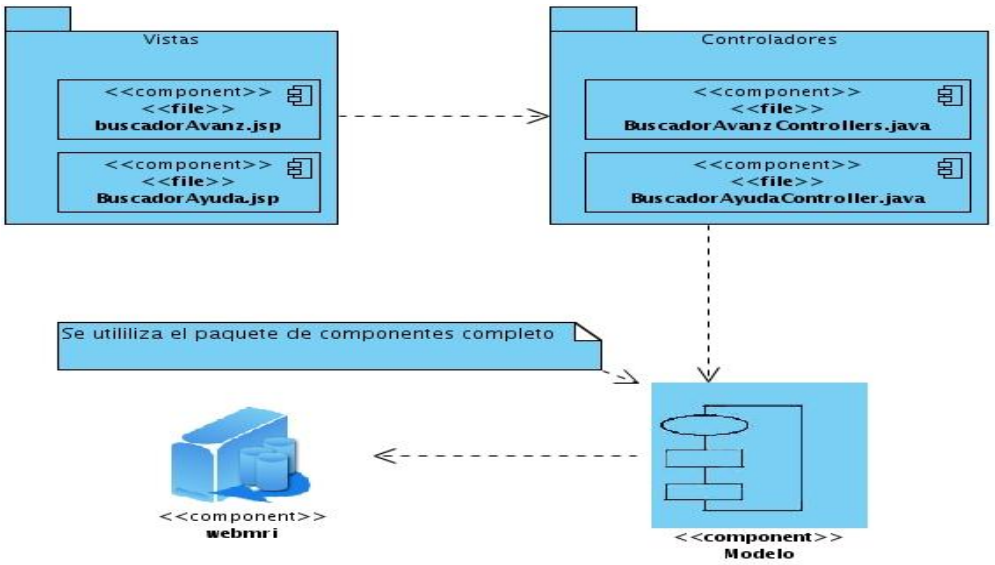


Figura 31

4.3 Código Fuente de las principales clases.

El código fuente de una aplicación informática no es más que un conjunto de instrucciones que la máquina debe interpretar para así poder ejecutar las funcionalidades requeridas por el software. El código fuente está escrito por un programador en un lenguaje de programación determinado, que debe ser traducido al lenguaje máquina o código objeto por los programas conocido como compiladores, ensambladores o interpretes.

En esta sección se describen algunos ejemplos de código de la aplicación web desarrollada, aunque no es posible abarcar todo el código de la misma, se mostrarán ejemplos de las clases más importantes del sistema y de algunos archivos .xml de configuración que constituyen un punto esencial para el funcionamiento del software.

En el siguiente ejemplo se abarcará lo mayor posible las relaciones entre todos los códigos relacionados con las clase Estudios en diferentes partes del sistema, haciendo un recorrido desde la tabla Estudio de la base de datos hasta la vista (EstudiosList.jsp), con el objetivo de evidenciar el uso de los framework Spring e Hibernate.

Ejemplo:

Panel SQL

```
-- Table: estudio

-- DROP TABLE estudio;

CREATE TABLE estudio
(
  idestudio character varying NOT NULL,
  fecha character varying,
  id_sujeto_estudio character varying,
  CONSTRAINT estudio_pkey PRIMARY KEY (idestudio)
)
WITH (OIDS=FALSE);
ALTER TABLE estudio OWNER TO postgres;
```

Figura 32

En la figura se muestra el código de la tabla Estudio en la base de datos, solo contiene tres campos o columnas.

Con el uso de la herramienta Hibernate Tool se genera la capa de acceso a todos y luego se pueden utilizar todas las funcionalidades del framework Hibernate. Dicha herramienta genera dos archivos, el mapeo de la tabla con la clases correspondiente al lenguaje seleccionado, en este caso el java.

En la siguiente figura se muestra el mapeo de la tabla Estudio a la clase Estudios.java, haciendo corresponder cada columna de la tabla con cada atributo de la clase.

```
1<?xml version="1.0"?>
2<!DOCTYPE hibernate-mapping
3"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
4<!-- Generated 04-jun-2008 23:06:38 by Hibernate Tools 3.2.0.CR1 -->
5<hibernate-mapping>
6    <class name="mri.comun.modelo.Estudio" table="estudio" schema="public">
7        <id name="idestudio" type="string" >
8            <column name="idestudio" />
9            <generator class="assigned" />
.0       </id>
.1       <property name="fecha" type="date">
.2           <column name="fecha" length="13" />
.3       </property>
.4       <property name="idSujetoEstudio" type="string">
.5           <column name="id_sujeto_estudio" />
.6       </property>
.7
```

tesis/src/mri/comun/dao/impl/mapeo/Estudio.hbm.xml

Figura 33

Estudio.hbm.xml

```
public class Estudio implements java.io.Serializable {

    private String idestudio;
    private Date fecha;
    private String idSujetoEstudio;
    private Set<Difusion> difusions = new HashSet<Difusion>(0);
    private Set<Anatomica> anatomicas = new HashSet<Anatomica>(0);
    private Set<Fase> fases = new HashSet<Fase>(0);
    private Set<Magnitud> magnituds = new HashSet<Magnitud>(0);

    public Estudio() {
    public Estudio(String idestudio) {
    public Estudio(String idestudio, Date fecha, String idSujetoEstudio,
    public String getIdestudio() {
    public void setIdestudio(String idestudio) {
    public Date getFecha() {
    public void setFecha(Date fecha) {
    public String getIdSujetoEstudio() {
    public void setIdSujetoEstudio(String idSujetoEstudio) {
    public Set<Difusion> getDifusions() {
    public void setDifusions(Set<Difusion> difusions)  {
    public Set<Anatomica> getAnatomicas() {
    public void setAnatomicas(Set<Anatomica> anatomicas) {
    public Set<Fase> getFases() {
    public void setFases(Set<Fase> fases) {
    public Set<Magnitud> getMagnituds() {
    public void setMagnituds(Set<Magnitud> magnituds) {

}
```

Estudio.java

Figura 34

Una vez mapeada la tabla Estudio de la base datos, se crean los DAO (objetos de acceso a datos) correspondientes a cada tabla, en este caso a Estudio.

En el siguiente ejemplo se evidencia que la clase EstudiosDAOImpl extiende de AbstractHibernateDAO, e implementa su interfaz, de esta forma la clase EstudiosDAOImpl podrá utilizar todos los métodos de acceso a los datos en la tabla Estudio, definidos en el framework Hibernate, aclarar que la clase AbstractHibernateDAO es una interfaz programada de modo personalizado, la clase interna que contiene los métodos definidos por el framework es HibernateDaoSupport.

```java
package mri.comun.dao.impl;

import mri.comun.dao.EstudioDAO;

public class EstudiosDAOImpl extends AbstractHibernateDAO<Estudio, String>
            implements EstudioDAO{

    protected EstudiosDAOImpl() {
        super(Estudio.class);
        // TODO Auto-generated constructor stub
    }

}
```

Figura 35

EstudiosDAOImpl.java

```java
package mri.comun.dao.impl;

import java.io.Serializable;

public class AbstractHibernateDAO<T extends Object, I extends Serializable>
        extends HibernateDaoSupport implements AbstractDAO<T, I> {

    private Class type;

    protected AbstractHibernateDAO(Class type) {

    public void eliminar(T t) {

    public boolean existe(I id) {

    public void modificar(T t) {

    public T obtener(I id) {

    public List<T> obtenerTodos() {

    public void salvar(T t) {

    public void salvarTodos(Collection<T> entidades){

    public void eliminarPorId(I key) {

    public List<T> buscarPorEjemplo(T ejemplo) {
}
```

Figura 36

AbstractHibernateDAO.java

Los DAO de una clase deben ser configurados en el archivo xml para cuando el sistema arranque sean cargados puede acceder a los datos, en este caso los DAO están configurados en el mri-accesData.xml.

