UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 7

TÍTULO:
IMPLEMENTACIÓN DEL MÓDULO NEUROLOGÍA DEL SISTEMA DE EVALUACIÓN DEL NEURODESARROLLO EN NIÑOS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Autoras: Maylin Valdés Silva
Roselys González González

Tutores: Ing. Lucía Rodríguez García
Ing. Landy González Enríquez

La Habana, 2011
"Año 53 de la Revolución."

SÍNTESIS DE LOS TUTORES

Ing. Lucía Rodríguez García: Graduada de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2008. Posee la categoría docente de Instructor. Durante su trabajo como profesora ha impartido las asignaturas de Matemática 1 y 2, Teleinformática 1 y 2 y Práctica Profesional 1. Se ha desempeñado como jefa de la asignatura Matemática 2. Actualmente imparte la asignatura Práctica Profesional 1.

En la vinculación con la producción, pertenece al Departamento de Sistemas Especializados en Salud del Centro de Informática Médica (CESIM) y específicamente, trabaja en el desarrollo del proyecto a Ensayos Clínicos donde se desempeña como Líder de Proyecto.

Correo electrónico: lrodriguezg@uci.cu

Ing. Landy González Enríquez: Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2009. Instructor recién graduado en adiestramiento. Durante su trabajo como profesor, ha impartido Idioma Extranjero III. Actualmente se desempeña como tutor de estudiantes en la asignatura de Práctica Profesional.

En la vinculación con la producción pertenece al Departamento de Sistemas Especializados en Salud del Centro de Informática Médica (CESIM) y específicamente, trabaja en el desarrollo del proyecto a Ensayos Clínicos, donde se desempeña como desarrollador.

Correo electrónico: lenriquez@uci.cu

RESUMEN

El Módulo Neurología perteneciente al Sistema de Evaluación del Neurodesarrollo en Niños, está dirigido a informatizar el programa de Atención Temprana. Este determina la presencia o no, de afecciones del neurodesarrollo. Para ello se realizan exámenes muy extensos, donde la información se registra de forma manual, provocando la pérdida de archivos, que se cometan errores, o se genere información duplicada. Para informatizar estos procesos se desarrolla el presente trabajo, el cual tiene como objetivo: la implementación y prueba del Módulo Neurología.

Para el desarrollo de la aplicación se utilizaron las herramientas establecidas por el Sistema de Gestión Hospitalaria. Como metodología, se utilizó el Proceso Unificado de Desarrollo de Software. Como lenguaje de programación Java; PostgreSQL como sistema gestor de bases de datos y como servidor de aplicaciones JBoss. Como herramienta de mapeo Hibernate y Java Server Faces como framework de presentación. Todo ello, integrado por el entorno de desarrollo Eclipse.

Con la implantación del Módulo Neurología en los centros hospitalarios, se espera agilizar el proceso de atención al paciente, a partir de los beneficios que ofrecen las tecnologías de la información. Permitirá también, realizar la gestión de información con mayor facilidad, lo que aportará calidad y rapidez a los servicios brindados. Además, la información recogida tanto en los exámenes como en los resultados, servirá de apoyo en estudios posteriores, ya que se podrán controlar los datos del paciente antes, durante y después de la consulta, favoreciendo así el conocimiento del médico.

Palabras clave:

Atención Temprana, Neurodesarrollo, Neurología.

TABLA DE CONTENIDOS

Tabla de Contenidos

INTRODUCCIÓN

A partir de la primera mitad del siglo XX, con las transformaciones que se realizan en la informática, la electrónica y las comunicaciones, se traspasan los límites de la tecnología para penetrar en todas las esferas de la actividad humana, modificando el modo de hacer y de pensar del hombre. El futuro de la humanidad dependerá en gran medida del potencial humano, de la gestión de la producción y de los conocimientos que se alcancen. "La informática en sus diferentes manifestaciones, tiene asegurado un papel protagónico en este futuro. Cuba, en momentos en que la globalización neoliberal arrasa despiadadamente por los más diversos escenarios, se propone su utilización justa y racional sobre principios éticos sostenibles." (1)

Las Tecnologías de la Información y las Comunicaciones (TIC), han jugado un papel fundamental en la vida actual del hombre y en la forma en que este utiliza el conocimiento. Estas tecnologías se han desarrollado de manera continua y con un avance exponencial en los últimos años, debido a las necesidades, cada vez mayores, de simplificar y agilizar las tareas. Muestra de ello es sin dudas, los avances que se han alcanzado en las ciencias de la salud y la medicina en particular. En estos campos, se registra un gran crecimiento tanto en el número de usuarios, como en el de instituciones y ubicaciones, que se han incorporado a la búsqueda de diferentes medios que permitan una mejor calidad de vida.

Desde los primeros años del triunfo de la Revolución cubana, fue una estrategia política, e interés del gobierno revolucionario y el Ministerio de Salud Pública (Minsap), el estudio y procesamiento de estas nuevas tecnologías.

"La informatización del Sistema Nacional de Salud (SNS), está dada por el conjunto de métodos, técnicas, procederes y actividades gerenciales, dirigidas al manejo de la información en la salud, el cual comprende la información sobre el estado de salud de la población, la información sobre el conocimiento de las ciencias de la salud y la información en general." (2)

El desarrollo tecnológico en la medicina ha propiciado un cambio asombroso. Su avance ha permitido conocer infinidad de procesos que explican el porqué de muchas enfermedades y eventos, que ocurren en el organismo humano. Durante los últimos 20 años, un grupo de instituciones cubanas han desarrollado sistemas encaminados a lograr determinados niveles de informatización en la salud. "En su desarrollo e implementación participan diferentes empresas del Ministerio de la Informática y las Comunicaciones como: Desoft, Softel, PcMax, Sys, Universidad de las Ciencias Informáticas (UCI), Infomed, Cedisap y las

Direcciones Nacionales del Ministerio de Salud Pública, implicadas directamente en los primeros productos.” (3)

En Cuba, se trabaja intensamente con el objetivo de utilizar las tecnologías de la información y las comunicaciones para apoyar la salud pública del país. Las acciones que se han emprendido en este sentido, parten de reconocer la importancia crucial de la revolución científico-técnica que se vive, pero se han caracterizado todo el tiempo, sobre todo, por priorizar el factor humano y adecuar estos avances a los problemas reales del país. Por tal razón, cada vez es mayor el número de instituciones interesadas en aplicar estas nuevas tecnologías a la asistencia médica, la docencia, la investigación y la gestión de la información. Para ello, se realizan transformaciones educacionales y sociales, como programas de la Batalla de Ideas, a partir de los cuales, se emprendieron nuevos proyectos destinados a elevar el nivel cultural de la población y su calidad de vida.

