Universidad de las Ciencias Informáticas
Facultad 6

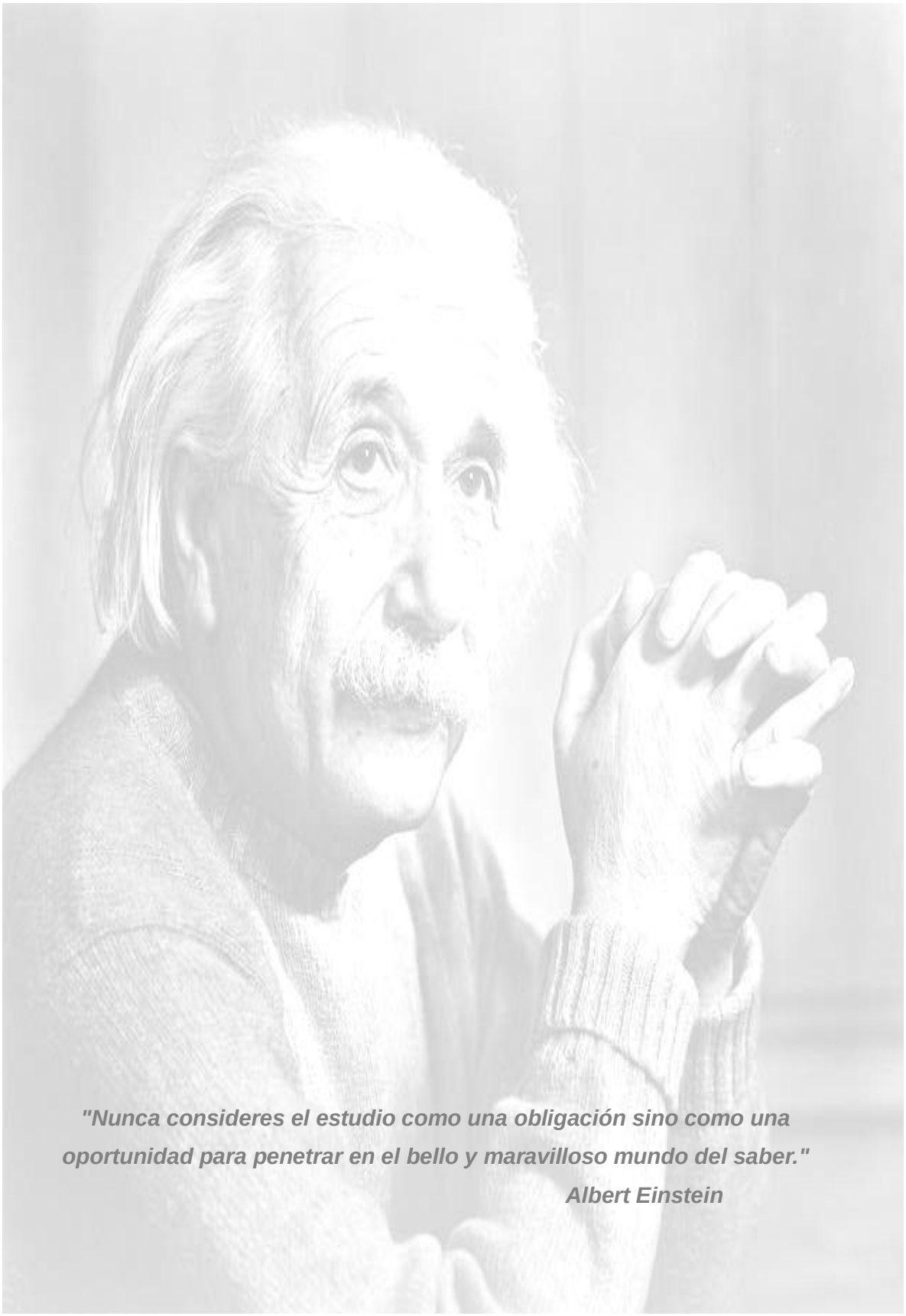
Título: "Solución informática para el tratamiento de imágenes de Resonancia Magnética en ambiente distribuido para el proyecto Mapeo Cerebral Humano Cubano."

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

Autor(es): Mirielys Ávila Leyva.
Raudelis Figueira Jiménez.

Tutor(es): Ing Yoamel Acosta Morales.
Co-tutor: Lic. Félix Argelio Martínez Nariño

Ciudad de la Habana, Junio del 2009



"Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

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 _____.

Mirielys Ávila Leyva	Raudelis Figueira Jiménez	Yoamel Acosta Morales
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

Datos de Contacto

Tutor:

Yoamel Acosta Morales (email yamorales@uci.cu**).**

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, en el año 2008. Se desempeña actualmente como Jefe de módulo en el proyecto de Mapeo Cerebral Humano Cubano.

Agradecimientos:

A nuestros padres, hermanos, tíos, primos y abuelos porque sin quererlo han sido autores de esta tesis; pues han sido nuestro faro y guía para poder realizarla.

A nuestro tutor Ing. Yoamel Acosta Morales y a Maikel pues sin su apoyo no hubiera sido posible la realización de esta tesis.

A nuestras amistades por toda la ayuda que nos han brindado.

A Yosbel nuestro agradecimiento especial, porque sin él y sin su ayuda, nuestra tesis no hubiese sido posible.

A Isbel por su apoyo incondicional.

Al profe Noel que tanto aportó para la realización de nuestra tesis.

A todos los que de una forma u otra aportaron su granito de arena en la realización de este trabajo y que no fueron mencionados.

A todos, Muchas Gracias.

Dedicatoria.

De Raudelis:

Dedico esta tesis a mi familia, en especial a mi mamá.

De Mirielys:

Dedico esta tesis a mi familia, en especial a mi papá, a mis hermanos Maikel y Miriel, a mi tío Nel y a Lela, pues sin ellos hoy no hubiera realizado el sueño de mi vida.

Resumen:

En la actualidad las ciencias informáticas ocupan un lugar muy importante dentro de la gama de ciencias a nivel mundial, sin embargo se trabaja bajo ciertas presiones con el objetivo de garantizar una mayor productividad y gestionar los recursos de una forma más eficiente, sobre todo cuando el escenario tecnológico varía con gran rapidez. En el marco del proyecto en conjunto con CNEURO y la Facultad de Bioinformática de la UCI se prevé realizar el procesamiento de imágenes de resonancia magnética a un considerable número de estudios realizados a disímiles pacientes. Estos procesamientos son de un elevado costo computacional, por lo que surge la idea de diseñar e implementar un módulo capaz de gestionar estos procesamientos de forma distribuida. Se analizó, diseñó e implementó un módulo para realizar dicho procesamiento utilizando las herramientas DODICOM e IBASPM facilitadas por el Centro de Neurociencias para Linux, con significativos tiempos de respuesta teniendo en cuenta la cantidad de recursos disponibles en el momento en que se envíe la petición, y la magnitud y complejidad de la misma. La aplicación se implementó haciendo uso del lenguaje Java. Para el desarrollo del módulo se utilizaron herramientas libres para garantizar el beneficio gratuito de la aplicación.

Palabras claves:

Sistemas distribuidos, procesamiento, imágenes, resonancia magnética.

Índice

Introducción:

Dos de los grandes retos de la ciencia en el siglo XXI son: el proyecto del Genoma Humano y el del Mapeo Cerebral Humano. Al concluir el primero de ellos, considerado hoy como una gran hazaña, surge otro de igual envergadura, este último tiene como objetivo fundamental crear una enorme base de datos que contenga toda la información que se conoce del cerebro humano hasta el momento.

El Proyecto Mapeo Cerebral Humano fue lanzado en el año 1993 con el propósito de avanzar en el discernimiento e interpretación de la relación que existe entre la estructura y el funcionamiento del cerebro humano. Los científicos en este campo pretenden profundizar en la comprensión de los procesos físicos que acompañan al conocimiento y percepción humano. Estos resultados pueden ser aplicados inmediatamente al tratamiento de desórdenes neurológicos, psicológicos y psiquiátricos en una fase precoz, y por consiguiente, en el logro de una calidad de vida superior de los pacientes afectados. **[1]**

Debido a que el estudio del cerebro humano ha ido evolucionando rápidamente, se pueden detectar a tiempo una serie de enfermedades que pueden afectar a los seres humanos, ejemplo de ellos son: tumores, enfermedades vasculares cerebrales o ictus, mal de Alzheimer, epilepsia, meningitis, esclerosis múltiple, hipoxia cerebral, enfermedades de Huntington, hidrocefalia, esquizofrenia entre otras.

Todos estos padecimientos pueden recibir el tratamiento adecuado si se aplican las técnicas más sofisticadas para detectarlos. Ejemplo de algunas de estas técnicas son: Magneto encefalografía (MEG), Potenciales evocados, Electroencefalografía (EEG), Melografía, Ecografía Doppler, Angiografía cerebral o arteriografía, Tomografía computarizada por emisión de fotón único (TCEFU), Tomografía por emisión de positrones (TEP), Ecoencefalografía, Tomografía computarizada (TC), y la Resonancia magnética (RM).

La Resonancia Magnética se realiza colocando la cabeza o el cuerpo del paciente en un espacio reducido donde el cráneo y médula se someten a intenso campo magnético (no recurre a rayos X) que prácticamente no representa riesgos. La calidad de las imágenes es excelente y sus resultados son más efectivos que los de la TC para detectar trastornos graves, como enfermedad cerebral vascular, tumores cerebrales, malformaciones y esclerosis múltiple.

Los principales inconvenientes son su precio elevado y la lentitud de la obtención de imágenes (de 10 a 45 minutos), además de que está contraindicada en individuos que utilizan respirador, sufren claustrofobia o son portadores de marcapasos cardíacos y prótesis metálicas. **[2]**

El Centro de Neurociencia se ha dado a la tarea de llevar a cabo un estudio sobre el cerebro humano cubano, tomando como muestra a una pequeña parte de la población cubana (el municipio de La Lisa).

Como parte de dicho estudio se les ha realizado a estos sujetos una serie de pruebas entre las que se pueden encontrar la de Resonancia Magnética. Las imágenes obtenidas en dicha prueba deben ser procesadas para obtener mayor información de las mismas, pero este proceso demora mucho tiempo actualmente, sobre todo cuando el número de estudios es sumamente elevado.

Esta situación conlleva al siguiente **problema científico**: ¿Cómo reducir los tiempos de procesamiento de las imágenes de Resonancia Magnética para el Proyecto Cubano de Mapeo Cerebral?

El objeto de estudio: el proceso de tratamiento de las imágenes de Resonancia Magnética del proyecto MCHC.

El campo de acción: el proceso de tratamiento de las imágenes de Resonancia Magnética del módulo MRI.

El objetivo general de este trabajo es desarrollar un servicio para el Proyecto Cubano de Mapeo Cerebral Humano que reduzca el tiempo de procesamiento de las imágenes de Resonancia Magnética en ambiente distribuido.

Los objetivos específicos propuestos son:

- Analizar el proceso de tratamiento de imágenes en ambiente distribuido.
- Definir los requisitos con los que debe cumplir dicho servicio.
- Diseñar el sistema a ser implementado.
- Implementar un servicio que cumpla con las exigencias del cliente.
- Validar el servicio propuesto.

Para la correcta realización de este servicio se plantearon las siguientes **tareas**:

- Definición de los requerimientos funcionales y no funcionales del sistema.
- Revisión de las posibles soluciones disponibles que puedan resolver el problema planteado.
- Análisis y selección de las herramientas apropiadas para el desarrollo de la solución informática.
- Selección de la metodología de desarrollo de software adecuada.
- Realización del diseño del sistema.
- Implementación del diseño del sistema.
- Realización de pruebas de caja negra y caja blanca.
- Validación del servicio que se propone.

La tesis está estructurada en Resumen, Introducción, cuatro capítulos como cuerpo fundamental de la tesis, Conclusiones, Recomendaciones, Referencias bibliográficas y Bibliografía. Los capítulos corresponden a diferentes aspectos temáticos, por ejemplo:

Capítulo 1: Fundamentación Teórica.

En este capítulo se hace referencia a las herramientas, lenguajes de programación, gestor de base de datos, metodología, lenguaje de modelado y herramienta CASE que se usarán a lo largo de la concepción del software.

Capítulo 2: Características del sistema.

En este capítulo se hace una reseña de los procesos a automatizar, además de explicar cómo debe funcionar el sistema. Se hace una lista de los requerimientos funcionales y no funcionales que se deben tener en cuenta a la hora de realizar el software. Se identifican los casos de uso del sistema con sus respectivas descripciones.

Capítulo 3: Análisis y Diseño del sistema.