La siguiente imagen es el fragmento de configuración del DAO correspondiente a la clase EstudioDAOImpl. La propiedad "sessionFactory" es la que abre y cierra las conexiones con la base de datos, para un mejor entendimiento ver **anexo 2**.

```
<bean id="estudioDAO" class="mri.comun.dao.impl.EstudiosDAOImpl">
    <property name="sessionFactory">
        <ref bean="sessionFactory" />
    </property>
</bean>
```

Figura 37

Los DAO definidos son utilizados por los servicios, que son definidos en el archivo mri-services.xml, la siguiente imagen muestra la clase EstudioServiceImpl.java encargada de hacer las peticiones al los DAO y entregarlas a los controladores.

```
package mri.serviceImpl;

import java.util.List;

public class EstudiosServiceImpl implements EstudiosService{

    EstudioDAO estudioDAO;

    public List<Estudio> GetListado() {
    public EstudioDAO getEstudioDAO() {
    public void setEstudioDAO(EstudioDAO estudioDAO) {
    public List<Estudio> FindAll(String id) {
    public Estudio Obtener(String id) {
    public void Eliminar(Estudio estudio) {

}
```

Figura 38

EstudioServiceImpl.java

Configuración del servicio EstudiosService en el archivo mri-services.xml.

Mri-services.xml

```
<bean id="estudiosService" class="mri.serviceImpl.EstudiosServiceImpl">
    <property name="estudioDAO">
        <ref bean="estudioDAO" />
    </property>
</bean>
```

Figura 39

EstudioControllers.java

```
package controllers;

import java.util.List;

public class EstudiosController extends AbstractController{

    private EstudiosService estudiosService;

    protected ModelAndView handleRequestInternal(HttpServletRequest request

    public EstudiosService getEstudiosService() {

    public void setEstudiosService(EstudiosService estudiosService) {

}
```

Figura 40

Hasta el momento se tiene una gran parte del camino que se recorre desde la capa de la vista hasta la capa modelo, solo falta por definir parte de la capa de los controladores pues ya se tienen los servicios declarados que pertenecen a esta capa. Los controladores son los encargados de construir las vistas relacionadas con las peticiones de un usuario que interactúa con la vista, en la siguiente imagen se muestra el código del controlador relacionado con el ejemplo. Y como estos utilizan a los servicios.

Los controladores también son configurados en el mri-servlet.xml. Como los controladores son los encargados de construir las vistas, en este caso nos construye la vista EstudiosList.jsp, a la cual corresponde el siguiente ejemplo de código.

EstudiosList.jsp

```
<display:table name="estudList"  pagesize="5" requestURI="EstudiosList.htm"
               class="${tableclass}" id="estud_list"  >

 <display:column property="idSujetoEstudio" title="ID del sujeto" sortable="true" > </display:column>
 <display:column property="idestudio" title="ID del estudio"></display:column>
 <display:column property="fecha"   title="Fecha" ></display:column>

 <display:column  title="Visualizar" >

 <table width="90%" align="left" class="its" cellpadding="1" cellspacing="1"  title="Imágenes del Estudio">

  <tr align="left" >

    <th><a name='<c:out value="${estud_list.idestudio}"/>' href='#<c:out value="${estud_list.idestudio}"/>' c
    <!--
    <th><a href="deleteEstudio.htm?id_est=<c:out value="${st.idestudio}"/>"><img id="img_visualizar"  src="ir
 -->
 </tr>
 <tr align="left" bgcolor="" >
   <td colspan="5" >
   <div  class="tabla_est" id='<c:out value="${estud_list.idestudio}"/>' align="center">
```

Figura 41

En el ejemplo anterior se observa el uso de muy poco código para crear una tabla con los valores a mostrar, se usa la librería Display-Tab para el trabajo con tablas y vinculada con los tabs del framework Spring se puede reducir el código al mínimo.

Los archivos de configuración son cargados en el web.xml de la aplicación para que siempre sean inicializados al arrancar el sistema. Ver **anexo 3**