“En estas circunstancias surge la idea de convertir el territorio que ocupaba un emplazamiento militar soviético, en la Universidad de las Ciencias Informáticas (UCI). La idea de construir esta universidad, ha sido un sueño del Comandante en Jefe Fidel Castro Ruz, que se realiza día a día en el empeño de convertirla en un centro de excelencia.” (4) Cada uno de los estudiantes que cursa sus estudios en la misma, lleva como convicción, las palabras que el Comandante en Jefe expresó en una reflexión titulada ¨Robo de cerebros¨, la cual estuvo dedicada especialmente a los graduados, y dada a conocer durante el acto por la primera graduación de la universidad, el 17 de Julio de 2007: “La tarea que los graduados de la UCI tienen por delante es grandiosa. Espero que la cumplan, y la cumplirán.”

Perteneciente a la UCI, se encuentra el Centro de Informática Médica (CESIM),el cual está trabajando en colaboración con el Hospital Pediátrico Universitario William Soler de La Habana, en donde se creó el proyecto médico Renacer Contigo. Este proyecto nace con el propósito de alcanzar mejores resultados en la atención a los infantes para su desarrollo neurológico, con lo que se espera una notable mejora en cuanto a calidad, a la hora de realizar las consultas y exámenes, así como eficiencia en los resultados obtenidos. En consecuencia, surge el proyecto: Sistema de Evaluación del Neurodesarrollo en Niños (SENDN).

El Proyecto SENDN, está dirigido a informatizar un Sistema Integrado de Gestión de Información de Salud del Programa Atención Temprana. El cual permite evaluar e intervenir en el neurodesarrollo del niño crítico al egreso hospitalario, logrando definir programas de intervención, así como, atención a la familia

que recuperará al niño con alteraciones. Todos serán evaluados por un Grupo Interdisciplinario de Atención Temprana, ya conformado, quienes determinarán la presencia o no, de afecciones del neurodesarrollo tras su período de gravedad, agrupándolos para su evaluación y seguimiento; teniendo como objetivo fundamental, el logro de una calidad de vida óptima en los afectados. Este equipo está compuesto por especialistas en: Fisiatría, Nutrición, Psicología, Neurofisiología, Logopedia, Genética y Neurología.

Los especialistas en Neurología, realizan un examen físico neurológico, exhaustivo y minucioso, de cada infante. Los exámenes que se realizan para determinar la evaluación del neurodesarrollo son muy extensos; además, la información se registra de forma manual, lo cual propicia la pérdida y deterioro de los archivos físicos, así como, que se cometan errores, o se genere información duplicada. Además, el aumento significativo de los documentos clínicos, ocasiona grandes dificultades para su procesamiento y almacenamiento. También se condicionan retrasos en la obtención de información y en la generación de datos estadísticos. Por otra parte, para realizar estas pruebas se necesita de mucho tiempo, debido a la complejidad de las mismas, lo que impide que en el mismo día se puedan atender a gran cantidad de pacientes.

Para darle solución a esta problemática, un grupo de analistas realizaron el diseño del Módulo Neurología para el sistema SENDN, donde se definieron un conjunto de requisitos funcionales que debe cumplir el sistema.

Teniendo en cuenta las contradicciones anteriormente especificadas y el diseño propuesto para el módulo, se plantea como **problema a resolver** en la presente investigación: ¿Cómo hacer funcional el diseño del Módulo Neurología del Sistema de Evaluación del Neurodesarrollo en Niños?

A partir del problema planteado, se define como **objeto de estudio** de la investigación: El proceso de desarrollo del Módulo Neurología del Sistema de Evaluación del Neurodesarrollo en Niños.

El **campo de acción** se enmarca en los procesos de implementación y prueba relacionados con el Módulo Neurología del Sistema de Evaluación del Neurodesarrollo en Niños.

Introducción

Para darle solución al problema se define como **objetivo**: Implementar el Módulo Neurología del Sistema de Evaluación del Neurodesarrollo en Niños, a partir del diseño propuesto por el analista.

Para dar cumplimiento al objetivo trazado, se proponen las siguientes **tareas de la investigación:**

1) Estudiar el Proceso de Desarrollo de Software que propone la metodología Proceso Unificado de Desarrollo de Software (RUP, por sus siglas en inglés).
2) Analizar las metodologías, plataforma, tecnologías, librerías y herramientas, definidas por el Departamento de Gestión Hospitalaria, del Centro de Informática Médica.
3) Describir la arquitectura propuesta por el Departamento de Gestión Hospitalaria, del Centro de Informática Médica.
4) Generar los artefactos correspondientes a los Flujos de Trabajo: “Implementación” y “Prueba”.
5) Implementar el sistema informático, aplicando las pautas de diseño y siguiendo lo establecido en la Especificación de Requisitos de Software.
6) Realizar las pruebas de calidad al Módulo Neurología.

Con las tareas de la investigación antes expuestas, se obtendrá como resultado el Módulo Neurología del Sistema de Evaluación del Neurodesarrollo en Niños, cumpliendo las pautas del Sistema de Gestión Hospitalaria (alasHIS).

El trabajo está estructurado en cuatro capítulos, cuyo contenido se expone a continuación.

Capítulo 1: Descripción del Proceso de Desarrollo del Módulo Neurología. En este capítulo, se realiza un estudio general del Sistema de Evaluación del Neurodesarrollo en Niños, además, de describir las características del Módulo Neurología, que se desarrolla dentro del proyecto SENDN. Se evalúa el RUP; el porqué de su utilización para desarrollar los procesos que se realizan tanto en el sistema, como en el módulo, así como el análisis de los flujos de trabajo presentes en el mismo. Por último, se fundamenta el uso de las distintas herramientas, se exponen las tendencias, técnicas, tecnologías y lenguajes de programación existentes, utilizados en el desarrollo de la aplicación.

Capítulo 2: Descripción de la arquitectura de Software del Módulo Neurología. En la elaboración de este capítulo, se describen los requisitos no funcionales y de la arquitectura. Se analizan las posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados. Se muestran las estrategias de codificación, los estándares y estilos a utilizar, así como la descripción de los requisitos definidos para la seguridad del sistema.

Capítulo 3: Descripción y análisis de la situación propuesta. Se realiza una valoración crítica del diseño propuesto por el analista. Se refinan los diagramas de clases del diseño y diagramas de interacción. Además, se describen las nuevas clases, u operaciones necesarias.