En este capítulo se hace una especificación de los requisitos con el objetivo de describir cómo implementar el sistema, a través del diseño. Se realizan los diagramas de clases del diseño y diagramas interacción más relevantes. Se muestra además el diagrama de clases persistentes, para tener un mejor conocimiento sobre la base de datos. El diseño en este capítulo queda definido y descrito, de modo tal que se define el patrón de diseño y otras técnicas a utilizar para el desarrollo del módulo.

Capítulo 4: Implementación y Prueba.

En el capítulo se detalla toda información necesaria para llevar a cabo la implementación del servicio. Se describe el sistema y se implementa en términos de componentes organizados de acuerdo a los nodos específicos en el modelo de despliegue. Se realizan las pruebas correspondientes para mejorar la calidad del software y para comprobar que cumple correctamente con los requisitos especificados.

Capítulo 1: Fundamentación teórica

1.1 Introducción.

La bioinformática y la neurociencia tienen una estrecha relación. En este capítulo se analiza la importancia del presente servicio para el desarrollo de las investigaciones relacionadas con el tema. Además se hace una pequeña reseña sobre aspectos muy importantes para la realización del software, como son la metodología y las diferentes herramientas y lenguajes utilizados durante el proceso.

1.2 Sistemas distribuidos, aplicación en la bioinformática.

Actualmente el procesamiento de las imágenes de resonancia magnética en el Centro de Neurociencias de Cuba es un poco lento y tedioso, sobre todo cuando se trata de procesar un alto número de estudios de diferentes sujetos. Para lograr aminorar el tiempo de procesamiento de estas imágenes de resonancia magnética se hace uso de un sistema distribuido que no es más que una colección de ordenadores autónomos enlazados por una red y soportados por aplicaciones que hacen que la colección actúe como un servicio integrado **[3]**. La construcción de estos presenta una solución que aumenta las posibilidades, ya no estando sujetos a las restricciones de una máquina, se estará en condiciones de utilizar y explotar mejor los recursos de toda la red.

La necesidad de aprovechar los recursos disponibles en los sistemas informáticos conectados a la red y simplificar su utilización ha dado lugar a una nueva forma de tecnología de la información conocida como *Grid Computing*. De este modo, los sistemas distribuidos se pueden emplear como un único sistema virtual en aplicaciones intensivas en datos o con alta demanda computacional. Las necesidades básicas de la neurociencia se pueden expresar como una mayor demanda de capacidad de almacenamiento y tratamiento de información, y un crecimiento desmesurado de la capacidad de cálculo. En otras palabras, la forma más efectiva, en coste y recursos, de resolver los problemas actuales de las Ciencias de la Vida y por extensión, responder a las crecientes demandas sociales sobre ellas, es recurrir a sistemas masivamente distribuidos de tecnología Grid **[4].**

1.2.1 Posibilidades y limitaciones que brindan los sistemas distribuidos.

Características:

1. **Concurrencia**.- Esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.

2. **Carencia de reloj global**.- Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, está más bien distribuida a los componentes.
3. **Fallos independientes de los componentes**.- Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando.

En general el desarrollo de sistemas distribuidos intenta poner solución a los siguientes objetivos:

1. Transparencia.
2. Fiabilidad.
3. Rendimiento.
4. Capacidad de crecimiento.
5. Flexibilidad.
6. Seguridad.

1.2.2 Ventajas y desventajas de los sistemas distribuidos.

Ventajas

1. Aumento de la disponibilidad.
2. Mejora del desempeño.
3. Balanceo en la carga de trabajo.
4. Compartición de recursos.
5. Compartición de información.
6. Confiabilidad, disponibilidad y tolerancia a fallas.
7. Modularidad en el desarrollo.
8. Flexibilidad.
9. Crecimiento incremental.
10. Reducción de costos.
11. Mayor capacidad de modelar estructuras organizacionales.

Desventajas

1. Capacidad reducida para administrar apropiadamente grupos de procesadores y memoria localizada en distintos sitios.
2. Enorme dependencia del desempeño de la red y de la confiabilidad de la misma.

3. Debilitamiento de la seguridad.
4. Mayor complejidad en la administración y mantenimiento.
5. Mayor complejidad en su construcción **[5].**

1.3 Procesos de Desarrollo de Software

El proceso de desarrollo de un software es un proceso largo y un poco difícil de controlar, es por ello que se requiere de una metodología de desarrollo que brinde la posibilidad de cumplir satisfactoriamente con la producción del software así como con las necesidades del cliente. Entre las metodologías más conocidas mundialmente podemos citar a RUP (Rational Unified Process), XP (Extreme Programming), y OpenUP.

1.3.1 Rational Unified Process (RUP).

Es un proceso bien definido, estructurado y adaptable a las características y necesidades de cada proyecto específico. RUP es un proceso iterativo e incremental que se encarga de dividir el trabajo en partes más pequeñas o en mini proyectos. Este proceso se divide en 4 fases para el desarrollo de 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 obtener la capacidad operacional inicial.
- Transición: El objetivo es llegar a obtener la liberación del proyecto.

RUP hace uso de arquitectura basada en componentes, permite el modelado visual del software y la verificación de la calidad del software, además pretende implementar las mejores prácticas en Ingeniería de Software. Además brinda un proceso integrado que utiliza el estándar de notación UML para permitir desarrollar un proceso de forma iterativa e incremental a partir de la identificación e implementación de los casos de uso. Es uno de los procesos más generales de los existentes actualmente, ya que está pensado para adaptarse a cualquier proyecto.

1.3.2 Extreme Programming (XP).

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP.

Entre las características fundamentales de este proceso de desarrollo se tienen:

- Pruebas unitarias: esta característica está basada en realizar pruebas a los procesos principales para poder detectar posibles errores que puedan suceder.
- Refabricación: esta característica se basa en la reutilización de código lo que le permite ser más flexible al cambio.
- Programación en pares: esta característica se basa en que dos desarrolladores participen en un proyecto en una misma estación de trabajo con el propósito de lograr un mismo objetivo, la satisfacción del cliente.

¿Qué propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Esta metodología permite la comunicación entre los desarrolladores y el cliente, simplifica el desarrollo de módulos y posibilita una retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. No existe documentación del proyecto lo que más se acerca a la documentación son las historias de usuario, pero al concluir el proyecto se descartan. Inclusive se recomienda hacer dos secciones, una con todas las historias de usuario que faltan desarrollar, y otra donde se archiven las concluidas, esto aproximará el estado de avance del proyecto.

1.3.3 Open Unified Process (OpenUP/Basic)

Para llevar a cabo este servicio se utilizará la metodología Open Unified Process (OpenUP/Basic) que no es más que un proceso de desarrollo unificado que está basado en Rational Unified Process (RUP). Mantiene las mismas características de RUP, que contiene el desarrollo iterativo, casos de uso y escenarios de conducción de desarrollo, gestión de riesgos, y el enfoque centrado en la arquitectura.

OpenUP/Basic es un proceso de desarrollo de software de código abierto diseñado para pequeños equipos organizados quienes quieren tomar una aproximación ágil del desarrollo. OpenUP/Basic es un proceso iterativo que es Mínimo, Completo, y Extensible. Se valora la colaboración y el aporte de los stakeholders sobre los entregables y la formalidad innecesarios.

El OpenUP estructura el ciclo de vida de un proyecto en 4 fases: concepción, elaboración, construcción y transición. El ciclo de vida del proyecto provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos.

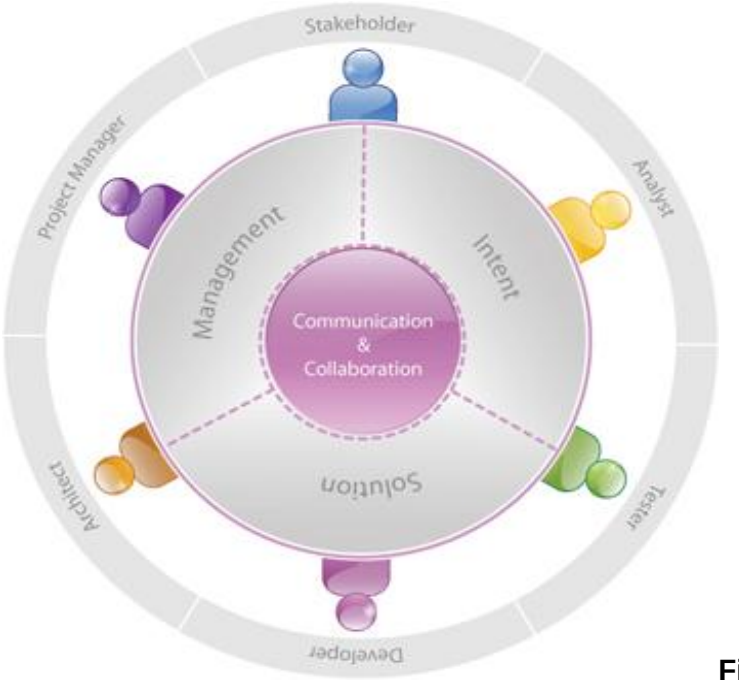


Figura 1. OpenUP/Basic.

OpenUP/Basic es un proceso unificado que incorpora técnicas ágiles probadas. El resultado es un proceso estructurado, robusto, eficiente y liviano**. [6]**

¿Por qué usar OpenUP/Basic?

El módulo MRI del Proyecto Mapeo Cerebral requiere de un tiempo relativamente corto de desarrollo es por ello que se decide optar por la metodología OpenUP/Basic basada en RUP. Es una metodología ágil que se aplica a proyectos de corta duración, diseñado para pequeños equipos de trabajo, y con una duración de 3-6 meses de esfuerzo de desarrollo. OpenUP preserva las tres características esenciales de RUP: centrado en la arquitectura, dirigido por Casos de Uso e iterativo e incremental, con la diferencia de que solo se toman las partes necesarias e imprescindibles teniendo como resultado un proceso muy simple que cumple con los principios fundamentales de la metodología RUP.

1.4 Tendencia de los roles.

1.4.1 Roles.

Los dos roles principales a desempeñar son:

Analista: Las personas en este rol representan al cliente y los usuarios finales involucrados, obteniendo información desde los stakeholders para entender el problema a ser resuelto, capturar y

ajustar las prioridades para los requerimientos. Este rol puede ser asignado en equipos ágiles y pequeños, es frecuentemente compartido entre varios miembros del equipo que también desempeñan otros roles, y además, uno o más miembros del equipo desempeñan este rol exclusivamente. Esta alternativa es comúnmente adoptada cuando los requerimientos complejos son difíciles de capturar. **[7]** Desarrollador: La persona en este rol es responsable por desarrollar una parte del sistema, incluyendo diseñar esta, para que se ajuste a la arquitectura, posiblemente prototipar la interfaz de usuario y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución. Una persona que desempeñe este papel puede tener conocimientos especializados en una determinada área técnica, pero también deben tener una amplia comprensión de todas las tecnologías que intervienen para poder trabajar con otros miembros del equipo técnico. [7]

1.4.2 Artefactos por roles.

Artefactos generados por el Analista:

Glosario: Este artefacto define términos importantes usados por el proyecto. Estos términos son las bases para una colaboración efectiva con los stakeholders y otros miembros del equipo.

Requisitos: Este artefacto captura requisitos generales del sistema no capturados en los escenarios o casos de uso, incluyendo requisitos sobre atributos de calidad y requisitos funcionales globales. [7]

Modelo de Casos de Uso: Este artefacto es la base para el acuerdo entre las partes interesadas y el equipo del proyecto en relación con la funcionalidad para el sistema. También ayuda a orientar las diversas tareas en el ciclo de vida de desarrollo de software. [7]

Casos de Uso: Este artefacto captura la secuencia de acciones que un sistema realiza, que produce un resultado observable de valor a su interacción con el sistema. Los Casos de Uso capturan el comportamiento requerido del sistema desde la perspectiva del usuario final, alcanzar una o más metas.