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/mri-services.xml,
        /WEB-INF/mri-accesData.xml
    </param-value>
</context-param>
```

Figura 42

web.xml

Mediante los ejemplos de código se ve el uso de diferentes patrones, entre los cuales se pueden mencionar el patrón fachada, el MVC, el experto, el creador, bajo acoplamiento, inyección de dependencia a trabes de las configuraciones en los archivos xml, por citar algunos.

Para un mejor entendimiento del tema, ver los archivos anexos y estudiar cada una de las bibliografías relacionadas con los framework de desarrollo Spring e Hibernate.

4.4 Pantallas de la aplicación

Página Principal

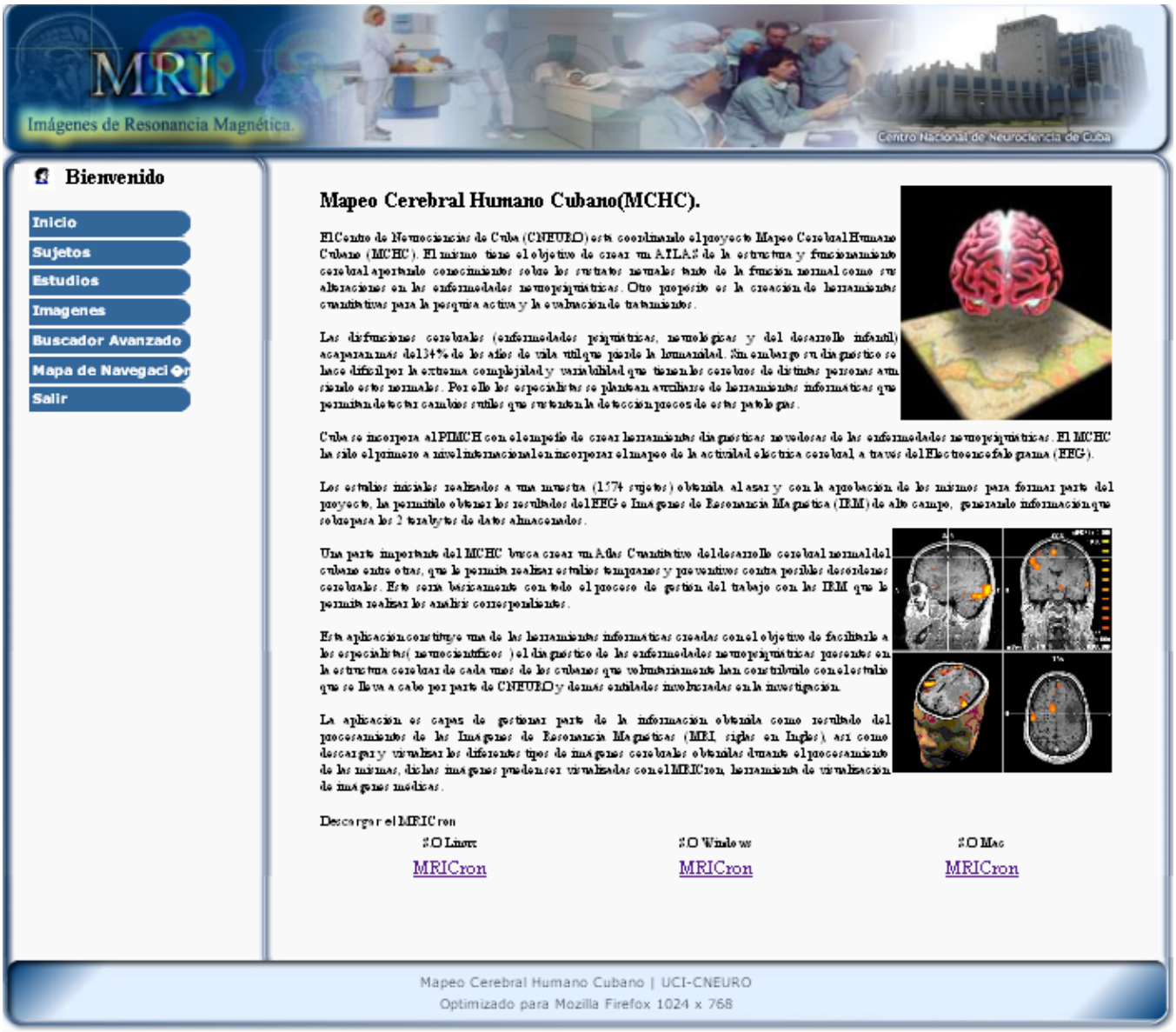


Figura 43

Manejar información de sujeto.

Listar Sujetos

Figura 44

Visualizar Expediente del Sujeto

Figura 45

Manejar estudios

Listar Estudios

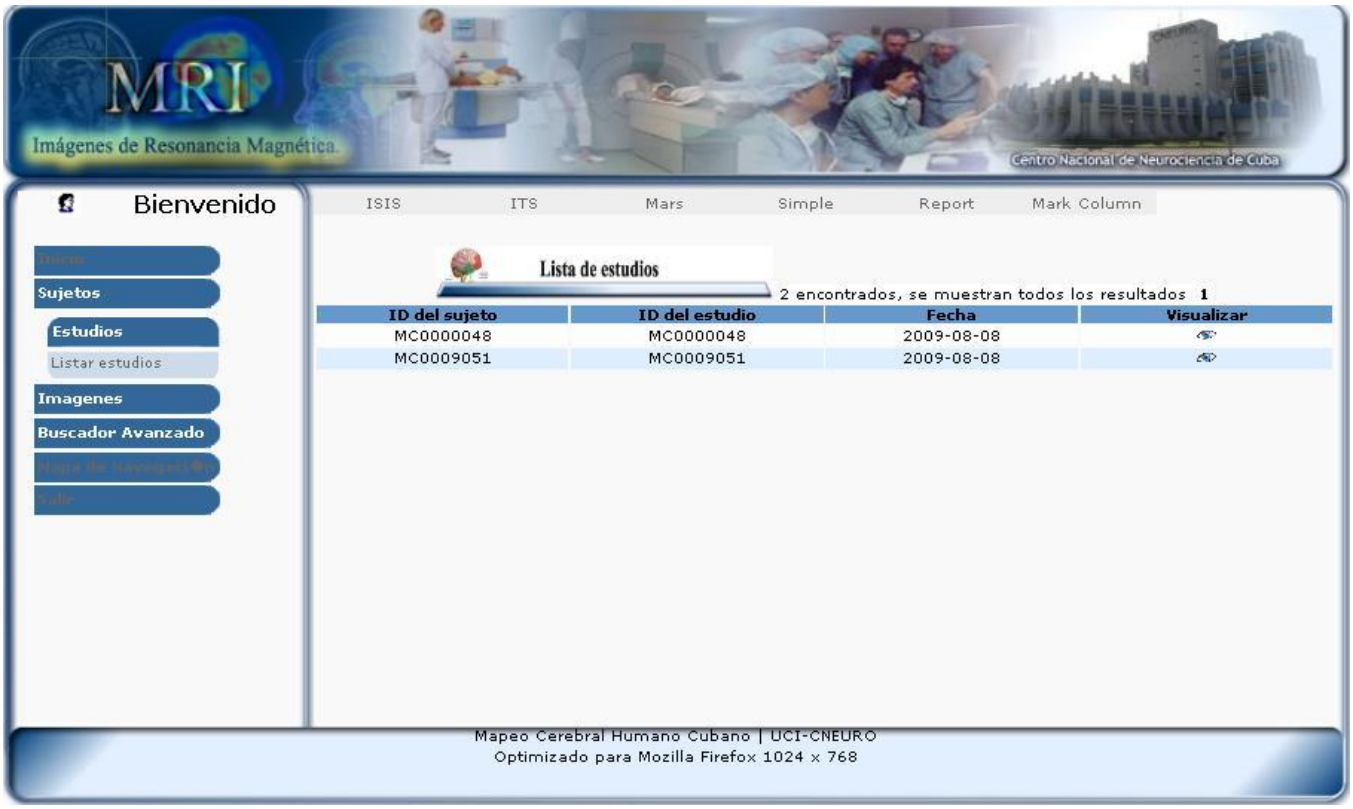


Figura 46

Visualizar Estudios

Figura 47

Manipular Información de Imágenes

Listar Imagen (Fase)

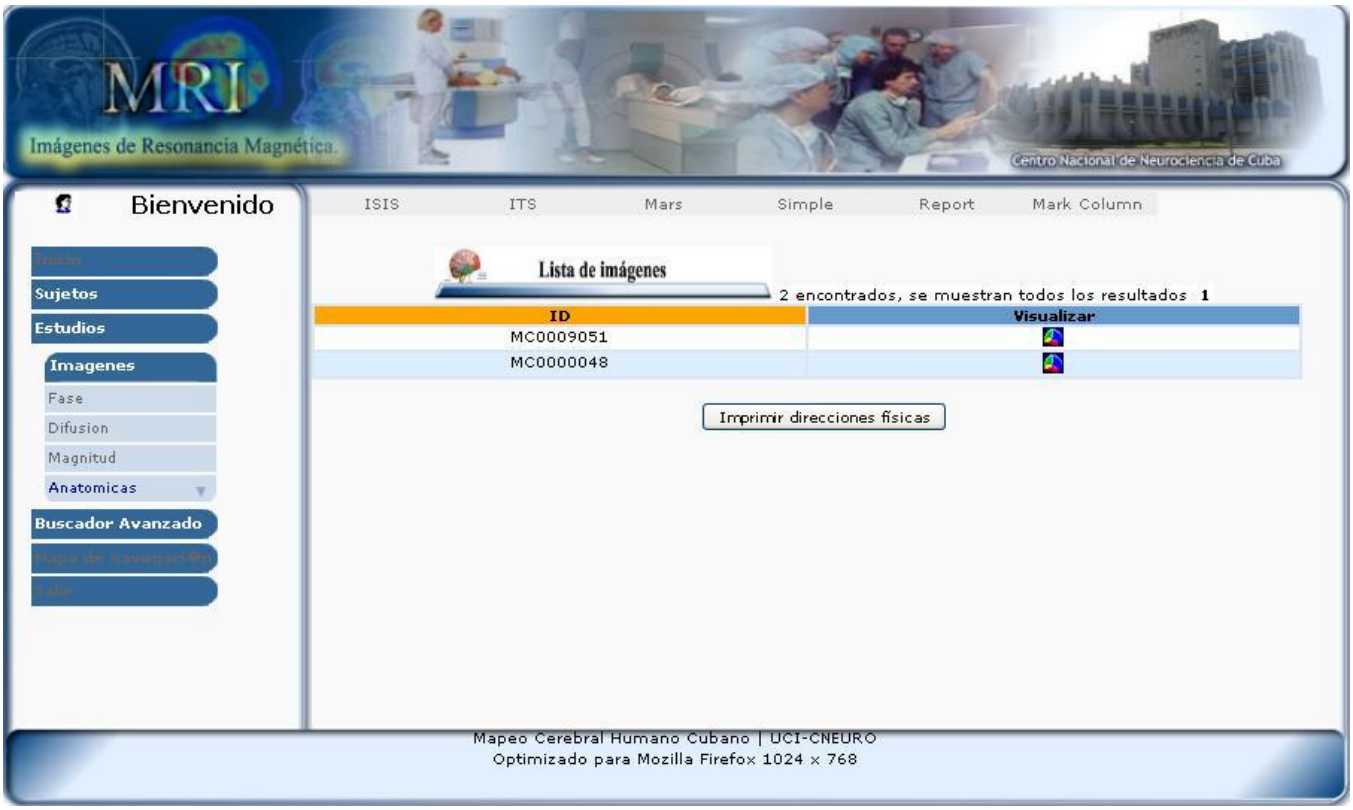


Figura 48

Visualizar Imagen

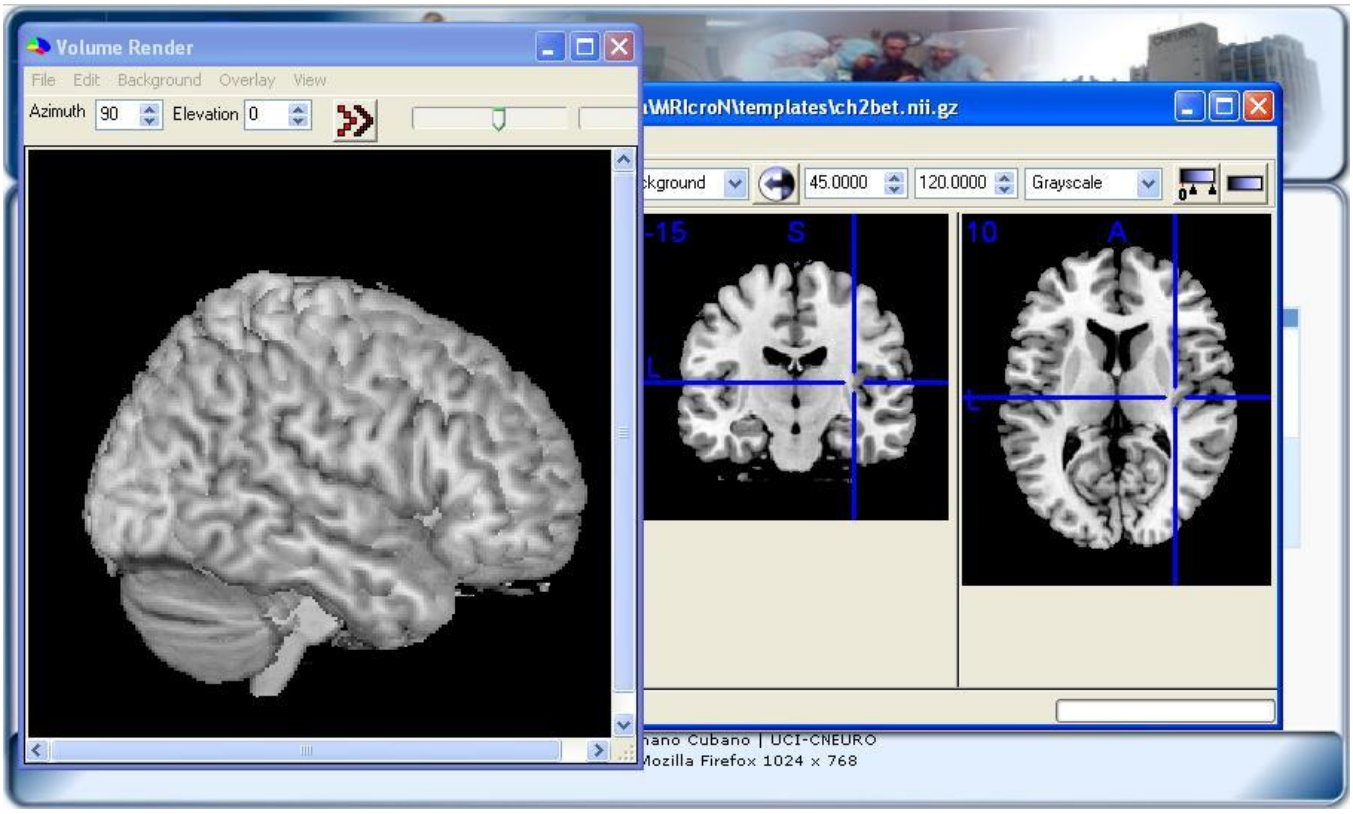


Figura 49

Realizar Búsqueda Avanzada de imágenes.

Figura 50

4.5 Casos de Prueba

Los casos de prueba son un conjunto de condiciones y variables mediante las cuales el analista puede determinar si un requisito cumple con las funcionalidades necesarias completa o parcialmente.

Para comprobar que todos los requisitos de una aplicación son revisados debe existir al menos un caso de prueba por cada requisito. Cuando se le realizan pruebas a un software se garantiza que este sea confiable y que requiera de la calidad necesaria.

Caso de Prueba del CU Manejar información de sujeto

Clases Válidas	Resultado	Resultado de la Prueba
<Se selecciona en la sección "Sujeto" la opción Buscar Sujeto y se entra el siguiente dato: 2>	<El sistema verifica que el dato entrado es correcto, busca al sujeto, muestra el ID del sujeto, nombre, los 2 apellidos y da la posibilidad de ver su expediente.>	<Satisfactorio>
<Se selecciona en la sección "Sujeto" la opción Buscar Sujeto y se entra el siguiente dato: 2e>	<El sistema verifica que el dato entrado es correcto, muestra un mensaje notificando que este ID no pertenece a ningún sujeto.>	<Satisfactorio>
<Se selecciona en la sección "Sujeto" la opción Buscar Sujeto y se entra el siguiente dato: 100>	<El sistema verifica que el dato entrado es correcto, muestra un mensaje notificando que este ID no pertenece a ningún sujeto.>	<Satisfactorio>

Figura 51

Caso de Prueba del CU Realizar Búsqueda Avanzada de Imágenes

Clases Válidas	Resultado	Resultado de la Prueba
<Se selecciona en la sección "Buscador Avanzado" la opción Buscador y se entran los siguientes datos>	<El sistema verifica que los datos entrados sean correctos, busca los sujeto que cumplan con estos parámetros, muestra el ID del sujeto, dirección física de donde	<Satisfactorio>

Sexo: M Raza: B Edad: 4 y 7 Manualidad: derecho Imagen: Anatómica	se encuentra la imagen y da la posibilidad de visualizarla.>	