Capítulo 4: Validación de la solución propuesta. Para validar la solución propuesta, se realiza una búsqueda o diseño de las pruebas de unidades. La descripción de las pruebas se desarrolla teniendo en cuenta: objetivo, alcance, tipo y detalles del test. Además, se describen los valores utilizados para estas pruebas, así como la evaluación y los resultados obtenidos.

CAPÍTULO 1: DESCRIPCIÓN DEL PROCESO DE DESARROLLO DEL MÓDULO NEUROLOGÍA.

En el presente capítulo se describe el Sistema de Evaluación del Neurodesarrollo en Niños, además, se elabora una breve introducción a los procesos realizados por el Módulo Neurología, perteneciente a dicho sistema.

En la actualidad, no existe en Cuba un sistema informático integral de gestión, que contribuya al Sistema de Evaluación del Neurodesarrollo en Niños, de manera que pueda ser utilizado como herramienta para potenciar el Programa de Atención Temprana; permitiendo evaluar e intervenir el neurodesarrollo.

La Universidad de las Ciencias Informáticas, ha desarrollado un número considerable de sistemas informáticos para diversos organismos cubanos y ha obtenido importantes ingresos para el país a través de la comercialización de software y servicios asociados, en el exterior. La alta concentración de capital humano, de recursos materiales y los resultados alcanzados hasta la fecha, condicionan el reconocimiento de la misma, como líder en la industria de software cubana. Por tal razón, se esperan similares resultados con el desarrollo del Módulo Neurología, que pretende lograr una solución altamente parametrizable y modular, de alcance nacional, para la gestión del programa de Evaluación del Neurodesarrollo en Niños. Este programa, se desea integrar a paquetes de gestión hospitalaria, con el objetivo de obtener un software que presente mayores prestaciones a las instituciones hospitalarias.

1.1 Descripción del Sistema de Evaluación del Neurodesarrollo en Niños.

El proyecto SENDN, no tiene como punto de partida una aplicación informática, sino que está basado en el proyecto denominado Renacer Contigo del Hospital Pediátrico Universitario William Soler de La Habana. Este proyecto, fue desarrollado por el Grupo Interdisciplinario de Atención Temprana perteneciente a dicho hospital; motivados por la influencia que sobre la evolución del niño en riesgo, ejercen los programas de atención temprana, utilizados hace ya varios años a nivel mundial; además de la pobre presencia de estos programas en los servicios de atención al paciente grave en el país y la necesidad cada vez mayor, de mejorar la calidad de vida de la población infantil.

El proyecto Renacer Contigo es un programa de atención y evaluación del neurodesarrollo en niños de cero a cinco años de edad, egresados de las Unidades de Terapia Intensiva Polivalente y Neonatal. Estas Unidades de Cuidados Intensivos Pediátricas (UCIP), surgen para dar respuesta a las demandas

asistenciales en la pediatría, proporcionando al servicio del hospital, una asistencia integral, la cual, se destaca por ser profunda y continua, tanto al niño críticamente enfermo, como al niño sin problemas en el neurodesarrollo.

La atención temprana y el surgimiento del proyecto Renacer Contigo.

"El desarrollo infantil en los primeros años, se caracteriza por la progresiva adquisición de funciones tan importantes como: el control postural, la autonomía de desplazamiento, la comunicación, el lenguaje verbal y la interacción social. Esta evolución está estrechamente ligada, al proceso de maduración del sistema nervioso, ya iniciado en la vida intrauterina y a la organización emocional y mental. Requiere una estructura genética adecuada y la satisfacción de los requerimientos básicos para el ser humano a nivel biológico." (5)

La atención temprana se propone fundamentalmente, reconocer mediante el análisis, cuál es la causa principal de la disfuncionalidad del niño y no solo sus problemas, se deberá identificar un denominador común para determinar las posibles soluciones. En todos los casos, se deberá encontrar respuestas a las siguientes preguntas ¿qué no hace el niño?, ¿por qué no lo hace?, ¿qué le falta para hacerlo?. Una vez identificadas las verdaderas causas de su falta de función, se procederá a la planeación de las estrategias de tratamiento, que al final, solo estarán conducidas a un objetivo en específico, que satisfaga los problemas con los que llega el niño a la consulta. Se procederá entonces, a una preparación directa del paciente para examinarlo y al final, emitir un resultado.

Esto lo van logrando una serie de programas que existen desde hace varios años, llamados en su inicio como estimulación temprana, al solo contemplar en su objetivo a los niños. "Para llegar al actual término de atención temprana se ha pasado, a lo largo de las últimas décadas del siglo XX, por diferentes acepciones: estimulación temprana, estimulación precoz, intervención temprana o precoz." (6)

La atención brindada al niño se ha estado desarrollando de manera continua, con el objetivo de alcanzar una base sólida y contar con los suficientes criterios para desarrollar mejores técnicas en la atención y cuidados, por lo cual, el estudio realizado en los infantes durante su crecimiento ha sido una tarea crucial. "La estimulación temprana, se fundamenta en la organización neurológica, partiendo de los niveles evolutivos, que es capaz de evaluar el nivel neurológico en el que se encuentra el niño y el desfase con su edad cronológica y a partir de aquí, se plantea un programa de estimulación para la casa, con revisiones

periódicas. Se repasan los reflejos primitivos, el sistema vesicular, la importancia del gateo y el *Sistema de Estimulación Neuro-Auditiva* (SENA)." (7)

Otro aspecto fundamental, es la implicación de los padres y cuidadores, y es así, como se desarrolla en toda su amplitud, el concepto de atención temprana al contemplar no solo al niño, sino además, el ambiente donde se desarrolla. Hoy no se concibe un trabajo de atención temprana, si los padres no están implicados. Está altamente documentado y estudiado, que la estimulación tanto médica, como la que puedan proporcionarle los padres, debe estar presente en todos los momentos de la vida del niño. Se trata de interiorizar el modo de intervenir, de acercarse, de cambiar muchos de los comportamientos de la vida diaria, para transformar la estimulación en un hábito, en un modo de vida.

Todo ello trajo consigo que la atención temprana fuera clave dentro del sistema de salud. "Debido a la poca presencia de programas informáticos destinados a este tipo de tratamientos. A finales del año 2000, se propuso la realización de un sistema que fuera capaz de evaluar e intervenir, el neurodesarrollo del niño crítico al egreso hospitalario, surgiendo de esta manera el proyecto Renacer Contigo. " (8)

El Sistema de Evaluación del Neurodesarrollo en Niños.