Artefactos generados por el Desarrollador:

Build: Este artefacto es una versión operativa de un sistema o parte de un sistema que demuestra un subconjunto de las capacidades que se incluirá en el producto final. Esta versión ejecutable del sistema suele tener un número de archivos de apoyo que también se consideran parte de este artefacto compuesto. En un ciclo de vida iterativo, cada iteración debe evolucionar la construcción de la iteración anterior a construir, añadiendo más funcionalidad y mejorar la calidad. [7]

Diseño**:** Este artefacto describe la realización de la funcionalidad necesaria del sistema en términos de componentes y sirve como una abstracción del código fuente. El objetivo es describir los elementos del sistema a fin de que puedan ser examinados y entendidos en un sentido que no sea posible la lectura

del código fuente. Se comunica abstracciones particulares de parte de la aplicación y puede describir un encapsulado subsistema, un alto nivel de análisis del sistema, un punto de vista del sistema en un solo contexto, u otras perspectivas que explican una solución a un problema específico que necesita ser comunicado. [7]

Implementación: Este artefacto es la colección de uno o más de estos elementos: archivos de código fuente, archivos de datos, crear scripts y el resto de archivos que se transforman en el sistema ejecutable. El objetivo de este artefacto es representar las partes físicas que componen el sistema a ser construido, organizado de una manera que sea comprensible y manejable. [7]

1.5 Lenguaje de modelado.

Se utilizó el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) debido a que la metodología de desarrollo seleccionada lo propone como lenguaje de notación. UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software orientado a objetos.

1.6 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras**. [8]**

1.6.1 Visual Paradigm for UML

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

- Eclipse/IBM WebSphere
- JBuilder
- NetBeans IDE
- Oracle JDeveloper
- BEA Weblogic **[9]**

Esta herramienta soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. **[10]**

Esta herramienta tiene unas características gráficas muy cómodas que facilitan la realización de los diagramas de modelado. Entre otras características importantes que tiene es la integración con algunos IDE de programación como Eclipse desarrollado por IBM, NetBeans de Sun o JBuilder de Borland.

El servicio a desarrollar se hará sobre el sistema operativo GNU/Linux, es por esta razón que se requiere de esta herramienta CASE para hacer el modelado de este trabajo.

1.7 Lenguajes de programación.

1.7.1 Lenguaje de programación Java.

El lenguaje Java se creó con varios objetivos fundamentales entre los que se puede citar que usa como metodología de programación la orientada a objetos, toma lo mejor de otros lenguajes orientados a objetos como C++, y que permite la ejecución de un mismo programa en múltiples sistemas operativos.

Este lenguaje tiene una serie de características muy útiles para el desarrollo de este trabajo:

- Distribuido: Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.
- Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).
- Robusto: Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.
- Indiferente a la arquitectura: Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.
- Portable: La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

- Multihebra: Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos. **[11]**

1.8 Herramientas a utilizar

1.8.1 Eclipse.

Es un entorno de desarrollo integrado de código abierto multiplataforma basado en Java. Fue desarrollado por IBM y su código fuente fue puesto a disposición de los usuarios.

Es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en plug-ins **[12],** lo que le permite integrar diversos lenguajes sobre un mismo IDE, además de introducir otras aplicaciones accesorias.

Entre las ventajas fundamentales que se le pueden atribuir se encuentra el hecho de que conserva un registro de versiones, además de otras características que se mencionan a continuación:

- Posee un editor visual con sintaxis coloreada.
- Posee compilación incremental de código.
- Modifica e inspecciona valores de variables.
- Avisa de los errores cometidos mediante una ventana secundaria. **[12]**

Otra de las características claves de Eclipse es su extensibilidad, pues a medida que se le van adicionando los plug-ins, va aumentando su funcionalidad. Eclipse permite completar métodos en el editor de Java, así como detectar errores de compilación en vivo.

La librería Jakarta ofrece un conjunto de soluciones del lado del servidor usando el lenguaje Java. Contiene un conjunto de clases implementadas que son de gran utilidad y que ahorran tiempo al programador.

Para desarrollar este servicio se hace necesario hacer conexiones a un servidor FTP con el objetivo de almacenar y recuperar archivos desde el mismo. Esta programación es un tanto engorrosa sin hacer uso de la librería Jakarta. Esta librería contiene diferentes clases como la FTPClient y la FTPFile cuyos métodos facilitan la implementación de dicha conexión, así como las operaciones a realizar.

1.8.2 Matlab 7.0.

(MATrix LABoratory, “laboratorio de matrices”) es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X. Es una poderosa herramienta para la resolución numérica de problemas. **[13]**

Es un excelente software matemático el cuál va orientado a matrices las cuáles posee más de 1000 funciones predefinidas dentro del mismo, listas para ser utilizadas. Además de eso, Matlab posee una gran variedad de herramienta extras como el Simulink el cuál es una simulación multidominio, y el Guide con el que se puede realizar un sin fin de interfaces para usuarios. **[14]**

1.10 Gestor de Base de Datos.

PostgreSQL es un servidor de base de datos relacional orientado a objetos de software libre, publicado bajo la licencia BSD. **[15]**

Esta herramienta tiene una serie de ventajas, por lo cual se decidió usarla como gestor de base de datos: El código fuente está disponible para todos sin mayores costos, es multiplataforma y además tiene una estrategia de almacenamiento de filas llamada MVCC muy útil para conseguir mejores respuestas cuando existen grandes volúmenes de información. **[16]**

1.10.1 PostgreSQL frente a MySQL:

Es más rápido y más eficiente que MySQL.

Tiene views.

Soporta tamaños de filas y bases de datos ilimitados.

Soporta herencia.

Tiene soporte para unicode.

Resiste caídas del sistema y cortes de luz.

Tiene triggers y rules.

Tiene soporte para conexiones encriptadas SSL.

1.10.2 PostgreSQL frente a Oracle:

Es gratuito y de código abierto.

No tiene un esquema de licencia complicado como Oracle.

Soporta herencia.

Permite indexar textos para realizar búsquedas sofisticadas a través de OpenFTS.

Puede ser traducido.

Tiene soporte para conexiones encriptadas SSL. **[17]**

1.11 Conclusiones.

Luego de estudiar todas las posibilidades existentes y poniéndolas con las necesidades del cliente, se puede arribar a una solución que respalde las exigencias de los neurocientíficos. Luego de un exhaustivo análisis y comparación se decide seguir la siguiente línea de trabajo:

- Como metodología para el desarrollo del software: OpenUP, ya que es una metodología ágil para el desarrollo de proyectos de corta duración.
- Como lenguaje de modelado, UML, ya que crea un lenguaje común para todos los desarrollos y la documentación que crea también es común, por lo que puede ser extendida por cualquier desarrollador que tenga conocimientos del lenguaje de modelado.
- La herramienta CASE utilizada para modelar el programa, Visual Paradigm.
- Como IDE se escogió eclipse desarrollado por IBM, que tiene una amplia gama de posibilidades para el desarrollo de este tipo de aplicaciones.
- El lenguaje de programación que se utilizó fue java, aprovechando RMI y sus posibilidades a la invocación de tareas remotas, así como por su posibilidad de uso gratuito.
- Para realizar el procesamiento de las imágenes de resonancia magnética se hizo uso de las herramientas DODICOM e IBASPM facilitadas por el Centro de Neurociencias de Cuba.

Capítulo 2: Características del sistema.

2.1 Introducción.

En este capítulo se hace una descripción del objeto de estudio y se muestra el modelo de dominio para un mayor entendimiento del cliente. Se especifican los requerimientos funcionales y no funcionales referentes al presente módulo en desarrollo. Se realizan además las descripciones de los casos de uso que representan las funcionalidades del sistema a desarrollar.

2.2 El proyecto de Mapeo Cerebral Humano Cubano hoy día.

El Centro de Neurociencias de Cuba pretende llevar a cabo el proyecto de Mapeo Cerebral Humano Cubano con la finalidad de crear una enorme base de datos que le permita sacar conclusiones médicas importantes sobre las diferentes funcionalidades del cerebro y las disímiles enfermedades asociadas al mismo. Para desarrollar este proyecto, los científicos de este centro realizan una serie de pruebas (como la de Resonancia Magnética), y obtienen como resultado un conjunto de imágenes en formato DICOMS (.dcm).

Estas imágenes son almacenadas en una computadora para luego ser procesadas. Ese procesamiento actualmente se realiza de forma manual y es un poco engorroso. Luego de ser almacenadas las imágenes de cada estudio, los científicos del centro utilizan la herramienta DODICOM para convertirlas a formato NIFTI (.nii), este proceso da como resultado una serie de imágenes divididas en cuatro carpetas (Anatómica, Difusión, Fase y Magnitud) de las cuales se toma la carpeta Anatómica que sirve de entrada a otra herramienta llamada IBASPM para un último procesamiento que arroja como resultados otra serie de imágenes almacenadas también en cuatro carpetas llamadas Atlas, Normalización, Segmentación y Valor Volumen Estructura. Este proceso demora de 3 a 4 horas por cada estudio que se procese, es por ello que se hace necesario realizar el procesamiento de forma automática y distribuida, para lograr aminorar el tiempo dedicado al procesamiento de los estudios.

¿Qué son las imágenes DICOMS?

DICOMS (Digital Imaging and Communications in Medicine) es un estándar reconocido mundialmente para el intercambio de imágenes médicas pensado para el manejo, almacenamiento, impresión y transmisión de imágenes médicas. **[18]**

Describe el formato de archivos y la especificación de los datos primordiales de un paciente en la imagen así como el encabezado requeridos, describiendo un lenguaje común a distintos sistemas médicos. De esta forma las imágenes vienen acompañadas de mediciones, cálculos e información descriptiva relevantes para diagnósticos. Utiliza archivos con extensión .dcm.

Un solo archivo de DICOM contiene una cabecera que almacena la información sobre el nombre del paciente, el tipo de exploración, etc. así como todos los datos de la imagen que pueden contener la información en tres dimensiones.

El formato genérico del archivo de DICOM consiste en dos partes: Header seguido inmediatamente por un Data Set de DICOM. El Data Set de DICOM contiene la imagen o las imágenes especificadas. El Header contiene sintaxis de transferencia UID (identificador único) que especifica la codificación y la compresión del Data Set **[19]**.

NIFTI: Es un formato estándar para imágenes que fue diseñado para el análisis científico de las imágenes cerebrales. El formato es simple, compacto y versátil. Las imágenes se pueden almacenar como un par de archivos (hdr / img) compatibles con la mayoría de los formatos, o simplemente como un solo archivo (NII). **[20]**

2.3 Modelo de Dominio.

El Modelo de Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los eventos que suceden en el entorno en que trabaja el sistema. Los conceptos definidos para dicho modelo son:

- Especialista: Persona encargada de dirigir el procesamiento manualmente.
- Imágenes MRI: Conjunto de imágenes obtenidas al realizar la prueba de resonancia magnética a los diferentes individuos.
- DODICOM: Herramienta utilizada para el procesamiento de las imágenes DICOMS.
- Imágenes NIFTI: Conjunto de imágenes resultantes del procesamiento de las imágenes DICOMS en la herramienta DODICOM y que sirven de entrada para la herramienta IBASPM.
- Anatómica: Tipo de imagen NIFTI.
- Difusión: Tipo de imagen NIFTI.
- Fase: Tipo de imagen NIFTI.
- Magnitud: Tipo de imagen NIFTI.
- IBASPM: Herramienta que procesa las imágenes NIFTI.
- Atlas: Imágenes resultantes del procesamiento de las primeras imágenes NIFTI mediante la herramienta IBASPM.
- Normalización: Imágenes resultantes del procesamiento de las primeras imágenes NIFTI mediante la herramienta IBASPM.

- Segmentación: Imágenes resultantes del procesamiento de las primeras imágenes NIFTI mediante la herramienta IBASPM.
- Valor Volúmenes Estructuras: Imágenes resultantes del procesamiento de las primeras imágenes NIFTI mediante la herramienta IBASPM.