<Se selecciona en la sección “Buscador Avanzado” la opción Buscador y se entran los siguientes datos> Sexo: M Raza: B Edad: 4 y 7 Manualidad: derecho Imagen: Ninguna	<El sistema verifica que los datos entrados sean correctos y muestra un mensaje notificando que debe llenar el campo de tipo de imagen para realizar la búsqueda.>	<Satisfactorio>

Figura 52

<Se selecciona en la sección "Buscador Avanzado" la opción Buscador y se entran los siguientes datos> Sexo: M Raza: B Edad: 7 y 4 Manualidad: derecho Imagen: Anatómica	<El sistema verifica que los datos entrados sean correctos y muestra un mensaje notificando que la primera edad no debe ser mayor que la segunda.>	<Satisfactorio>

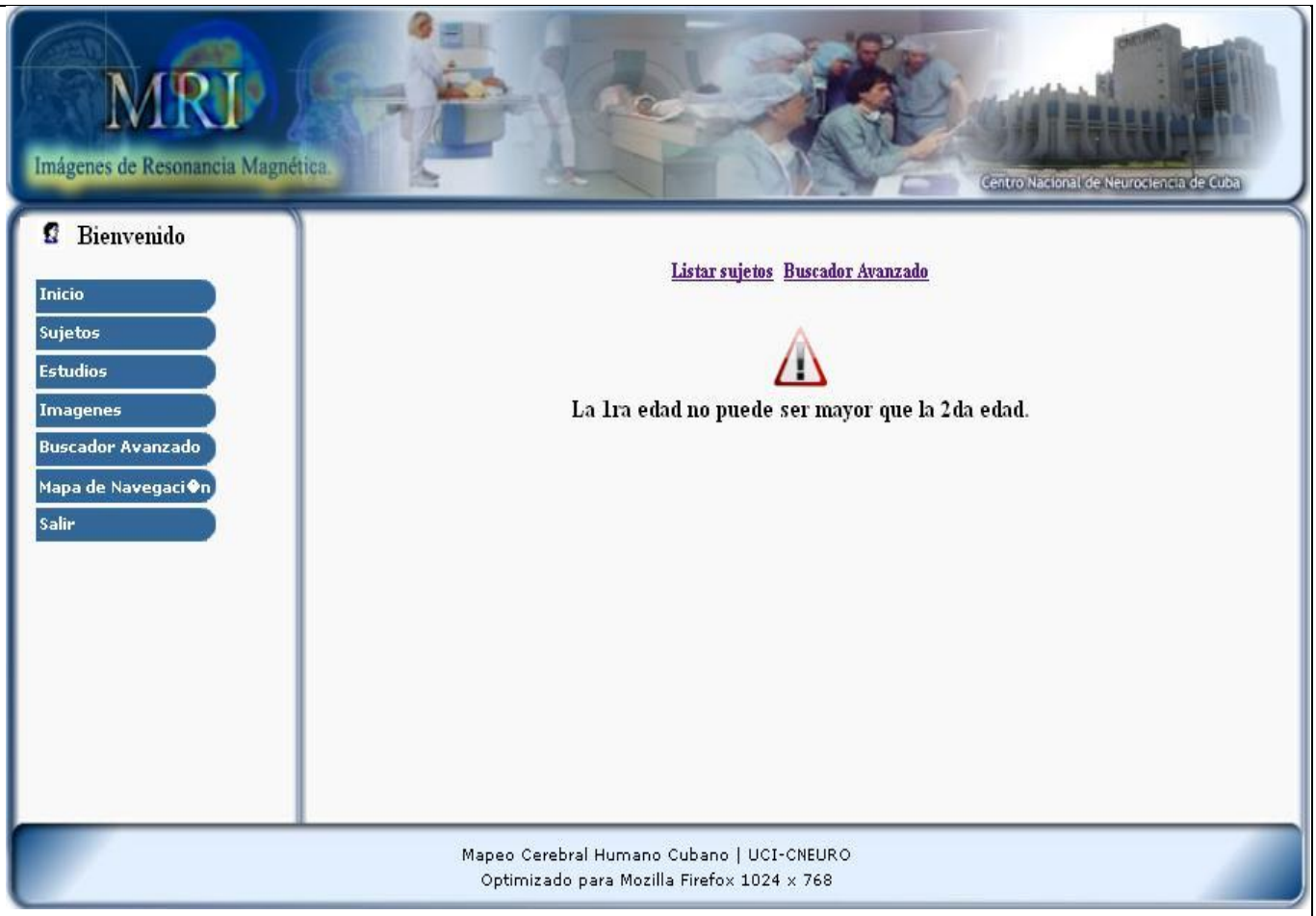


Figura 53

4.6 Conclusiones

Como resultado de este capítulo se obtuvieron los diagramas de componentes logrando mostrar las estructura y relaciones entre los componentes del sistema desarrollado. Se mostraron algunas de las pantallas funcionales de la aplicación para tener una visión gráfica del sistema. También se muestran los resultados de algunas pruebas que se le realizaron a dicho sistema para comprobar que cumple con las funcionalidades especificadas.

Conclusiones

- La tesis realizada logró automatizar la gestión de la información resultante del procesamiento de las imágenes de resonancia magnética.
- Fue diseñado el sistema informático y definido como se implementaría.
- Se implementó el módulo que permite la gestión de la información resultante del procesamiento de las imágenes de resonancia magnética.
- La propuesta fue validad a partir de las pruebas de caja negra realizadas a la aplicación.

Recomendaciones

Con el fin de contribuir a la continuidad de este trabajo se recomienda:

- Actualizar la aplicación agregándole nuevas funcionalidades y componentes según surjan necesidades en el centro CNEURO.
- Integrar el sistema al módulo principal del proyecto para establecerse un vínculo entre todos los módulos del mismo.

REFERENCIAS BIBLIOGRÁFICAS

1. ¿Qué es la gestión de la información? (3 de 4) 16 noviembre 2008 [Disponible en] http://informationmanagement.wordpress.com/2006/11/28/%c2%bfque-es-la-gestion-de-la-informacion-3-de-4/

2. Lic. Yuniet Rojas Mes De la gestión de información a la gestión del conocimiento, 16 noviembre 2008. [Disponible en] http://bvs.sld.cu/revistas/aci/vol14_1_06/aci02106.htm

3. Las Imágenes por Resonancia Magnética (su sigla en inglés es MRI) [En línea] 20 octubre 2008.

http://wo-pub2.med.cornell.edu/cgi-bin/WebObjects/PublicA.woa/1/wa/viewHContent?website=nyp+spanish&contentID=5400&wosid=9mDzhlxohbEXuTbzPy9QtM

4. Java Server Pages TM (JSP) [En línea] 20 noviembre 2008.

http://mipagina.cantv.net/williamyanez/jsp/default.htm

5. Capítulo 2: Características del lenguaje java. 23 noviembre 2008. [Disponible en] http://www.mailxmail.com/curso-java/caracteristicas-lenguaje-java-1

6. Miguel Ángel Manzanedo del Campo, Francisco José García Peñalvo. I.3. CARACTERÍSTICAS DE JAVA [En línea] 20 octubre 2008. http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.htm

7. Don Denoncourt. Elección del servidor de aplicaciones web [En línea] febrero 2003. http://www.help400.es/asp/scripts/nwart.asp?Num=131&Pag=10&Tip=T

8. Introducción - Bases de datos [En línea] 18 octubre 2008.

http://es.kioskea.net/contents/bdd/bddintro.php3