El proyecto SENDN se propone como objetivo general el desarrollo de una aplicación informática, que permita la gestión, control y seguimiento, de los procesos que desarrolla el grupo interdisciplinario de atención temprana del Hospital Pediátrico Universitario William Soler.

Además, el proyecto debe cumplir con determinados objetivos específicos que requiere el sistema:

- Definir y validar los requerimientos de la aplicación informática a desarrollar.
- Analizar, diseñar, implementar y probar, la aplicación informática para la gestión, control y seguimiento de los procesos que desarrolla el grupo interdisciplinario de atención temprana.
- Instalar la aplicación informática a desarrollar en los servidores definidos.
- Capacitar al personal seleccionado para el uso de la aplicación informática.
- Brindar soporte al sistema que se instalará.

"El proyecto médico Renacer Contigo es un estudio prospectivo y longitudinal, cuyo universo serán los niños egresados de la UCIN y UCIP del Hospital Pediátrico Universitario William Soler, menores de cinco

años de edad.” (9) Este proyecto cuenta con cinco módulos: Fisiatría, Neurología y Neurofisiología, Psicología, Logopedia y Foniatría, Nutrición. Estos módulos estarán integrados cada uno, por un equipo interdisciplinario que evaluará a los pacientes agrupándolos para su evaluación y seguimiento en: grupo de niños con secuelas neurológicas y grupo de niños sin secuelas neurológicas. Por lo que tendrá un carácter analítico - observacional, siendo seguidos por un año, con reevaluaciones trimestrales, independientes de los resultados encontrados. La recolección de datos primarios, se hará en un expediente creado, donde las variables empleadas permitirán discutir a posteriori, las que influyeron en su estado al egreso, para evaluar el riesgo de fallecer al ingreso.

En los Hospitales Pediátricos que utilicen la solución, se vincularán a ellos, los especialistas relacionados con la evaluación del neurodesarrollo de los niños. Con la utilización de dicha solución, los especialistas comprobarán una mejora sustancial en la eficiencia y control de las actividades que actualmente desempeñan a través de la informatización de la mayoría de sus tareas, disponiendo de este modo, de más tiempo para el análisis y la planificación de las actividades que desarrollan.
A los usuarios se les proveerá una completa y moderna aplicación web, la cual será fácilmente accesible desde cualquier puesto de trabajo, evitando de este modo, las limitantes de las aplicaciones de escritorio y además, se disminuirán los ciclos de reposición de equipamiento informático por este concepto. La rotura de una estación de trabajo, no afectará en absoluto, pues con el uso de la tecnología web, podrán continuar desarrollando su labor, sin que esto implique pérdida parcial o total de su trabajo.

1.2 Descripción del Módulo Neurología.

El desarrollo del presente trabajo, está basado en la implementación y prueba del Módulo Neurología del Sistema de Evaluación del Neurodesarrollo en Niños. Por lo cual se describen y precisan, los objetivos a alcanzar con la construcción del mismo.
Dentro del sistema de salud, específicamente en la especialidad de Neurología, se ha visto una importante modificación en la práctica clínica, debido a la explosión de nueva información y nuevas tecnologías, dejando prácticamente en segundo plano, la medicina clínica tradicional. “El desarrollo de la Neurología en Cuba desde el año 1959 hasta la actualidad, se encuentra estrechamente vinculado a las transformaciones ocurridas en la salud pública con el triunfo de la Revolución y a la hostilidad del gobierno

de los Estados Unidos de América. A pesar de las limitaciones impuestas, la Neurología ha sufrido cambios radicales en el ámbito cualitativo y cuantitativo. La universalidad, gratuidad y accesibilidad del sistema de salud, han sido conquistas valiosas de la sociedad revolucionaria cubana." (10)

La Neurología es además, una de las especialidades que se atienden en el Programa de Atención Temprana. "Durante la etapa de cero a seis años en los infantes, se perfecciona la actividad de todos los órganos de los sentidos, en especial, los relacionados con la percepción visual y auditiva del niño, esto le permitirá reconocer y diferenciar colores, formas y sonidos. Por otro lado, los procesos psíquicos y las actividades que se forman en el niño durante esta etapa, constituyen habilidades que resultarán imprescindibles en su vida posterior." (11)

El cerebro evoluciona de manera sorprendente en los primeros años de vida y es el momento en el que se hace más eficaz el aprendizaje. "Para desarrollar la inteligencia, el cerebro necesita de información. Los bebés reciben información de diversos estímulos a través de los sentidos, lo hacen día y noche; si estos estímulos son escasos o de pobre calidad, el cerebro tardará en desarrollar sus capacidades o lo hará de manera inadecuada, por el contrario al recibir una estimulación oportuna el infante podrá adquirir niveles cerebrales superiores y lograr un óptimo desarrollo intelectual." (12)

Con el objetivo de alcanzar mejores resultados y facilitar el trabajo de los especialistas en Neurología y con el propósito de obtener de esta manera, notables mejoras en los procesos que se realizan de forma manual y que resultan en ocasiones sumamente engorrosos, se realiza el diseño del Módulo Neurología.

El Módulo Neurología tendrá como objetivo:

La realización de un examen físico exhaustivo y minucioso de cada paciente, en donde se procederá a realizar estudios de:

- Electroencefalograma (EEG).
- Potenciales Evocados Visuales (PEV).
- Potenciales Evocados Auditivos de Tallo Cerebral (PEATC).
- Potenciales Evocados Somato sensoriales (PESS).
- Interrogatorio al familiar.
- Examen físico neurológico completo.
- Evaluaciones trimestrales.

Como funcionalidades específicas del Módulo Neurología, se tienen:

- Realizar test. (Test neurológico).
- Generar resúmenes.
- Determinar tratamiento.
- Generar reportes.
- Gestionar la configuración.

1.2.1 Sistema categorial empleado.

Estimulación Temprana: "Conjunto de acciones dirigidas a promover las capacidades físicas, mentales y sociales del niño, a prevenir el retardo psicomotor, a curar y rehabilitar las alteraciones motoras, los déficits sensoriales, las discapacidades intelectuales, los trastornos del lenguaje y, sobre todo, a lograr la inserción de estos niños en su medio." (13)

Neurodesarrollo: Es un conjunto de mecanismos a través de los cuales, se organiza el sistema nervioso como un sistema de relación. El sistema nervioso no es pasivo, interactúa intrínsecamente, genera diferentes variables como: atención, intencionalidad, emoción, pensamiento, memoria, lenguaje, socialización y control motor, para responder a las demandas del medio.