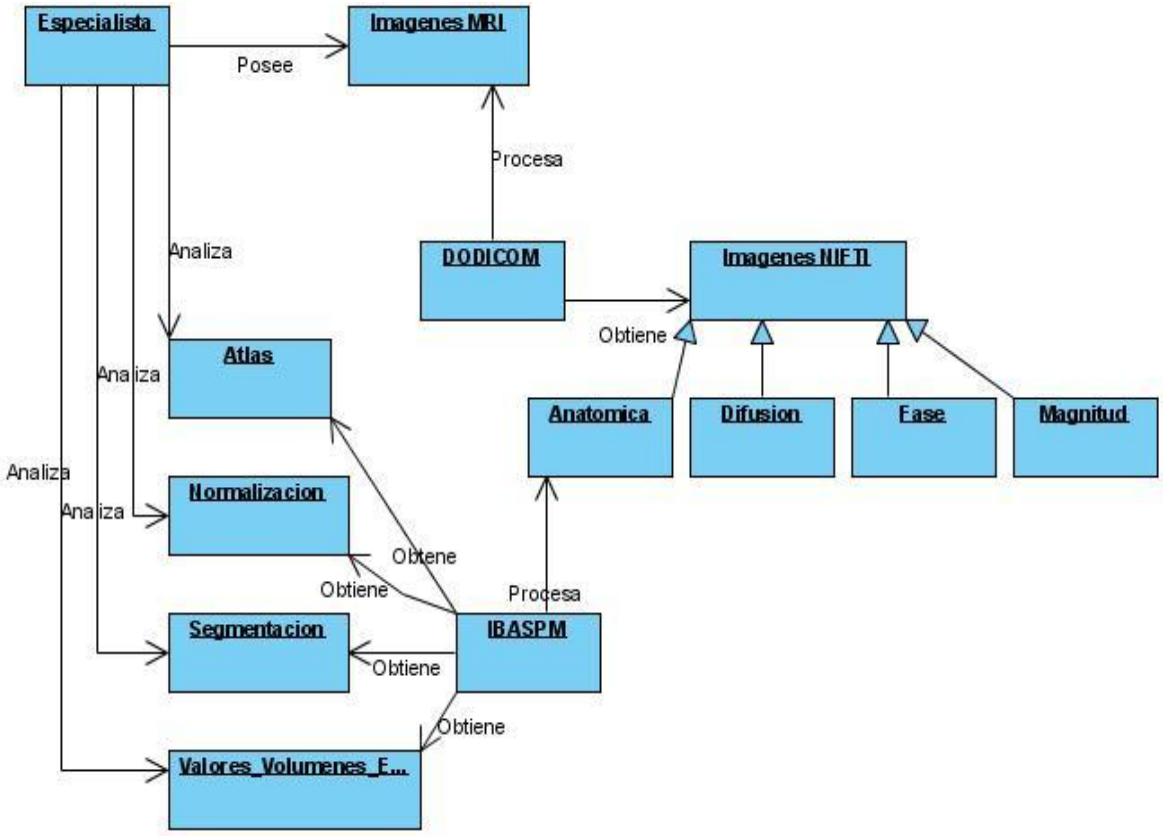


Figura 2. Modelo de Dominio.

El objetivo que se pretende alcanzar con este trabajo es el de crear un servicio para el proyecto de Mapeo Cerebral Humano Cubano que se encargará de funcionar como un centinela, es decir, este servicio estará constantemente velando, en el servidor FTP, por la llegada de nuevos estudios al mismo. En caso de que la respuesta sea positiva el servicio ordena el procesamiento de los nuevos estudios sobre la plataforma de cómputo distribuida. Para ello se hace necesario copiar los ficheros del estudio a procesar en la máquina asignada por la plataforma donde serán procesados, luego se realiza el procesamiento de dichas imágenes utilizando para ello las librerías de Matlab facilitadas por el Centro de Neurociencia de Cuba. Estos resultados son de vital importancia para los especialistas de este centro científico ya que les permitirá realizar diagnósticos, hacer comparaciones entre diferentes estudios, en fin, llegar a conclusiones importantes derivadas del resultado del procesamiento de los estudios. Por tal motivo se hace necesario almacenar toda esa información, por lo que el servicio se conecta nuevamente al servidor FTP y copia los ficheros resultantes del procesamiento de los estudios en sus respectivas carpetas, luego se conecta a la base de datos y actualiza los campos de las tablas con las diferentes direcciones físicas donde se encuentran los resultados del procesamiento.

2.4 Especificaciones de los requerimientos del software.

Los casos de uso se emplean para capturar el comportamiento deseado del sistema en desarrollo, sin tener que especificar cómo se implementa ese comportamiento. Proporcionan un medio para que los desarrolladores, los usuarios finales del sistema y los expertos del dominio lleguen a una comprensión común del sistema.

Además ayudan a validar la arquitectura y a verificar el sistema mientras evoluciona a lo largo del desarrollo. Por lo general el nombre de un caso de uso comienza con un verbo en infinitivo. Un caso de uso describe un proceso de principio a fin, es decir, una secuencia de eventos, las acciones y las transacciones que se requieren para realizarlo.

2.4.1 Requisitos Funcionales.

RF1: Buscar nuevos estudios.

RF2: Procesar nuevos estudios.

RF3: Actualizar servidor FTP.

RF4: Actualizar base de datos.

2.4.2 Requisitos no funcionales.

- **Requerimientos Legales:** Este servicio, junto con toda la información generada acerca del mismo durante su desarrollo pertenece a la Universidad de las Ciencias Informáticas en conjunto con el Centro de Neurociencias de Cuba.
- **Requerimientos de Disponibilidad:** Debe estar disponible las 24 horas del día y los siete días de la semana.
- **Requerimientos de Software:** Sistema Operativo Ubuntu 7.04, debe tener instalado el Matlab 7.0, la plataforma de cómputo distribuido versión 1.0 (T-Arenal), el Eclipse 3.2 y el Java Runtime Enviroment (JRE) versión 1.5.
- **Requerimientos de Hardware:** Procesador Pentium 4 o superior, 512 MB de memoria RAM como mínimo.

2.5 Definición de los casos de uso.

Actores	Justificación
Reloj del sistema	Es el encargado de, cada cierto tiempo, activar el proceso para buscar nuevos estudios en el servidor FTP.

Tabla 1. Definición de los Actores del Sistema

Para llevar a cabo el desarrollo de este servicio se hizo un exhaustivo análisis del problema para lograr determinar los casos de uso que serian necesarios para la implementación del mismo. Como resultado de dicho análisis se obtuvo un solo caso de uso denominado Procesar Estudio el cual se describe a continuación.

CU-1	Procesar Estudio
Actor	Reloj del Sistema
Descripción	Este caso de uso se encarga de hacer una búsqueda de nuevos estudios, en caso de que el resultado sea positivo se encargaría de mandar a procesar estos estudios utilizando las herramientas DODICOM e IBASPM, luego del procesamiento, guarda los resultados en un servidor FTP y actualiza la base de datos con las direcciones físicas donde se encuentran los resultados de dicho procesamiento.
Referencia	RF1, RF2, RF3, RF4

Tabla 2. Listado de Casos de Usos del Sistema

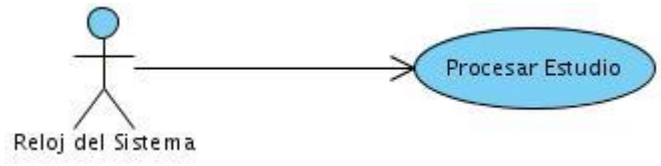


Figura 3. Diagrama de Caso de Uso

Caso de uso	
CU-1	Procesar Estudio
Propósito	Realizar el procesamiento de los estudios nuevos llegados al servidor FTP, y actualizar la base de datos con las direcciones físicas donde se encuentran almacenados los resultados de dicho procesamiento.
Actores	Reloj del sistema
Resumen:	Este caso de uso se encarga de hacer una búsqueda de nuevos estudios, en caso de que el resultado sea positivo se encargaría de mandar a procesar estos estudios utilizando las herramientas DODICOM e IBASPM, luego del procesamiento, guarda los resultados en un servidor FTP y actualiza la base de datos con las direcciones físicas donde se encuentran los resultados de dicho procesamiento.
Referencias	RF1, RF2, RF3, RF4
Flujo Normal de Eventos	
Sección ”Buscar nuevos estudios”	
Acción del actor	Respuesta del sistema
1. El reloj activa el proceso.	2. El proceso se conecta al servidor FTP.
	3. Hace una búsqueda de nuevos estudios.
	4. Ir a Sección “Procesar estudios”.
Flujo alternativo	
Acción del actor	Respuesta del sistema

	3.1. El proceso espera a ser activado nuevamente por el reloj del sistema.
Flujo Normal de Eventos	
Sección ”Procesar estudios”	
Acción del actor	Respuesta del sistema
	1. Ordena el procesamiento de los estudios, por separado, sobre la plataforma de cómputo distribuida.
	2. Copia los ficheros de los estudios en las computadoras asignadas por la plataforma.
	3. Realiza el procesamiento del estudio.
	4. Ir a Sección” Guardar Resultado del Procesamiento”.
Flujo alternativo	
Acción del actor	Respuesta del sistema
Flujo Normal de Eventos	
Sección ” Guardar Resultado del Procesamiento”	
Acción del Actor	Respuesta del Sistema
	1. Se conecta al servidor FTP.
	2. Copia los ficheros resultantes del procesamiento de los estudios.
	3. Se conecta a la base de datos.
	4. Actualiza la base de datos con las direcciones físicas de los datos generados en el procesamiento de los estudios.
	5. Ir a Sección” Buscar nuevos estudios”
Flujos Alternos	
Acción del Actor	Respuesta del Sistema

Tabla 3. Descripción del Caso de Uso Procesar Estudio.

2.6 Conclusiones.

Siguiendo los pasos que plantea UML y debido a que la metodología empleada no realiza el proceso de negocio, se definieron en este capítulo conceptos, que fueron relacionados mediante un diagrama de modelo del dominio. El mismo muestra de forma general los distintos objetos existentes en el negocio. Se definieron los requisitos que debe cumplir el sistema para su correcto funcionamiento. Determinado así las diferentes funcionalidades del mismo resumidas como caso de uso, del que se realizó una descripción detallada para su mejor comprensión.

Capitulo 3: Análisis y Diseño.

3.1 Introducción.

En este capítulo se presenta el diagrama de clases del diseño que dará solución al caso de uso identificado y descrito en el capítulo anterior y se realizarán los diagramas de interacción para los diferentes escenarios. Finalmente se presentará el diagrama de despliegue, con el objetivo de tener una idea más clara de cómo quedará distribuida la infraestructura una vez desplegada.

3.2 Representación gráfica del diagrama de clases del diseño.

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces (por ejemplo, las de Java) en una aplicación.

Contiene la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.**[21]**

A continuación se muestra el diagrama de clases del diseño correspondiente al servicio que se está implementando. Este diagrama cuenta con 16 clases del diseño, de ellas 6 son clases controladoras y el resto son clases entidades correspondientes a la base de datos. De cada clase se muestran los atributos y métodos que la conforman, los cuales serán detallados en un próximo epígrafe. Este diagrama no cuenta con interfaces gráficas ya que lo que se está implementando es un servicio que se estará ejecutando de forma oculta constantemente.

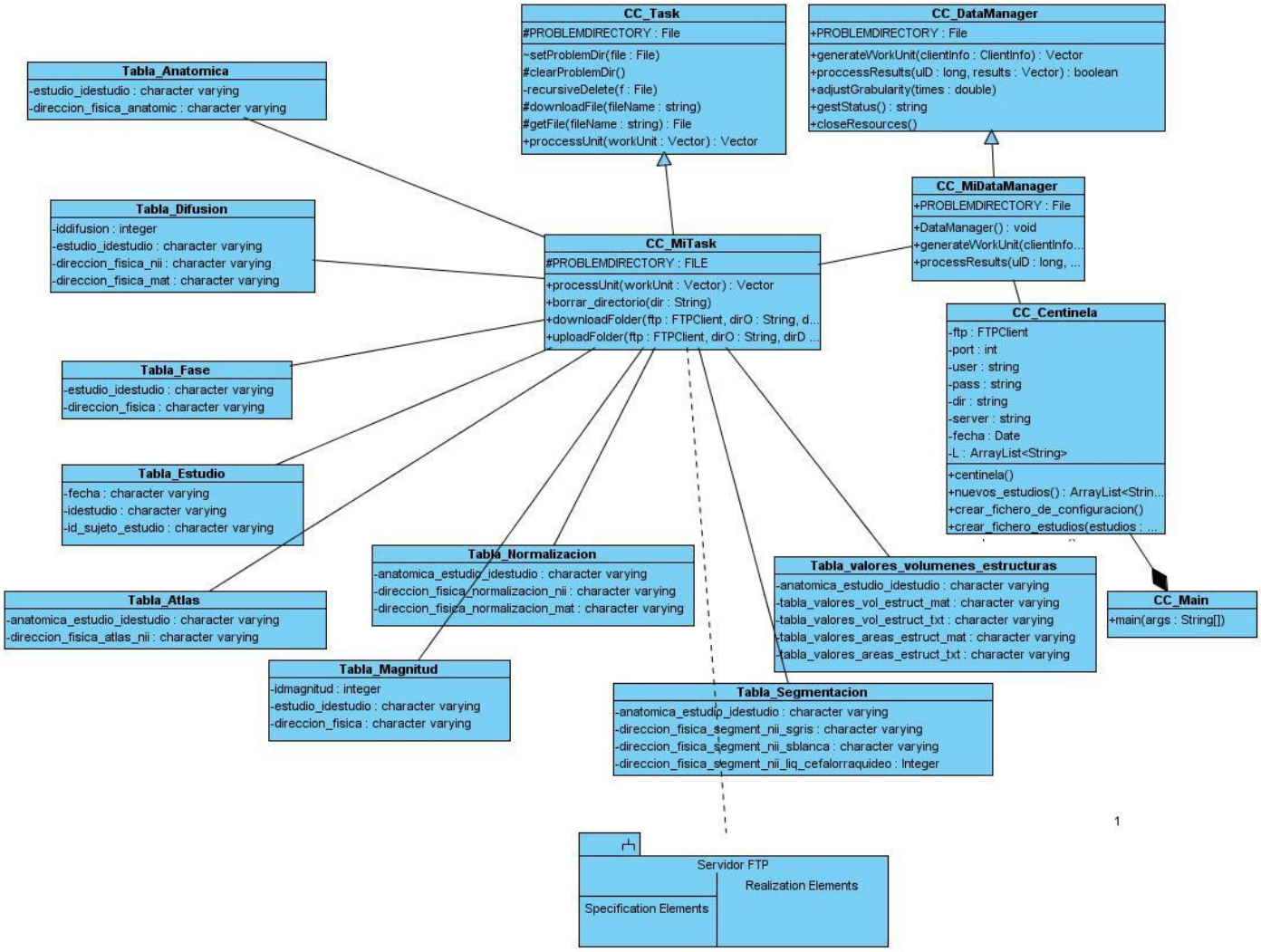


Figura 4. Diagrama de Clases del Diseño.