9. **Humberto Espinoza.** PostgreSQL. Una Alternativa de DBMS Open Source. 10 noviembre 2008. [Disponible en] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf

10. Germán Matosas, Fernando Telechea. Frameworks de Aplicación. [En línea] 20 noviembre 2008. www.asiap.org/jiap/presentaciones/bull.pps

11. **Mario Alfredo Sánchez Rico**. Spring, un framework de aplicación 16 octubre 2008 [Disponible en] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf

12. **Josefina Adriana Vázquez Rodríguez**. Creación de un repositorio para programas de Java utilizando la herramienta de ORM Hibernate, 24 abril 2008 [Disponible en]

http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/vazquez_r_ja/capitulo2.pdf

13. **Ing. Martín Tuesta Pereyra**. Herramientas CASE. [En línea] 26 noviembre 2009

http://www.unapiquitos.edu.pe/intranet/pagsphp/docentes/archivos/Capitulo%205%20-%20Herramientas%20CASE.doc?PHPSESSID=a8bf8196ddc53f256dff70e017d71fb1

14. Introducción a las Herramientas Case Introducción a las Herramientas Case. 25 noviembre 2008. [Disponible en] www.iesjorgemanrique.es/hda/descargas/tema1.ppt

15. Paradigma visual para UML (Plataforma Java) (Visual Paradigm for UML [Java Platform]) 6.0 [En línea] 10 diciembre 2008

http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

16. León Welicki. Patrones y Antipatrones: una Introducción - Parte I. 27 enero 2009. [Disponible en] http://msdn.microsoft.com/es-es/library/bb972242.aspx#M2

17. Larman, Craig. UML y Patrones

18. Ana Fernández Vilas. Diagrama de clases (estructura estática). [En línea] 23 abril 2009 http://tvdi.det.uvigo.es/~avilas/UML/node37.html

BIBLIOGRAFÍA

1. ¿Qué es la gestión de la información? (3 de 4) 16 noviembre 2008 [Disponible en] http://informationmanagement.wordpress.com/2006/11/28/%c2%bfque-es-la-gestion-de-la-informacion-3-de-4/

2. Lic. Yuniet Rojas Mes De la gestión de información a la gestión del conocimiento, 16 noviembre 2008. [Disponible en] http://bvs.sld.cu/revistas/aci/vol14_1_06/aci02106.htm

3. Las Imágenes por Resonancia Magnética (su sigla en inglés es MRI) [En línea] 20 octubre 2008.

http://wo-pub2.med.cornell.edu/cgi-bin/WebObjects/PublicA.woa/1/wa/viewHContent?website=nyp+spanish&contentID=5400&wosid=9mDzhlxohbEXuTbzPy9QtM

4. Java Server Pages TM (JSP) [En línea] 20 noviembre 2008.

http://mipagina.cantv.net/williamyanez/jsp/default.htm

5. Capítulo 2: Características del lenguaje java. 23 noviembre 2008. [Disponible en] http://www.mailxmail.com/curso-java/caracteristicas-lenguaje-java-1

6. Miguel Ángel Manzanedo del Campo, Francisco José García Peñalvo. I.3. CARACTERÍSTICAS DE JAVA [En línea] 20 octubre 2008. http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.htm

7. Don Denoncourt. Elección del servidor de aplicaciones web [En línea] febrero 2003. http://www.help400.es/asp/scripts/nwart.asp?Num=131&Pag=10&Tip=T

8. Introducción - Bases de datos [En línea] 18 octubre 2008.

http://es.kioskea.net/contents/bdd/bddintro.php3

9. **Humberto Espinoza.** PostgreSQL. Una Alternativa de DBMS Open Source. 10 noviembre 2008. [Disponible en] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf

10. Germán Matosas, Fernando Telechea. Frameworks de Aplicación. [En línea] 20 noviembre 2008. www.asiap.org/jiap/presentaciones/bull.pps