Neurología: "La Neurología es la especialidad de la medicina que se aplica al diagnóstico y tratamiento de las enfermedades del cerebro, la médula espinal, los nervios periféricos y los músculos." (14)

Unidades de Cuidados Intensivos Pediátricos (UCIP): Asistencia eficiente a las urgencias pediátricas. Es el servicio del hospital, dedicado a la asistencia intensiva integral y continuada al niño críticamente enfermo, independientemente de cuál sea el origen de esta.

UCIP/UCIN: Unidades de Terapia Intensiva Polivalente y Neonatal.

Electroencefalograma (EEG): Examen que mide la actividad eléctrica en el cerebro humano. La máquina utilizada para el examen puede tener un computador, que registra los trazos, en un papel, para indicar la actividad cerebral en forma de ondas cerebrales. Los médicos estudian los resultados del electroencefalograma, para conocer la forma en la que está funcionando el cerebro.

Evaluación Neurológica: Es la evaluación que se realiza para detectar las enfermedades del cerebro, la médula espinal, los nervios periféricos y los músculos.

Examen Físico Neurológico: Un examen neurológico o exploración neurológica, es una evaluación del sistema nervioso de una persona. Se puede realizar con instrumentos como linternas o martillos, para los reflejos y por lo general, no resulta doloroso para el paciente. El examen neurológico se compone de varios aspectos, entre los que se incluyen la evaluación de las capacidades motoras y sensoriales, el equilibrio y la coordinación, el estado mental (el nivel de consciencia e interacción del paciente con el entorno), los reflejos y el funcionamiento de los nervios. La minuciosidad del examen depende de muchos factores, incluyendo el problema inicial que padece el paciente, su edad y las condiciones en que se encuentra.

Potenciales Evocados Visuales (PEV): Es la única prueba clínicamente objetiva, para valorar el estado funcional del sistema visual. Registra las variaciones del potencial en la corteza occipital, provocada por un estímulo sobre la retina. Por esta razón, puede evaluar las funciones retino cortical en niños, retrasados mentales y pacientes afásicas. También, puede distinguir entre pacientes con ceguera psicológica y los que la padecen por una causa orgánica.

Potenciales Evocados Auditivos de Tallo Cerebral (PEATC): Representan la actividad eléctrica generada por las vías auditivas ascendentes, en respuesta a un estímulo acústico adecuado.

Potenciales Evocados Somato Sensoriales (PESS): Los potenciales evocados somato sensoriales, estudian la transmisión de las sensaciones corporales al cerebro. Se estimulan nervios en brazos y piernas, generándose un estímulo eléctrico, que es registrado mediante electrodos, colocados en diferentes puntos de la superficie cutánea.

Requisitos Funcionales: Define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas, que muestran cómo los casos de uso serán llevados a la práctica. Son complementados por los requisitos no funcionales, que se enfocan, en cambio, en el diseño o en la implementación.

Clasificación de Zuluaga: Es la clasificación que uniforma en base al desarrollo motor, la edad madurativa neurológica, lo que permite estandarizar e introducir a los pacientes en el programa de atención temprana y establecer cambios de grupo, según la ganancia de la edad madurativa.

Sistema de Estimulación Neuro-Auditiva (SENA): Tiene como finalidad reeducar la escucha y devolverle al oído su capacidad funcional. Esto se realiza a través de terapias de reeducación auditiva que

emplean programas informáticos, que se adaptan con una mayor flexibilidad y de manera más personalizada, a las dificultades de escucha de cada paciente.

1.3 Metodología de desarrollo de software a emplear.

"El Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés), es el resultado de varios años de trabajo y uso práctico, en el que se han unificado técnicas de desarrollo, a través del Lenguaje Unificado de Modelado (UML, por sus siglas en inglés). En RUP se han dividido las actividades en grupos lógicos en los que se definen nueve flujos de trabajo principales. Los seis primeros, son conocidos como flujos de ingeniería y los tres últimos, como flujos de apoyo. " (15)

Es usual que los equipos de desarrollo, basados en las exigencias de los clientes, respecto a la rapidez con que necesitan tener el producto de software en explotación, dediquen poca atención al total entendimiento del negocio. Si se tiene en cuenta que la gran mayoría de las organizaciones no representan esquemáticamente cómo son sus procesos y que algunas de las metodologías de desarrollo de software más utilizadas, como es el caso del RUP, proponen una gran cantidad de artefactos para esta modelación cuya construcción puede volverse lenta y engorrosa, entonces, se crean todas las condiciones para que no se modele el negocio con la rigurosidad que amerita.

El resultado de esta práctica son productos de software enfocados a necesidades o requerimientos planteados por un cliente, que en ocasiones, no son capaces de determinar exactamente cómo puede un sistema de software, mejorar su línea de productos o servicios. Además, es común que se obtengan productos de software con costos de implantación extremadamente altos y alejados de la objetiva realidad de la entidad que lo pretende utilizar. Los desarrolladores tienden a ser creativos, buscando su realización profesional en la creación de sistemas informáticos ideales, a la vez que se alejan de la realidad del negocio y de los clientes.

"Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar." (16) Es así como se introduce el RUP, conformando una parte importante, en el desarrollo de un software.

1.3.1 Proceso Unificado de Desarrollo de Software.

RUP es un proceso para el desarrollo de un proyecto que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto.

"Esta metodología que guía el desarrollo del módulo, aumenta la visión de desarrollo del mismo, es decir, permite prever los cambios que un software pueda tener, de acuerdo a los requerimientos y avances sociales que se tengan, brindando objetivos más amplios y una visión de requerimientos global." (17)

Visto desde su punto más simple, el RUP, es aquel método que da cabida al cambio en las etapas del desarrollo de software, no siguiendo al pie de la letra los requerimientos, sino, por el contrario, mostrando otros campos que mejoren y optimicen el desarrollo del mismo. Se encarga de unificar todo el equipo de desarrollo de software; además, optimiza su comunicación proveyendo a cada miembro de una aproximación al desarrollo de software, con una base de conocimiento de acuerdo con las necesidades específicas del proyecto.

Dentro de sus disciplinas gestiona el control de cambios, que permite mantener al equipo trabajando en los mismos artefactos, en cualquier momento del desarrollo del producto. En su modelación, define como sus principales elementos a los trabajadores, las actividades, los artefactos y los flujos de actividades.

Los trabajadores son los propietarios de elementos o artefactos y se encargan de realizar las actividades, las cuales describen cómo una tarea es realizada por un trabajador y a su vez, manipulan los elementos. Los artefactos, constituyen los productos tangibles del proyecto que son producidos, modificados y usados por las actividades. El flujo de actividades describe cuándo las tareas son realizadas por trabajadores y produce un resultado de valor observable.