3.3 Diagrama de Secuencia.

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. El mostrar los componentes tiene sentido ya que se trata de objetos reutilizables, en cuanto a los casos de uso hay que recordar que se implementan como objetos cuyo rol es encapsular lo definido en el caso de uso. **[22]**.

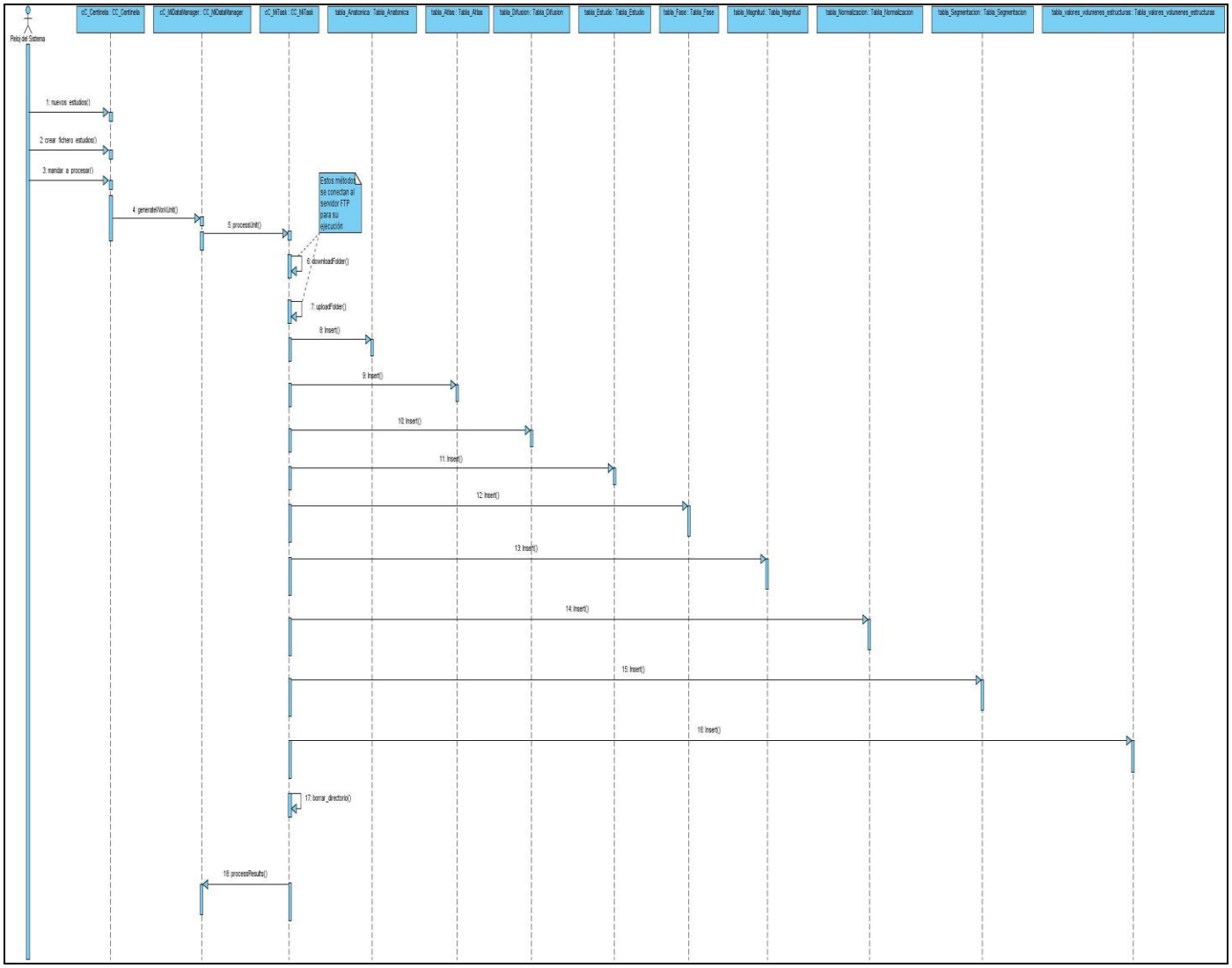


Figura 5. Diagrama de Secuencia del Diseño. Flujo Normal.

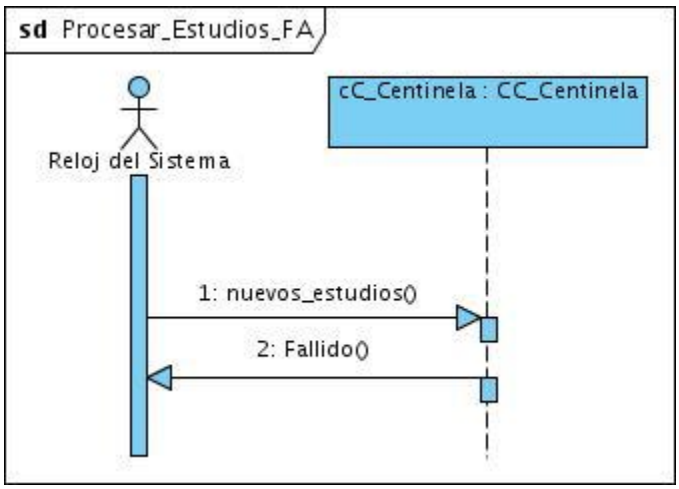


Figura 6. Diagrama de Secuencia del Diseño. Flujo Alterno.

3.4 Descripción de las clases del diseño.

Nombre: CC _Centinela	
Tipo de clase: Controladora	
Atributo	**Tipo**
1. ftp	● FTPClient
2. port	● int
3. user	● string
4. pass	● string
5. dir	● string
6. server	● string
7. fecha	● Date
8. L	● ArrayList <String>
Para cada responsabilidad:	
Nombre:	Centinela(): void
Descripción:	Constructor por defecto
Nombre:	nuevos estudios(): ArrayList <String>
Descripción:	Devuelve una lista con el nombre de todos los estudios nuevos encontrados.
Nombre:	Mandar_a_procesar()

Descripción:	Manda a procesar los estudios para la plataforma de cómputo distribuida.
Nombre:	Crear_fichero_de_configuracion()
Descripción:	Crea a un fichero de configuración con los datos para realizar las conexiones al servidor FTP, al servidor de base de datos y al de la plataforma de cómputo distribuido.

Tabla 4. Descripción de la CC _Centinela.

Nombre: CC_ DataManager	
Tipo de clase: Controladora	
Atributo	**Tipo**
1. File	● PROBLEMDIRECTORY
Para cada responsabilidad:	
Nombre:	DataManager(): void
Descripción:	Constructor por defecto
Nombre:	generateWorkUnit(clientInfo: ClientInfo): Vector
Descripción:	Es llamado por el sistema cada vez que un cliente solicita una unidad de trabajo para procesar. El tipo de dato que devuelve es java.util.Vector. El algoritmo que está corriendo en el cliente recibe este java.util.Vector que no es más que la unidad de trabajo que debe computar. Por lo tanto, cuando todos los elementos contenidos en el java.util.Vector son enviados, deben ser casteados a su tipo de dato original al llegar al Task. Si en algún momento no hay unidades de trabajo disponibles para el problema, esta función retornará null. Esto le indica al servidor de que el problema no tiene unidades de trabajo en ese momento.
Nombre:	processResults(ulD: long, results: Vector): boolean
Descripción:	Es invocado cada vez que un cliente retorna un conjunto de resultados. Esta función tiene dos parámetros. El primero es el ID que identifica a la unidad de trabajo (su tipo es java.lang.Long) generada por el método generateWorkUnit (). El segundo parámetro (de tipo

	java.util.Vector) es el conjunto de resultados que envió el algoritmo que se ejecuta en el cliente. En caso de que el cálculo distribuido termine, este método debe devolver true, de lo contrario devuelve false. Cuando este método devuelve true, el servidor elimina del sistema de cómputo al problema, comprime el directorio de trabajo y lo coloca a disposición del usuario para su descarga.
Nombre:	adjustGranularity(times: double): void
Descripción:	Se llama periódicamente en el servidor y recibe un parámetro correspondiente al porcentaje (positivo o negativo) de cuanto la granularidad paralela debe ajustarse de forma que la medida del tiempo de procesamiento coincida con el tiempo de procesamiento óptimo.
Nombre:	getStatus(): string
Descripción:	Se llama cada vez que un usuario hace una solicitud acerca del estado de un problema en particular al servidor. La información que devuelve este método es del tipo java.lang.String, y se espera que contenga información útil acerca del estado actual del cálculo.
Nombre:	CloseResources(): void
Descripción:	Es invocado justo antes de que el problema se elimine del sistema. El objetivo de este método es cerrar cualquier recurso, por ejemplo: archivos, directorios, entrada/salida de las corridas, que puedan estar abiertas. El usuario debe implementar el código necesario para cerrar cualquier recurso, de manera que los archivos de cualquier ejecución puedan ser comprimidos y posteriormente eliminados del disco del servidor. Este método no devuelve ningún valor ni posee parámetro alguno.

Tabla 5. Descripción de la CC_ DataManager.

Nombre: CC_Mi_DataManager	
Tipo de clase: Controladora	
Atributo	**Tipo**

ArrayList<String>	● estudios
int	● fin
ArrayList<String>	● clientes
Para cada responsabilidad:	
Nombre:	Mi_DataManager(): void
Descripción:	Se inicializan todos los atributos de la clase y se leen los datos de los estudios a procesar.
Nombre:	generateWorkUnit(clientInfo: ClientInfo): Vector
Descripción:	Se redefine el método Se le pasa como parámetro la información de la maquina cliente y genera la unidad de trabajo.
Nombre:	processResults(ulD: long, results: Vector): boolean
Descripción:	Se redefine el método, pasándole como parámetro, el identificador de la unidad de trabajo y el atributo fin que indica la cantidad de estudios a procesar que se ejecutaran en el cliente. Cuando este atributo sea igual a cero, el método retorna true y la operación habrá terminado.
Nombre:	adjustGranularity(times: double): void
Descripción:	No se redefine el método.
Nombre:	getStatus(): string
Descripción:	Devuelve la cantidad de estudios que no se han terminado de procesar en las PCs ociosas.
Nombre:	CloseResources(): void
Descripción:	No se redefine el método.

Tabla 6. Descripción de la CC_Mi_DataManager.

Nombre: CC_Task	
Tipo de clase: Controladora	
Atributo	**Tipo**
2. File	● PROBLEMDIRECTORY
Para cada responsabilidad:	
Nombre:	Task(): void
Descripción:	Constructor por defecto

Nombre:	setProblemDir(file: File): void
Descripción:	Se le pasa como parámetro una carpeta con todas las carpetas y archivos que lo conforman y cambia el directorio del problema.
Nombre:	ClearProblemDir(): void
Descripción:	Limpia el directorio del problema.
Nombre:	recursiveDelete(f: File): void
Descripción:	Borra todos los archivos y carpetas contenidas en la carpeta pasada como parámetro, incluyéndola a ella.
Nombre:	processUnit(workUnit: Vector): Vector
Descripción:	Se le pasa como parámetros la unidad de trabajo y todos los tipos de datos de los elementos del vector son casteados a su tipo de dato original a través de los mecanismos de casteo de Java. Los resultados de las unidades de trabajo son devueltos en un java.lang.Vector y posteriormente, regresan al método processResults () en el DataManager. Cualquier error o excepción será capturada por la cláusula throws de este método y enviada al servidor, donde se registrarán en el archivo “error.log” de la ejecución. Si una unidad de trabajo provoca reiterados errores durante su procesamiento, entonces la ejecución es eliminada del sistema.