11. **Mario Alfredo Sánchez Rico**. Spring, un framework de aplicación 16 octubre 2008 [Disponible en] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf

12. **Josefina Adriana Vázquez Rodríguez**. Creación de un repositorio para programas de Java utilizando la herramienta de ORM Hibernate, 24 abril 2008 [Disponible en]

http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/vazquez_r_ja/capitulo2.pdf

13. **Ing. Martín Tuesta Pereyra**. Herramientas CASE. [En línea] 26 noviembre 2009 http://www.unapiquitos.edu.pe/intranet/pagsphp/docentes/archivos/Capitulo%205%20-%20Herramientas%20CASE.doc?PHPSESSID=a8bf8196ddc53f256dff70e017d71fb1

14. Introducción a las Herramientas Case Introducción a las Herramientas Case. 25 noviembre 2008. [Disponible en] www.iesjorgemanrique.es/hda/descargas/tema1.ppt

15. Paradigma visual para UML (Plataforma Java) (Visual Paradigm for UML [Java Platform]) 6.0 [En línea] 10 diciembre 2008

http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

16. León Welicki. Patrones y Antipatrones: una Introducción - Parte I. 27 enero 2009. [Disponible en] http://msdn.microsoft.com/es-es/library/bb972242.aspx#M2

17. Larman, Craig. UML y Patrones

18. Ana Fernández Vilas. Diagrama de clases (estructura estática). [En línea] 23 abril 2009 http://tvdi.det.uvigo.es/~avilas/UML/node37.html

19. Jorge. Sánchez. Sistema De Gestión Base De Datos.

http://www.slideshare.net/gabos/sistemadegestionbasededatos-jorgesanchez

20. Daniel M. Maldonado. ¿Qué es PostgreSQL? [En línea] 22 marzo 2009.

http://www.elcodigok.com.ar/2008/07/%C2%BFque-es-postgresql/

21. Marcus Eduardo Markiewicz, Carlos J.P. de Lucena. El Desarrollo del Framework Orientado al Objeto. [En línea] 16 febrero 2009 http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html

22. Herramientas CASE 10 abril 2009 [Disponible en]

http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htmç

23. DBDesigner Fork [En línea] 12 noviembre 2008. http://spanish.osstrans.net/software/dbdesigner-fork.html

24. DB Designer Fork 1.4, 12 noviembre 2008 [Disponible en]

http://www.todoprogramas.com/programa/dbdesignerfork

25. Ramiro Lago. Patrones de diseño software. [En línea] 23 Abril 2009. http://www.proactiva-calidad.com/java/patrones/index.html

26. Roles y artefactos 28 abril 2009. [En línea] http://epf.eclipse.org/wikis/openup/index.htm

27. ¿Qué es OpenUP / Basic? 28 abril 2009. [En línea]

http://www.cendesi.com/site/es/articles.php?lng=es&pg=11

28. Centro de Neurociencias - Historia. [En línea] 10 noviembre 2008

http://www.cneuro.co.cu/?q=es/cnch.

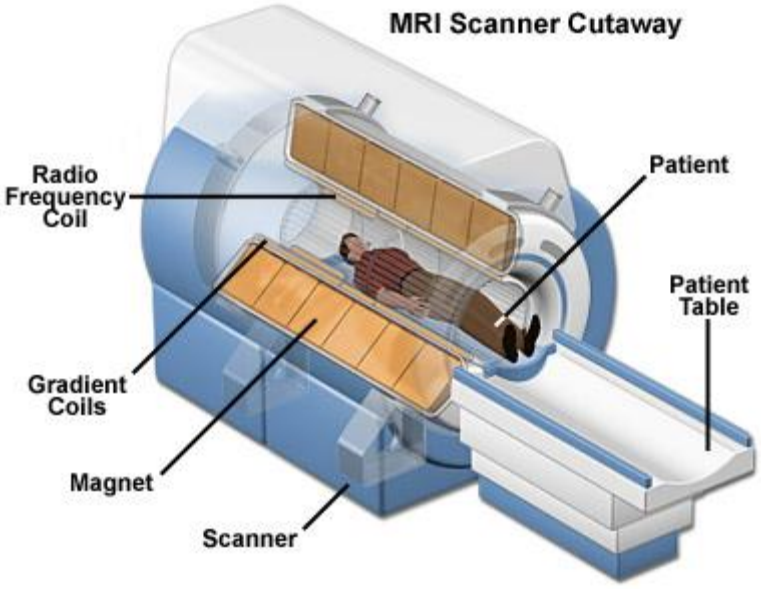
29. Ventajas de las aplicaciones web [Disponible en] 7 junio 2009

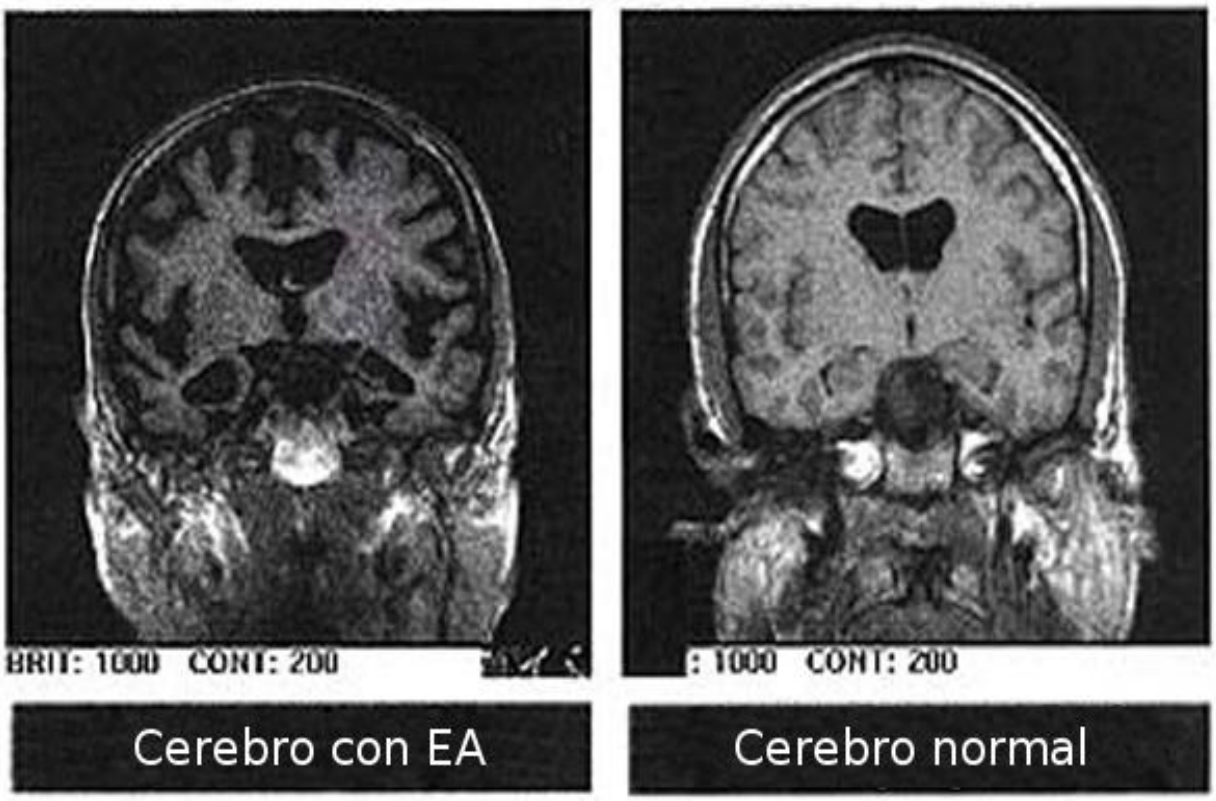
http://www.solcre.com/espanol/files/ventajas_de_las_aplicaciones_web.pdf

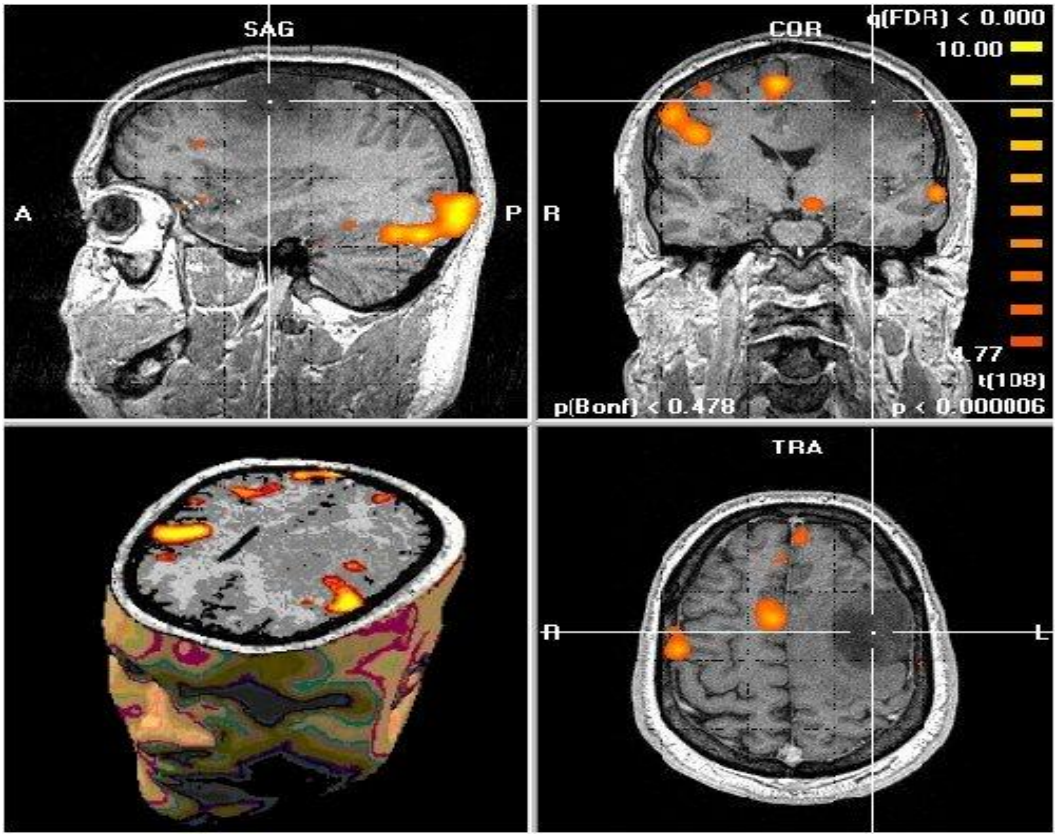
30. Daniel S. Marcus, Timothy R. Olsen, Mohana Ramaratnam, Randy L. Buckner. The Extensible Neuroimaging Archive Toolkit An Informatics Platform for Managing, Exploring,and Sharing Neuroimaging Data.

ANEXOS

Anexo 1 Imágenes y equipo MRI.






SAG
A
P
COR
R
q(FDR) < 0.000
10.00
4.77
t(108)