1.3.2 Características esenciales del RUP.

Dentro de las características esenciales del RUP se encuentran:

- **Dirigido por casos de uso:** Se define un caso de uso como un fragmento de funcionalidad del sistema, que proporciona al usuario un valor añadido. Los casos de uso, representan los requisitos funcionales del sistema. Estos no solo inician el proceso de desarrollo, sino, que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

- **Proceso centrado en la arquitectura:** La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema. Está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema, ayudando a determinar en qué orden se realizará. Además, la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, como son: rendimiento, reutilización y capacidad de evolución, por lo que debe ser flexible durante todo el proceso de desarrollo.
- **Proceso iterativo e incremental:** El equilibrio correcto entre los casos de uso y la arquitectura es algo muy parecido al equilibrio entre la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP, es tener un proceso iterativo e incremental, en donde el trabajo se divide en partes más pequeñas o mini proyectos.

1.3.3 Elementos principales del RUP.

La ventaja que ofrece el RUP, es su funcionalidad, porque no es usado únicamente para la elaboración de productos de software, sino, que tiene además, aplicaciones en la elaboración de otro tipo de proyectos.
En su modelación, define cuatro elementos principales:

- **Trabajadores** (¿Quién?): Definen la responsabilidad (rol), el comportamiento de un individuo y realizan las actividades.
- **Actividades** (¿Cómo?): Son tareas que tienen un propósito claro; además, son realizadas por un trabajador.
- **Artefactos** (¿Qué?): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades.
- **Flujo de actividades** (¿Cuándo?): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

1.3.4 Fases del RUP.

El ciclo de vida del RUP, es una implementación del desarrollo en espiral, ya que es creado ensamblando los elementos en secuencias semiordenadas, organizando las tareas en fases e iteraciones. Este proceso unificado, consta de ciclos que se pueden repetir a lo largo del ciclo de vida de un sistema. Un ciclo

consiste en cuatro fases: Conceptualización, Elaboración, Construcción y Transición, y el mismo concluye con una liberación.

Las fases se caracterizan por:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances, con la identificación de los casos de uso del sistema.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales), identificados de acuerdo al alcance definido.
- **Construcción:** Se obtiene un producto listo para su utilización, el cual está documentado y tiene un manual de usuario. Además, se obtiene uno o varios *release,* o versiones del producto que han pasado las pruebas.
- **Transición:** El producto ya está listo para su instalación en las condiciones reales, aunque puede implicar reparación de errores.

El resultado de cada una de estas fases dentro de un ciclo es:

Inicio:

- Se obtiene un modelo de casos de uso simplificado.
- Se identifican y priorizan los riesgos más importantes.
- Se planifica en detalle la fase de elaboración.
- Se estima el proyecto de manera aproximada.

Elaboración.

- Se obtienen los modelos de casos de uso del análisis, del diseño, de implementación y de despliegue.
- Se planifica el plan de actividades y estimación de recursos para terminar el proyecto.

Construcción.

- Se obtiene el producto con todos los casos de uso que la dirección y el cliente han acordado, para el desarrollo de la versión.

Transición.

- Se realiza la corrección de los defectos.

1.3.5 Flujos de Trabajo del RUP.

Los flujos de trabajo representan una relación de actividades que producen resultados observables. El Proceso Unificado de Desarrollo, identifica a los flujos de trabajo fundamentales, que se producen durante el proceso de desarrollo de software. Estos flujos incluyen el modelado de negocio, requerimientos, análisis, diseño, implementación y prueba. Los flujos no son secuenciales y serán realizados preferentemente durante las cuatro fases; los cuales son descritos por separado durante el proceso, pero de hecho, se ejecutan en forma concurrente, interactuando y utilizando los artefactos que cada uno genera.

"Los flujos de trabajo con los que cuenta RUP son:

- **Modelamiento del negocio:** Describe los procesos del negocio, identificando quiénes participan y cuáles son las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y Diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán, la ubicación en ellos de los componentes y la estructura en capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a lo largo del ciclo de vida del proyecto de software.
- **Instalación o despliegue:** Produce el *release* del producto y realiza actividades (empaque, instalación, asistencia a usuarios, entre otras), para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades, con las que se busca producir un producto que satisfaga las necesidades de los clientes.

- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización y actualización concurrente de elementos, control de versiones, entre otros.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización." (18)

En la construcción del Módulo Neurología se mantendrá el RUP como metodología de desarrollo, partiendo del diseño que ya se tiene definido, aunque en este caso, solo se desarrollarán los flujos de trabajo de Implementación y Prueba, debido a que el producto no se liberará, hasta que no esté integrado con los demás módulos que constituyen el Sistema de Evaluación del Neurodesarrollo en Niños.

1.4 Procesos realizados en el negocio.

Con el fin de solucionar los problemas existentes en el Hospital Pediátrico Universitario William Soler, en el proceso de evaluación del neurodesarrollo por los especialistas en Neurología, se diseñó un sistema que permite el manejo de la información, con mayor calidad y rapidez. Se tuvo como objeto de automatización inicial, las pruebas realizadas por el especialista, lo cual posibilitará la creación de los sumarios de forma automática; brindando una mayor confidencialidad, seguridad y control de la información, así como la disminución del tiempo, para aplicar dichas pruebas.
Para ello se realizó además, el levantamiento del modelo del negocio. Este modelo es una técnica para describir los procesos de la organización bajo estudio, que permite la especificación de los requisitos más importantes del sistema, determinados a través del propio negocio.

En el desarrollo del producto se cuenta con un solo proceso: realizar consulta de Neurología. Este proceso se encargará de aplicar las pruebas necesarias, para obtener la evaluación neurológica de los pacientes en la especialidad de Neurología, y tendrá como actor principal al neurólogo del hospital.

1.4.1 Requerimientos Funcionales del Software.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir; indican qué es lo que el software debe hacer, especifican cómo debe comportarse el sistema en situaciones particulares y cómo debe ser el comportamiento de entrada y salida del sistema.

A partir del modelado del negocio, se determinaron los siguientes requisitos funcionales para el módulo:

RF 1- Realizar el Examen Físico Neurológico.
RF 2- Visualizar los detalles del Examen Físico Neurológico.
RF 3- Modificar el Examen Físico Neurológico.
RF 4- Listar los pacientes atendidos en Neurología.
RF 5- Buscar los pacientes atendidos en Neurología.
RF 6- Realizar el resumen neurológico.
RF 7- Visualizar los detalles del resumen neurológico.
RF 8- Modificar el resumen neurológico.
RF 9- Buscar los resultados de la consulta.
RF 10- Visualizar los resultados de la consulta.
RF 11- Buscar el resumen neurológico.
RF 12- Actualizar el resumen neurológico.