Tabla 7. Descripción de la CC_Task.

Nombre: CC_Mi_Task	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	processUnit(workUnit: Vector): Vector

Descripción:	Se le pasa como parámetros la unidad de trabajo , se capturan los datos de la unidad de trabajo, copio el estudio en la PC cliente, se manda a ejecutar el Matlab para procesar el estudio, luego se salvan los datos generados en el servidor FTP, se procede a eliminar las carpetas creadas en la PC cliente y se salva en la base de datos las direcciones físicas donde se encuentran las imágenes procesadas en el servidor FTP.
Nombre:	UploadFolder(F: FTPClient , O: string, D, string): void
Descripción:	Copia una carpeta con todos los ficheros y carpetas que lo conforman en el servidor FTP, desde la dirección origen hacia la dirección destino.
Nombre:	DownloadFolder(f: FTPClient , dir_o: string, dir_d, string): void
Descripción:	Copia una carpeta con desde la dirección origen todos los ficheros y carpetas que lo conforman desde el servidor FTP hacia la dirección destino.
Nombre:	Borrar_directorio(String dir)
Descripción:	Borra el directorio del directorio local donde se encontraba almacenado el estudio para su procesamiento.

Tabla 8. Descripción de la CC_Mi_Task.

3.5 Diseño de Base de Datos.

3.5.1 Diagrama Entidad Relación de la Base de Datos.

Los diagramas o modelos entidad-relación (a veces denominado por su siglas, E-R “Entity relationship”) son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

El Modelo Entidad-Relación es un concepto de modelado para bases de datos, propuesto por Peter Chen, mediante el cual se pretende ‘visualizar’ los objetos que pertenecen a la Base de Datos como entidades (esto es similar al modelo de Programación Orientada a Objetos) las cuales tienen unos

atributos y se vinculan mediante relaciones(Es una representación lógica de la información).

El modelado entidad-relación es una técnica para el modelado de datos utilizando diagramas entidad relación. No es la única técnica pero sí la más utilizada. Brevemente consiste en los siguientes pasos:

1. Se parte de una descripción textual del problema o sistema de información a automatizar (los requisitos).
2. Se hace una lista de los sustantivos y verbos que aparecen.
3. Los sustantivos son posibles entidades o atributos.
4. Los verbos son posibles relaciones.
5. Analizando las frases se determina la cardinalidad de las relaciones y otros detalles.
6. Se elabora el diagrama (o diagramas) entidad-relación.
7. Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama. [23].

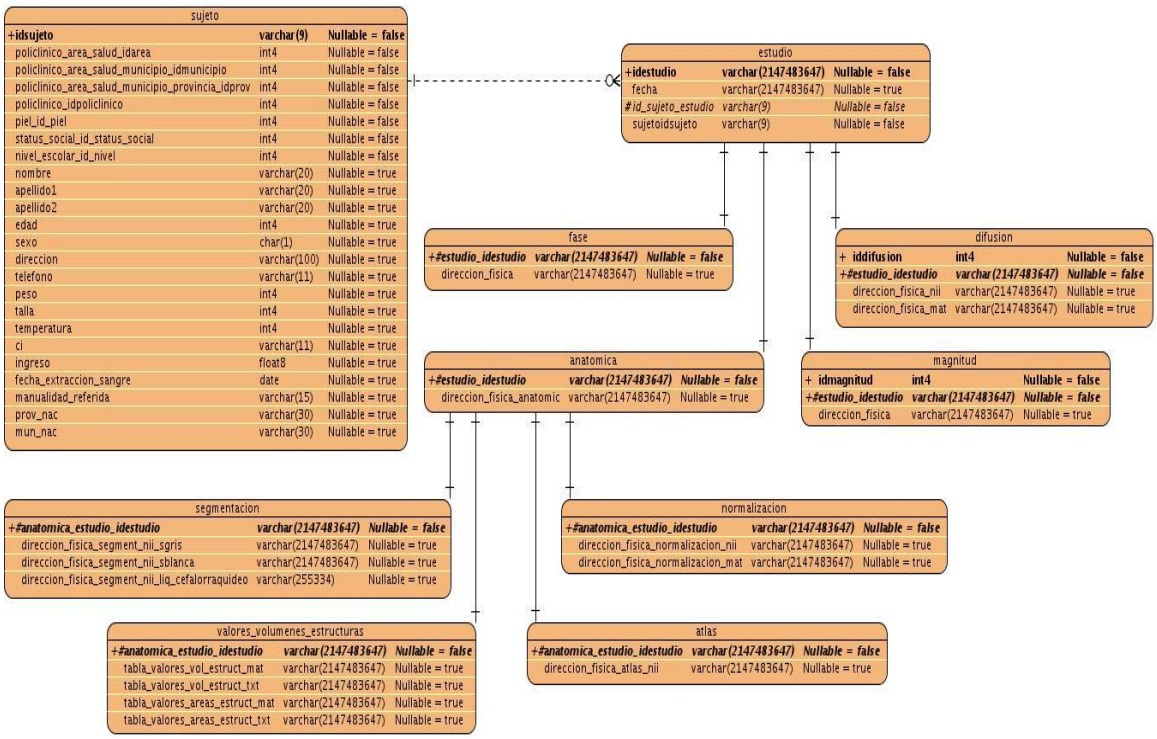


Figura 7. Diagrama Entidad-Relación.

3.5.2 Descripción de las tablas.

Nombre: Estudio		
Descripción: Almacena las direcciones físicas de las imágenes DICOMS obtenidas para cada estudio.		
Atributo	**Tipo**	**Descripción**
Fecha	character varying	Guarda la fecha en que se realizó el estudio.
Ide_estudio	character varying	Guarda el identificador del estudio realizado
Ide_sujeto_estudio	character varying	Guarda el identificador del estudio realizado conjuntamente con el identificador del sujeto.

Tabla 9. Descripción de la tabla Estudio.

Nombre: Anatómica		
Descripción: Almacena las direcciones físicas de las imágenes NIFTI de tipo anatómicas, luego de un primer procesamiento		
Atributo	**Tipo**	**Descripción**
estudio_idestudio	character varying	Guarda el identificador del estudio conjuntamente con el identificador de la imagen anatómica.
dirección_física_anatómica	character varying	Guarda la dirección física de la imagen NIFTI de tipo anatómica.

Tabla 10. Descripción de la tabla Anatómica.

Nombre: Difusión		
Descripción: : Almacena las direcciones físicas de las imágenes NIFTI de tipo difusión, luego de un primer procesamiento		
Atributo	**Tipo**	**Descripción**
iddifusion	integer	Guarda el identificador de la imagen

		NIFTI de tipo difusión.
estudio_idestudio	character varying	Guarda el identificador del estudio conjuntamente con el identificador de la imagen de difusión.
Dirección_física_nii	character varying	Guarda la dirección física de la imagen NIFTI de tipo difusión.
Dirección_física_mat	character varying	Guarda la dirección física del archivo .mat generado en el procesamiento.

Tabla 11. Descripción de la tabla Difusión.

Nombre: Fase		
Descripción: Almacena las direcciones físicas de las imágenes NIFTI de tipo fase, luego de un primer procesamiento.		
Atributo	**Tipo**	**Descripción**
estudio_idestudio	character varying	Guarda el identificador del estudio conjuntamente con el identificador de la imagen de fase.
Dirección_física	character varying	Guarda la dirección física de la imagen NIFTI de tipo fase.

Tabla 12. Descripción de la tabla Anatómica.

Nombre: Magnitud		
Descripción: Almacena las direcciones físicas de las imágenes NIFTI de tipo magnitud, luego de un primer procesamiento.		
Atributo	**Tipo**	**Descripción**
idmagnitud	integer	Guarda el identificador de la imagen NIFTI de tipo magnitud.
estudio_idestudio	character varying	Guarda el identificador del estudio conjuntamente con el identificador de la imagen de magnitud.

Dirección_física	character varying	Guarda la dirección física de la imagen NIFTI de tipo magnitud.

Tabla 13. Descripción de la tabla Magnitud.

Nombre: Atlas		
Descripción: : Almacena las direcciones físicas de las imágenes NIFTI de tipo atlas, luego de un segundo procesamiento		
Atributo	**Tipo**	**Descripción**
anatómica_estudio_ide studio	character varying	Guarda el identificador de la imagen anatómica a la que pertenece, conjuntamente con un identificador de la imagen atlas creada y el identificador del estudio.
dirección_física_atlas_ nii	character varying	Guarda la dirección física de la imagen NIFTI de tipo atlas.

Tabla 14. Descripción de la tabla Atlas.

Nombre: Normalización		
Descripción: Almacena las direcciones físicas de las imágenes NIFTI de tipo normalización, luego de un segundo procesamiento.		
Atributo	**Tipo**	**Descripción**
anatómica_estudio_ide studio	character varying	Guarda el identificador de la imagen anatómica a la que pertenece, conjuntamente con un identificador de la imagen de normalización creada y el identificador del estudio.
dirección_física_norma lización_nii	character varying	Guarda la dirección física de la imagen NIFTI de tipo normalización.
dirección_física_norma lización_mat	character varying	Guarda la dirección física del archivo .mat generado en el procesamiento.

Tabla 15. Descripción de la tabla Normalización.

Nombre: Segmentación		
Descripción: Almacena las direcciones físicas de las imágenes NIFTI de tipo segmentación luego de un segundo procesamiento.		
Atributo	**Tipo**	**Descripción**
anatómica_estudio_ide studio	character varying	Guarda el identificador de la imagen anatómica a la que pertenece, conjuntamente con un identificador de la imagen de segmentación creada y el identificador del estudio.
dirección_física_segme nt_nii_sgris	character varying	Guarda la dirección física de la imagen NIFTI de tipo segmentación para la sustancia gris.
dirección_física_segme nt_nii_sblanca	character varying	Guarda la dirección física de la imagen NIFTI de tipo segmentación para la sustancia blanca.
dirección_física_segme nt_nii_liq_cefalorraquíd eo	integer	Guarda la dirección física de la imagen NIFTI de tipo segmentación para el líquido cefalorraquídeo.

Tabla 16. Descripción de la tabla Segmentación.

Nombre: Valor Volúmenes Estructuras		
Descripción: Almacena las direcciones físicas de las imágenes NIFTI de tipo valor volúmenes estructuras luego de un segundo procesamiento.		
Atributo	**Tipo**	**Descripción**
anatómica_estudio_ide studio	character varying	Guarda el identificador de la imagen anatómica a la que pertenece, conjuntamente con un identificador de la imagen de valor volúmenes estructuras,

		creada y el identificador del estudio.
tabla_valores_vol_estruct_mat	character varying	Guarda la dirección física del archivo .mat generado en el procesamiento.
tabla_valores_vol_estruct_txt	character varying	Guarda la dirección física del archivo .txt generado en el procesamiento.
tabla_valores_áreas_estruct_mat	character varying	Guarda la dirección física del archivo .mat generado en el procesamiento.
tabla_valores_áreas_estruct_txt	character varying	Guarda la dirección física del archivo .txt generado en el procesamiento.

Tabla 17. Descripción de la tabla Valor Volúmenes Estructuras.

3.6 Definiciones del Diseño.

3.6.1 Patrón de diseño.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. En este caso se realizó una extensa búsqueda y estudio de los patrones existentes sobre estas tecnologías distribuidas. Luego de repasar los más utilizados y flexibles para este tipo de trabajo se decidió seguir los basamentos e ideas de los desarrolladores del patrón amo - esclavo (Master – Slave) o jerárquico, donde un componente amo distribuye el trabajo a los componentes auxiliares esclavos esperando entonces que retornen respuestas para formular un resultado final. Por lo que se utilizó como patrón de diseño.

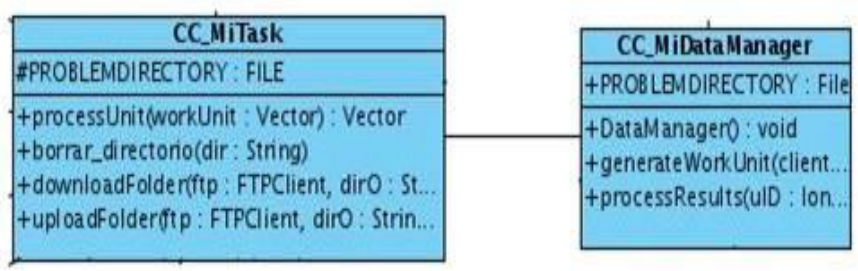


Figura 8. Patrón de diseño Amo-Esclavo.