p(Bonf) < 0.478
p < 0.000006
TRA
R
L

Anexo 2 Configuración del mri – accesData.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

	<! ====  RESOURCE DEFINITIONS  ================ -->

	<bean id="propertyConfigurer"

		class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

		<property name="locations">

			<list>

				<value>

					/WEB-INF/classes/config/hibernate.properties

				</value>

			</list>

		</property>

	</bean>

	<bean id="sessionFactory"

		class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

		<property name="dataSource">

			<ref bean="dataSource" />

		</property>

		<property name="mappingResources">
```

```
                <list>
                        <value>mri/comun/dao/impl/mapeo/Anatomica.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Atlas.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Difusion.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Estudio.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Fase.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Magnitud.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Normalizacion.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Segmentacion.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Sujeto.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/TipoSerie.hbm.xml</value>
                        <value>mri/comun/dao/impl/mapeo/Usuario.hbm.xml</value>

<value>mri/comun/dao/impl/mapeo/ValoresVolumenesEstructuras.hbm.xml</value>
                </list>
        </property>
        <property name="hibernateProperties">
                <props>
                        <prop key="hibernate.dialect">${hibernate.dialect} </prop>
                        <prop key="hibernate.hbm2ddl.auto">${hibernate.hbm2ddl.auto}</prop>
                </props>
        </property>
```

```
</bean>
<bean id="dataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="url">
                <value>${hibernate.connection.url}</value>
        </property>
        <property name="driverClassName">
                <value>${hibernate.connection.driver_class}</value>
        </property>
        <property name="username">
                <value>${hibernate.connection.username}</value>
        </property>
        <property name="password">
                <value>${hibernate.connection.password}</value>
        </property>
        <property name="initialSize">
                <value>${dbcp.initialSize}</value>
        </property>
        <property name="maxActive">
                <value>${dbcp.maxActive}</value>
```

```
            </property>
            <property name="maxIdle">
                <value>${dbcp.maxIdle}</value>
            </property>
            <property name="minIdle">
                <value>${dbcp.minIdle}</value>
            </property>
            <property name="maxWait">
                <value>${dbcp.maxWait}</value>
            </property>
        </bean>
        <!==== DAO DEFINITIONS: HIBERNATE IMPLEMENTATIONS ======= -->
        <bean id="estudioDAO" class="mri.comun.dao.impl.EstudiosDAOImpl">
            <property name="sessionFactory">
                <ref bean="sessionFactory" />
            </property>
        </bean>
</beans>
```