1.5 Descripción del Flujo de Trabajo Implementación.

"En el Flujo de Trabajo Implementación, se implementan las clases y objetos en ficheros fuente, binarios y ejecutables. Además, se deben hacer las pruebas de unidad, en donde cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- Planificar qué subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.
- Cada implementador decide en qué orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.

- Se prueban los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.” (19)

La estructura de todos los elementos implementados, forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento solo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso, se pueden implementar prototipos para reducir el riesgo. Su utilidad consiste en ver, si el sistema es viable desde el principio, probar tecnologías, o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

“Como producto que se obtiene en estos procesos está: el modelo de implementación. Este consiste en una visión general de lo que tiene que ser implementado, y un apartado para cada iteración, con los componentes y subsistemas a implementar durante esa iteración, así como los resultados que se han de obtener y el testeo que se ha de realizar sobre ellos.” (20)

Dentro del flujo de trabajo implementación, se encuentran los trabajadores del mismo, que son nombrados desarrolladores. Estos desarrolladores, desempeñan labores específicas, que pueden ser:

- Arquitecto de software.
- Diseñador.
- Diseñador de interfaz de usuario.
- Diseñador de cápsulas.
- Diseñador de base de datos.
- Implementador.
- Integrador.

En este flujo también se elabora el Diagrama de Componentes, donde se incluyen los componentes y archivos que se utilizan en la implementación, para hacer disponible el sistema físicamente. Los componentes son empaquetados físicos de elementos, tales como:

- Los ficheros con código fuente de una o varias clases. Es normal que un componente implemente varios elementos o varias clases.
- Ejecutables.

- Librerías.
- Tablas de base de datos.
- Documentos.
- *Stubs.*
- *Drivers.*

Se tiene además, al Modelo de Implementación. Este modelo, no es más que un Diagrama de Componentes, que describe cómo se organizan los componentes de acuerdo a las estructuras y los mecanismos de modularización, de los cuales se dispone en el entorno y lenguaje de programación elegido, y de cómo dependen los componentes unos de otros. Es importante tener en cuenta, que esto significa realizar una tarea incremental y en cada paso se deberán incluir pocos componentes, pues en caso contrario, puede ser difícil integrar el sistema y llevar a cabo con éxito las pruebas de integración.

1.5.1 Tecnologías utilizadas en el desarrollo de la aplicación.

Para la realización del objetivo propuesto, que es la implementación del software, se requieren un conjunto de tecnologías, herramientas y lenguajes que permitirán un buen desarrollo. Estas herramientas fueron definidas previamente por el Departamento de Gestión Hospitalaria y por el de Sistema de Evaluación del Neurodesarrollo en Niños, los cuales forman parte del Sistema de Gestión Hospitalaria (alasHIS). Estas tecnologías aparecerán según su ubicación en las capas de presentación, negocio y acceso a datos.

Patrones de arquitectura y diseño.

"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después la solución a ese problema, de tal manera, que la solución pueda ser usada más de un millón de veces, sin hacerlo siquiera dos veces de la misma forma." (21)

"Una Arquitectura de Software, consiste en un conjunto de patrones y abstracciones coherentes, que proporcionan el marco de referencia necesario, guiando la construcción del software para un sistema de información." (22)

Esta arquitectura establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. Para el desarrollo de las funcionalidades, se propone la utilización del patrón de diseño Modelo-Vista-Controlador y el patrón en capas.

Modelo-Vista-Controlador.

El Modelo-Vista-Controlador (MVC), es un patrón de arquitectura de software, que separa los datos de una aplicación, la interfaz de usuario y la lógica de control, en tres componentes distintos: el modelo, las vistas y las controladoras.

Patrón en capas.

El patrón en capas es un estilo de programación, cuyo objetivo principal, es la separación y agrupamiento de los componentes del software, atendiendo a su función en el mismo, con relación al usuario del sistema, la información que maneja y las operaciones que el usuario realiza sobre la misma. Esta división, muchas veces se hace en tres capas: la capa de presentación, capa de negocio y la capa de acceso a datos.

Tecnologías horizontales.

Existen un conjunto de tecnologías horizontales que se extienden por todas las capas antes mencionadas y sirven de soporte a las tecnologías que se utilizan en cada una de ellas. Las mismas se describen a continuación.

Java Platform Enterprise Edition.

"La tecnología horizontal denominada Java Platform Enterprise Edition (JavaEE 5, por sus siglas en inglés), es una plataforma de programación. Utiliza la plataforma Java para desarrollar y ejecutar software de aplicaciones, en lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares, ejecutándose sobre un servidor de aplicaciones. Java EE es también considerada informalmente como un estándar, debido a que los

proveedores deben cumplir ciertos requisitos de conformidad, para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process.” (23)

Java EE también configura algunas especificaciones únicas para Java EE. Estas incluyen Enterprise JavaBeans, Servlets, Portlets, JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador, crear una aplicación portable entre plataformas y además, escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y la gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes, en lugar de en tareas de mantenimiento de bajo nivel.

Java Runtime Environment.

“La tecnología horizontal denominada convencionalmente Java Runtime Environment (JRE, por sus siglas en inglés), es un conjunto de utilidades, que permite la ejecución de programas Java. En su forma más simple, el entorno en tiempo de ejecución de Java, está conformado por una máquina virtual de Java (JVM, por sus siglas en inglés), un conjunto de bibliotecas Java y otros componentes necesarios, para que una aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.” (24)

La JVM es el programa que ejecuta el código Java previamente compilado, mientras que las librerías de clases estándar, son las que implementan el API de Java. Ambas, JVM y API deben ser consistentes entre sí, de ahí que sean distribuidas de modo conjunto.

Un usuario solo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje, es necesario un entorno de desarrollo, que además del JRE incluye, entre otros, un compilador para Java.

Tecnologías que pertenecen a la capa de presentación:

Java Server Faces.