3.6.2 Tratamiento de errores.

El tratamiento de errores en los leguajes más avanzados como por ejemplo java, c++, c#, entre otros, se basa en un nuevo mecanismo, excepciones, como la aplicación esta desarrollada en java se brinda una pequeña descripción de cómo funcionan estas excepciones (14).

- Permiten separar claramente el tratamiento de errores del código normal.
- Evitan que haya errores que pasen inadvertidos.
- Permiten agrupar en un lugar común el tratamiento de errores que ocurren en varios lugares del programa.

Las excepciones en Java son objetos de una clase extendida de otra clase especial llamada Throwable. Cuando ocurre un error:

- Se crea una instancia de la excepción, y se lanza (throw) la excepción.
- El bloque donde ocurre la excepción puede decidir tratarla (catch) o dejarla pasar.
- Si se deja pasar, el siguiente bloque puede a su vez tratarla o dejarla pasar.
- Si se trata, se ejecutan las instrucciones de un manejador.
- Si nadie la trata, el programa se interrumpe y aparece un error.

Así funciona el sistema de gestión de errores en java, y de esta forma quedaron garantizados los fallos del sistema, de este modo se trataron los posibles errores.

3.6.3 Seguridad.

La aplicación desarrollada en java y utilizando su tecnología RMI, consta de todas las políticas de seguridad establecidas y controladas por el lenguaje. Los datos que se envían a los clientes son controlados desde el servidor distribuidor de forma organizada. En caso de que un cliente se retrase o no cumpla con la unidad de trabajo que se le envíe, pasará la tarea como expirada, entonces esa misma tarea será reasignada a otro cliente que tenga las mismas características que el anterior o sea que esté listo para cumplirla además de que cumpla con los requerimientos para hacerlo. De esta forma se tiene un buen por ciento de probabilidad de que se concluya con el procesamiento sin pérdidas de unidades de trabajo. Los datos viajan de forma compacta además de serializados, lo que permite que lleguen a su destino sin problemas por los canales establecidos. Java brinda la clase SecurityManager que establece una serie de políticas de seguridad típicas del lenguaje y que se encarga de lanzar excepciones en caso de cualquier tipo de violaciones de las mismas.

3.7 Conclusiones.

En este capítulo se realizó el diseño del servicio que se quiere implementar, se mostró el diagrama de clases del diseño así como el diagrama de interacción tanto para el flujo normal como para el flujo alterno, además de hacer las descripciones de las clases del diseño y de las tablas de la base de datos. Se hizo mención también al patrón de diseño empleado durante el desarrollo de dicho servicio.

Capitulo 4: Implementación y Prueba.

4.1 Introducción.

En este capítulo se desarrolla el modelo de implementación y se analizan los diagramas de implementación y despliegue que lo conforman. En algunos de estos diagramas se detallan claramente las diferentes clases y objetos que se utilizan en la implementación, en forma de componentes, para que se tenga una idea de cómo funcionan las mismas.

En el diagrama de despliegue se muestran las relaciones físicas entre los componentes hardware y software en el sistema final, la configuración de los elementos de procesamiento en tiempo de ejecución, así como procesos y objetos que se ejecutan en ellos.

4.2 Modelo de implementación.

4.2.1 Diagrama de Despliegue.

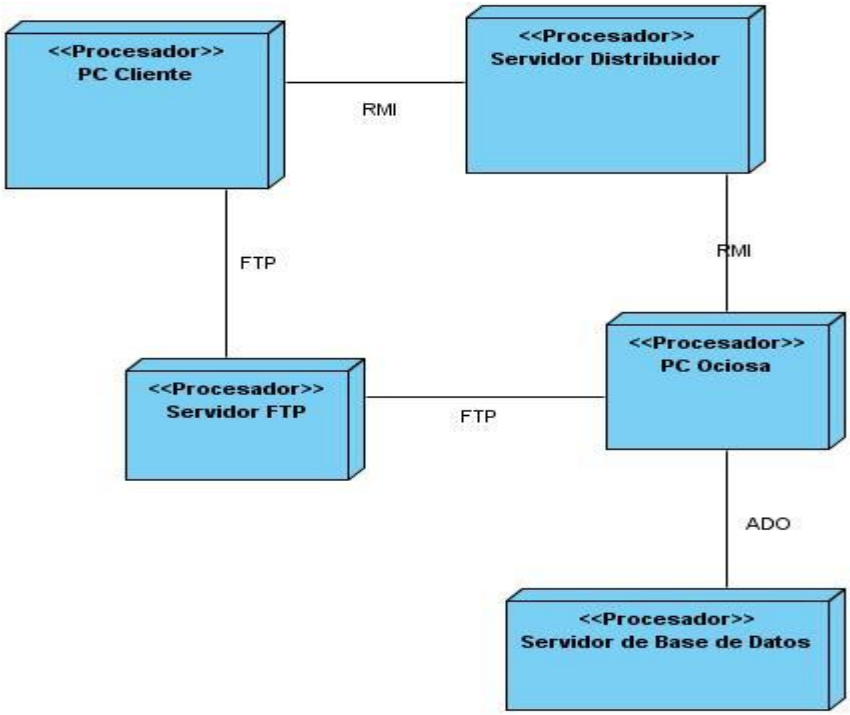


Figura 9. Diagrama de Despliegue.

4.2.2 Diagrama de Componentes.

Un diagrama de componentes representa la dependencia entre componentes software, incluyendo componentes de código fuente, componentes de códigos binarios y componentes ejecutables. Un módulo de software se puede representar como un componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias

de estas (otro).

Los diagramas de componentes de este trabajo fueron realizados por casos de uso para simplificar su comprensión.

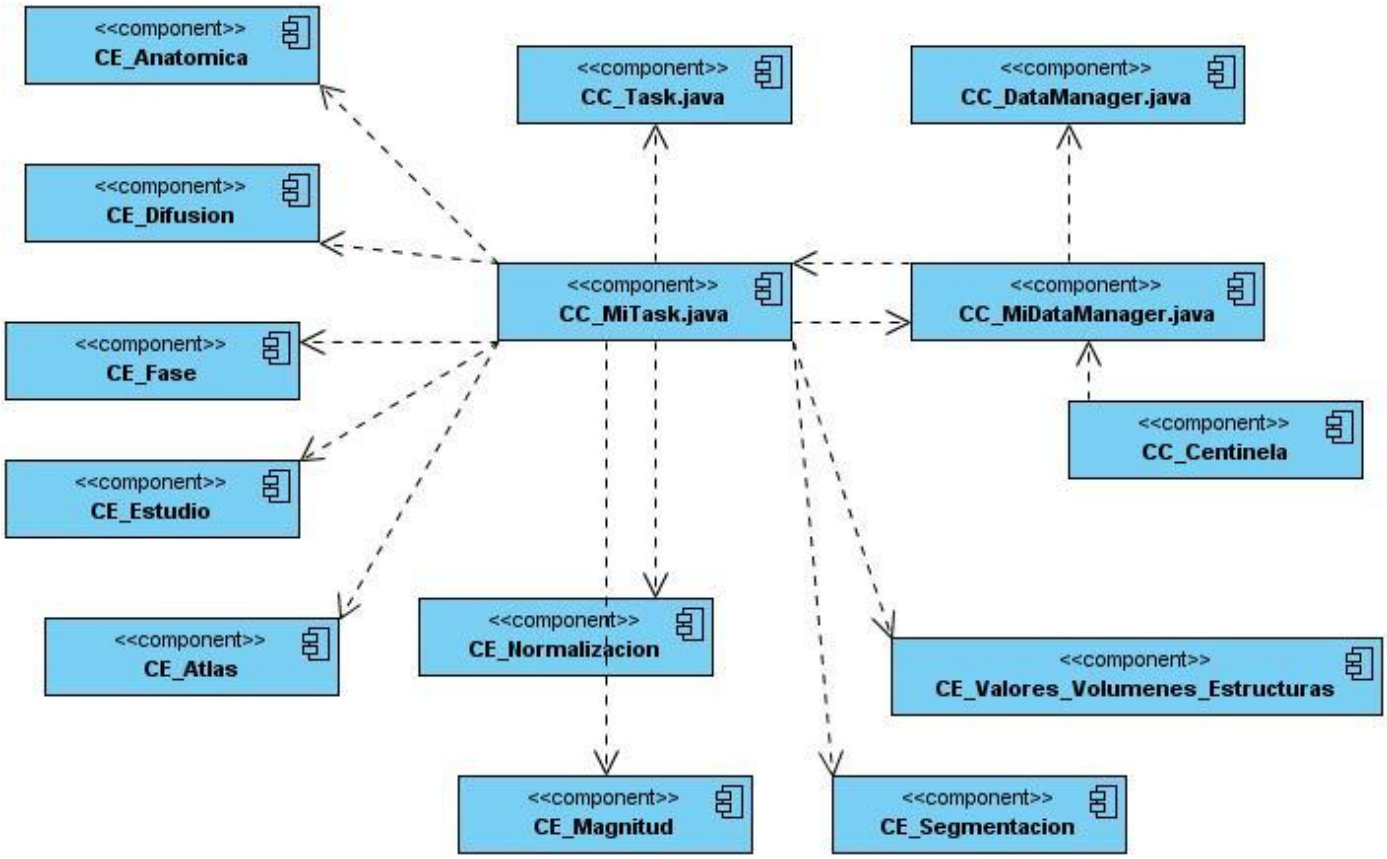


Figura 10. Diagrama de Componentes.

4.3 Prueba.

La prueba es el proceso de ejecutar un sistema bajo ciertas condiciones o requerimientos especificados con la intención de descubrir errores. Las pruebas mejoran la integridad de un sistema, al detectar las desviaciones del diseño y los errores en el sistema. Las pruebas tienen como objetivo detectar las áreas propensas a errores. Esto ayuda a la prevención de errores en el sistema, incrementa el valor del producto al adaptarlo a las necesidades del usuario.

4.3.1 Pruebas de caja negra.

Las pruebas de caja negra están enfocadas directamente al exterior del módulo, sin importar el código fuente. Son pruebas funcionales en las que se trata de encontrar fallas. En el caso del módulo presente, se llevan a cabo este tipo de pruebas para comprobar las respuestas del sistema a las diferentes entradas.

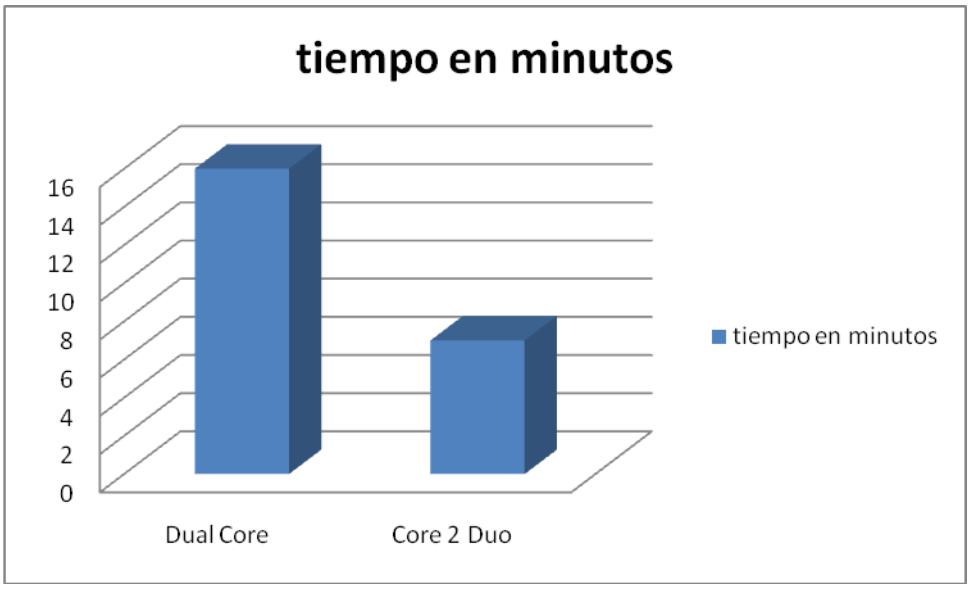
Nombre del caso de uso:	Procesar Estudio.	
Entrada	**Resultados**	**Condiciones**
Se le proporciona a la aplicación uno o varios estudios nuevos que son el contenido de entrada a este sistema.	Las imágenes quedan procesadas, llevadas al formato de entrada del software que posteriormente realiza el procesamiento.	Que existan nuevos estudios.