Anexo 3 Configuración del web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">

        <display-name>Imágenes de Resonancia Magnética </display-name>

        <welcome-file-list>

                <welcome-file>redirect.jsp</welcome-file>

                <welcome-file>index.htm</welcome-file>

         </welcome-file-list>

        <error-page>

                <error-code>404</error-code>

                <location>

                        /Error.htm?error=404: La página a la que hace referencia no existe

                </location>

        </error-page>

        <servlet>

                <servlet-name>mri</servlet-name>

                <servlet-class>

                        org.springframework.web.servlet.DispatcherServlet

                </servlet-class>

                <load-on-startup>30</load-on-startup>

        </servlet>
```

```
        <servlet-mapping>
                <servlet-name>mri</servlet-name>
                <url-pattern>*.htm</url-pattern>
        </servlet-mapping>
        <context-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>
                        /WEB-INF/mri-services.xml,
                        /WEB-INF/mri-accesData.xml,
                        <!--
                        /WEB-INF/mri-security.xml,
                         -->
                </param-value>
        </context-param>
        <listener>
                <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>
        <listener>
                <listener-class>
                        org.springframework.web.util.Log4jConfigListener
                </listener-class>
        </listener>
</web-app>
```

GLOSARIO DE TÉRMINOS

BSD: Son las iniciales de Berkeley Software Distribution (en español, Distribución de Software Berkeley) y se utiliza para identificar un sistema operativo derivado del sistema Unix nacido a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.

CGI: Una tecnología que se usa en los servidores web.

Glándula pituitaria: La hipófisis o glándula pituitaria, (Aristóteles le atribuyó la función de secretar flema, en latín pituita, de allí el nombre pituitaria), es la glándula que controla el resto, entre ellas el tiroides. Es una glándula compleja que se aloja en un espacio óseo llamado silla turca del hueso esfenoides, situada en la base del cráneo, en la fosa cerebral media, que conecta con el hipotálamo a través del tallo pituitario o tallo hipofisario. Tiene un peso aproximado de 0,5 g.

HTML: siglas de **HyperText Markup Language** (*Lenguaje de Marcas de Hipertexto*), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

MRI: Una imagen por resonancia magnética (MRI, por sus siglas en inglés), también conocida como tomografía por resonancia magnética (TRM) es una técnica no invasiva que utiliza el fenómeno de la resonancia magnética para obtener información sobre la estructura y composición del cuerpo a analizar. Esta información es procesada por ordenadores y transformada en imágenes del interior de lo que se ha analizado.

Es utilizada principalmente en medicina para observar alteraciones en los tejidos y detectar cáncer y otras patologías. También es utilizada industrialmente para analizar la estructura de materiales tanto orgánicos como inorgánicos.

Pesquisa: Investigación o indagación encaminadas a descubrir algún cambio en el cerebro

Pipeline: Flujo de acciones que se lleva a cabo para el procesamiento de las imágenes.

Sustratos neurales: Diferentes tipos de sustancias que contiene el cerebro.