4.3.2 Prueba para demostrar eficiencia en el procesamiento de las imágenes de forma distribuida.

Se realizaron estas pruebas con el objetivo de demostrar que tan eficiente es el procesamiento de las imágenes de forma distribuida, para ello se analizó el tiempo de ejecución del sistema y se realizaron comparaciones con el tiempo de demora del procesamiento de los estudios como se realiza actualmente en una computadora personal. Se llegó a las siguientes conclusiones.

Estudios procesados	∑ Tiempos en Clientes	Tiempo JDS(con 2 clientes)
2 estudios	37 min	20 min
4 estudios	76 min	39 min

4.3.3 Prueba que demuestra la dependencia del tiempo con respecto al hardware en el que se desarrolle el procesamiento.



La gráfica demuestra la dependencia del tiempo con respecto al hardware en el que se despliegue el servicio, evidenciándose que si se procesa en una PC que tenga un microprocesador Core 2 Duo se reduce a más de un 50% el tiempo de procesamiento de las imágenes de resonancia magnética.
Por tanto, se puede afirmar que la distribución del procesamiento de las imágenes realizada por el presente módulo cumple su objetivo con mayor eficiencia que el proceso ejecutado en una sola computadora personal.

4.4 Conclusiones.

En este capítulo:

- Se analizó el funcionamiento del módulo analizado.
- Se presentaron los diagramas de despliegue y de componentes para una mejor visión del sistema.

- Se pusieron muestras de pruebas realizadas al software para refinar su funcionamiento y lograr los resultados esperados utilizando métodos conocidos como el de Caja Negra.

Conclusiones generales

- Se analizó, diseñó e implementó un servicio capaz de realizar el procesamiento de imágenes de resonancia magnética a través de una red para el sistema operativo Ubuntu 7.04.
- El servicio implementado accede a una base de datos de direcciones físicas de imágenes de resonancia magnética, así como a un servidor FTP donde se encuentran almacenadas las mismas.
- Se demostró la eficiencia del servicio ya que reduce en un gran porcentaje el tiempo de procesamiento de las imágenes de resonancia magnética.

RECOMENDACIONES.

- Se recomienda que se continúe actualizando este servicio con las nuevas funcionalidades que vayan surgiendo como parte del progreso de la investigación en el Centro de Neurociencias de Cuba.
- Se propone además que se perfeccione este servicio con el objetivo de lograr que sea multiplataforma.

REFERENCIAS BIBLIOGRÁFICAS.

1. **Cruz, Yodaysis Chirino**, Multimedia mapeo cerebral humano, 2008, [Disponible en: http://www.revistaciencias.com/publicaciones/EkpAVpAFupTkLakvhJ.php].

2. **Cortes, Israel**, ¿EN QUÉ CONSISTE UN MAPEO CEREBRAL?, 2008,[Disponible en: http://www.saludymedicinas.com.mx/nota.asp?id=1767]

3. **Cataluya, Universidad Politecnica de**, Conceptos Generales de Sistemas Distribuidos, 2005, [Disponible en: http://ccc.inaoep.mx/~lamorales/distribuidos/FSD-ConceptosGenerales.pdf]

4. **Martín, Ignacio.** *Propuesta para la Creación de un Programa de e-Ciencia.* 2003.

5. **JARA, OMAR HURTADO.** *NUEVOS PARADIGMAS DE LOS SISTEMAS DE INFORMACIÓN.* 2006, [Disponible en: http://www.monografias.com/trabajos16/sistemas-distribuidos/sistemas-distribuidos.shtml].

6. Introducción a OpenUP/Basic, [Disponible en: http://epf.eclipse.org/wikis/openupsp/openup_basic/customcategories/introduction_to_openup_basic,_BTJ_YMXwEduywMSzPTUUwA.html]

7. **Reserved., IBM Corp. All Rights.** Introducción a OpenUP/Basic., 2007 . [Disponible en: http://epf.eclipse.org/wikis/openupsp/openup_basic/customcategories/introduction_to_openup_basic,_BTJ_YMXwEduywMSzPTUUwA.html.]

8. Herramientas CASE[Disponible en: http://www.mitecnologico.com/Main/HerramientasCase]

9. **Hernández, José Alberto,** Visual Paradigm for UML, 2005, [Disponible en: http://www.versioncero.com/noticia/210/visual-paradigm-for-uml]

10. **International, Visual Paradigm,** Visual paradigm for UML, 2007, [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/]

11. **Marañón, Gonzalo Álvarez,** Características del lenguaje Java, [Disponible en: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html]

12. **Monreal, Enrique Gómez,** Eclipse como IDE, [Disponible en: http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf -]

13. Catálogo Final, [Disponible en: http://www.scribd.com/doc/8216421/Catalogo-Final].

14. Matlab, Poderosa Herramienta Matemática!, 2008,[Disponible en: http://www.programasde.com/matlab-76-poderosa-herramienta-matematica/]

15. **Guzmán, Christian Peralta**, Taller de Introducción a las Bases de Datos (PostgreSQL) con Software Libre GNU/LINUX, 2008, [Disponible en:

16. http://colectivoquimica.dnsalias.org/Members/admin/taller-de-introduccion-a-las-bases-de-datos-postgresql-con-software-libre-gnu-linux]

17. Ventajas de PostgreSQL, 2003, [Disponible en: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html]

18. ¿Por qué PostgreSQL? [Disponible en: http://www.efaber.net/formacion/fp/curso_acs/3_a.html]

19. DICOM, 2008, [Disponible en: http://es.wikipedia.org/wiki/DICOM]

20. PAS, Grupo de la Universidad de Deustro, Estándar y Protocolo de Imágenes Médicas, [Disponible en: www.pas.deusto.es/recursos/DICOM.pdf]

21. dcm2nii DICOm to NIFTI conversion, [Disponible en: http://www.sph.sc.edu/comd/rorden/mricron/dcm2nii.html]

22. **Valencia, María Eugenia,** Diagrama de Clases del Diseño, [Disponible en: http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/DISCLASES_A12.pdf]

23. **Vilas, Ana Fernández.** Introducción a UML, 2001. [Disponible en :http://www-gris.det.uvigo.es/~avilas/UML/UML.html]

24. Diagrama Entidad Relacion ER, [Disponible en: http://www.mitecnologico.com/Main/DiagramasEntidadRelacionER]

BIBLIOGRAFÍA

- Catálogo Final, [Disponible en: http://www.scribd.com/doc/8216421/Catalogo-Final].
- Matlab, Poderosa Herramienta Matemática!, 2008,[Disponible en: http://www.programasde.com/matlab-76-poderosa-herramienta-matematica/]
- **Cataluya, Universidad Politecnica de**, Conceptos Generales de Sisteas Distribuidos, 2005, [Disponible en: http://ccc.inaoep.mx/~lamorales/distribuidos/FSD-ConceptosGenerales.pdf]
- **Cruz, Yodaysis Chirino**, Multimedia mapeo cerebral humano, 2008, [Disponible en: http://www.revistaciencias.com/publicaciones/EkpAVpAFupTkLakvhJ.php].
- **Cortes, Israel**, ¿EN QUÉ CONSISTE UN MAPEO CEREBRAL?, 2008,[Disponible en: http://www.saludymedicinas.com.mx/nota.asp?id=1767]
- DICOM, 2008, [Disponible en: http://es.wikipedia.org/wiki/DICOM]
- **Guzmán, Christian Peralta**, Taller de Introducción a las Bases de Datos (PostgreSQL) con Software Libre GNU/LINUX, 2008, [Disponible en: http://colectivoquimica.dnsalias.org/Members/admin/taller-de-introduccion-a-las-bases-de-datos-postgresql-con-software-libre-gnu-linux]
- **Hernández, José Alberto,** Visual Paradigm for UML, 2005, [Disponible en: http://www.versioncero.com/noticia/210/visual-paradigm-for-uml]
- Herramientas CASE[Disponible en: http://www.mitecnologico.com/Main/HerramientasCase]
- **International, Visual Paradigm,** Visual paradigm for UML, 2007, [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/]
- Introducción a OpenUP/Basic, [Disponible en: http://epf.eclipse.org/wikis/openupsp/openup_basic/customcategories/introduction_to_openup_basic_BTJ_YMXwEduywMSzPTUUwA.html]
- **JARA, OMAR HURTADO.** *NUEVOS PARADIGMAS DE LOS SISTEMAS DE INFORMACIÓN.* 2006, [Disponible en: http://www.monografias.com/trabajos16/sistemas-distribuidos/sistemas-distribuidos.shtml].
- **Llanes, Clara Carmen,** Mapeo Cerebral Humano. Un reto del siglo XXI, 2005, [Disponible en: www.voltairenet.org/article128202.html]
- Mapeo Cerebral Humano: Un reto del siglo XXI. Laberintos de la mente, 2005, [Disponible en: http://www.desdeabajo.info/index.php/ediciones/128-edicion-105/364-mapeo-cerebral-humano-un-reto-del-siglo-xxi-laberintos-de-la-mente.html

- **Marañón, Gonzalo Álvarez,** Características del lenguaje Java, [Disponible en: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html]
- **Martín, Ignacio.** *Propuesta para la Creación de un Programa de e-Ciencia.* 2003.
- Más de un Java IDE: Eclipse, 2007, [Disponible en: http://www.gatzet.com/es/more-than-a-java-ide-eclipse.html/]
- **Molpeceres, Alberto.** Procesos de Desarrollo: RUP, XP y FDD, 2003, [Disponible en: http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF]
- **Monreal, Enrique Gómez,** Eclipse como IDE, [Disponible en: http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf -]
- PAS, Grupo de la Universidad de Deustro, Estándar y Protocolo de Imágenes Médicas, [Disponible en: www.pas.deusto.es/recursos/DICOM.pdf]
- ¿Por qué PostgreSQL? [Disponible en: http://www.efaber.net/formacion/fp/curso_acs/3_a.html]
- Rational Unified Process, [Disponible en: http://www-01.ibm.com/software/awdtools/rup/]
- Rojo, J. Oscar, Introducción a los Sistemas Distribuidos, 2003 [Disponible en: http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos]
- **Valencia, María Eugenia,** Diagrama de Clases del Diseno, [Disponible en: http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/DISCLASES_A12.pdf]
- Ventajas de PostgreSQL, 2003, [Disponible en: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html
- **Vilas, Ana Fernández.** Introducción a UML, 2001. [Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/UML.html]

GLOSARIO DE TÉRMINOS.

- **MCHC:** Mapeo Cerebral Humano Cubano.
- **Bioinformática:** Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.
- **MRI:** Imágenes de resonancia magnética (de sus siglas en inglés Magnetic Resonance Image).
- **CNEURO:** Centro de Neurociencias de Cuba.
- **DICOM:**(**D**igital **I**maging and **Co**mmunication in **M**edicine) es el estándar reconocido mundialmente para el intercambio de imágenes médicas, pensado para el manejo, almacenamiento, impresión y transmisión de imágenes médicas. Incluye la definición de un formato de fichero y de un protocolo de comunicación de red (TCP/IP). Los ficheros DICOM pueden intercambiarse entre dos entidades que tengan capacidad de recibir imágenes y datos de pacientes en formato DICOM. Los ficheros DICOM poseen una extensión .dcm.
- **NIFTI:** Es un formato estándar para imágenes que fue diseñado para el análisis científico de las imágenes cerebrales. El formato es simple, compacto y versátil. Las imágenes se pueden almacenar como un par de archivos (hdr / img) compatibles con la mayoría de los formatos, o simplemente como un solo archivo (NII).
- **GRID:** se refiere a una infraestructura que permite la integración y el uso colectivo de ordenadores de alto rendimiento, redes y bases de datos que son propiedad y están administrados por diferentes instituciones. Su propósito es facilitar la integración de recursos computacionales.
- **RMI:** (Java **R**emote **M**ethod **I**nvocation) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y provee de un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia, recolección de basura distribuida y paso de tipos.
- **PC_Ociosa:** Cliente del servidor distribuidor, máquina encargada de realizar los cálculos químico-teóricos. El proceso de recibir tareas cobrara fuerza en su tiempo ocio, tiempo en que la CPU este en desuso.
- **Sistemas Distribuidos:** Sistemas en que existen varias CPU conectadas entre sí, las cuales trabajan de manera conjunta.
- **Runtime:** Cuando un programa está en ejecución o corriendo.

- **JRE: (Java Runtime Environment o entorno en tiempo de ejecución Java)**, se corresponde con un conjunto de utilidades que permiten la ejecución de programas java sobre todas las plataformas soportadas.