Java Server Faces (JSF, por sus siglas en inglés), es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa Java Server Pages (JSP, por sus siglas en inglés), como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como *XUL.*

"Permite además crear interfaces de usuario para aplicaciones web, mediante componentes reutilizables, el manejo de estados y eventos, así como la asociación entre los datos de la interfaz y los datos de la aplicación web. Permite desarrollar rápidamente aplicaciones de negocio dinámicas, en las que toda la lógica de negocio se implementa en Java, o es llamada desde Java, creando páginas para las vistas muy sencillas." (25)

Richfaces

"RichFaces es una librería de componentes visuales para JSF, además, RichFaces posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF." (26)

Cuando RichFaces es utilizado como framework de código abierto, permite aumentar la capacidad de Ajax en las aplicaciones JSF, sin recurrir a JavaScript. Esto ocurre debido a que sus componentes están construidos con soporte Ajax y sus interfaces gráficas poseen un alto grado de personalización, facilitando así, su incorporación en las aplicaciones JSF. Además, los componentes de interfaz de usuario de Richfaces, vienen preparados para su uso fuera del paquete, así los desarrolladores pueden ahorrar tiempo.

"Richfaces, aprovecha al máximo los beneficios del framework JSF incluyendo, la validación y conversión de instalaciones, junto con la gestión de estática y dinámica de los recursos." (27)

Ajax4JSF

"Ajax4jsf es una librería de código abierto, que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas, dotándolas con tecnología Ajax de forma limpia y sin añadir código

JavaScript. Mediante este framework, podemos variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas, el control de cualquier evento de usuario, etcétera. En definitiva, Ajax4JSF permite dotar a la aplicación JSF de contenido mucho más profesional, con muy poco esfuerzo." (28)

Facelets

JavaServer Facelets es un framework para plantillas, centrado en la tecnología JSF, por lo que se integran de manera muy fácil.

Características de JavaServer Facelets:

- Facilidad en la creación de plantillas para los componentes y páginas.
- Un buen sistema de reporte de errores.
- No es necesaria la configuración XML.

"Las principales ventajas que ofrece Facelets son:

- Construcción de interfaces basadas en plantillas.
- Rápida creación de componentes por composición.
- Fácil creación de funciones y librerías de componentes. " (29)

Lenguaje Extensible de Marcado de Hipertexto.

El Lenguaje Extensible de Marcado de Hipertexto (XHTML, por sus siglas en inglés), es un lenguaje de marcado, pensado para sustituir a HTML como estándar para las páginas web. Es la versión XML de HTML con las mismas funcionalidades, pero cumple las especificaciones más estrictas de XML. Su objetivo es lograr una web semántica, donde la información y la forma de presentarla, estén claramente separadas.

"Las principales ventajas del XHTML sobre el HTML son:

- Se pueden incorporar elementos de distintos espacios de nombres XML (como MathML y Scalable Vector Graphics).

- Un navegador no necesita implementar heurísticas para detectar qué quiso poner el autor, por lo que el parser puede ser mucho más sencillo.
- Como es XML se pueden utilizar herramientas creadas para el procesamiento de documentos XML genéricos.” (30)

Tecnologías que pertenecen a la capa de negocio:

JBoss Seam

“JBoss Seam es un framework desarrollado por JBoss. Cada componente de Seam existe dentro de un contexto. El contexto conversacional por ejemplo, captura todas las acciones del usuario hasta que éste sale del sistema o cierra el navegador. ” (31)

Seam permite usar EJB3 y JSF de una forma muy sencilla, además de permitir añadir herramientas tremendamente útiles, para el desarrollo de aplicaciones web, todo en un solo framework bien acoplado y basado en estándares ampliamente utilizados y probados.

JBoss Seam tiene como principales características, que es de código abierto, que es una programación declarativa que contiene reglas de negocio accesibles, tanto para desarrolladores como administradores, y que es además, basado en web, es decir, basado en AJAX.

Tecnologías que pertenecen a la capa de acceso a datos:

Hibernate

Hibernate es una herramienta de Mapeo Objeto-Relacional (ORM, por sus siglas en inglés), para la plataforma Java, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML), o anotaciones en los *beans* de las entidades, que permiten establecer estas relaciones.

“Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. Busca solucionar el problema de la diferencia entre los dos modelos de datos, coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos), y el usado en las bases de datos (modelo relacional). Para lograr esto, permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. ” (32) Con esta información, Hibernate le permite a la aplicación, manipular los datos de la base operando sobre objetos, con todas las características de la POO. Hibernate, convertirá los datos

entre los tipos utilizados por Java y los definidos por SQL. Además, genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos, con un ligero incremento en el tiempo de ejecución.

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible. Ofrece además, un lenguaje de consulta de datos llamado Hibernate Query Language.

Enterprise JavaBeans

Enterprise JavaBeans (EJB3, por sus siglas en inglés), es una plataforma para construir aplicaciones de negocios portables, reutilizables y escalables, usando lenguaje de programación JAVA. Los Enterprise JavaBeans, forman parte del estándar de construcción de aplicaciones empresariales J2EE de Sun Microsystems. Su especificación, detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor que son, precisamente, los EJB.

"Enterprise JavaBeans ofrece múltiples beneficios, entre ellos:

- La lógica del negocio reside en los enterprise beans y no en el lado del cliente, permitiendo que el desarrollo del lado del cliente, esté desacoplado de la lógica del negocio.
- Los enterprise bean son componentes portables, reutilizables y pueden ser desplegados en servidores que usen los estándares del API JEE.
- Pueden residir en diferentes servidores y ser invocados por un cliente remoto. " (33)

Java Persistence API

"Java Persistence API (JPA, por sus siglas en inglés), es la API de persistencia desarrollada para la plataforma Java EE. Java Persistence API es un framework del lenguaje de programación Java, que maneja datos relacionales en aplicaciones, usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE). El objetivo que persigue el diseño de esta API, es no perder las ventajas de la

orientación a objetos al interactuar con una base de datos, como ocurría con EJB2, y permitir usar objetos regulares (conocidos como *POJO´s*). Java Persistence API, proporciona un modelo de persistencia basado en *POJO's*, para mapear bases de datos relacionales en Java.
En su definición, se han combinado ideas y conceptos de los principales frameworks de persistencia como son: Hibernate, Toplink y de las versiones anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA." (34)

1.5.2 Lenguajes utilizados en el desarrollo de la aplicación.

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones, las cuales pueden ser llevadas a cabo por máquinas, como es el caso de las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas, que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se describe, se prueba, se depura, se compila y se mantiene el código fuente, de un programa informático, se le llama programación.

Java

"Java es un lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros, o memoria.
Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java, también es posible." (35)

Notación para el modelado de procesos del negocio.

La Notación para el Modelado de Procesos del Negocio (BPMN, por sus siglas en inglés), es una notación gráfica común para cerrar la brecha de comunicación, que frecuentemente se presenta entre el diseño de los procesos de negocio y su procesos públicos y privados, orquestación, coreografía, etcétera; así como

conceptos avanzados de modelado, por ejemplo: manejo de excepciones y compensación de transacciones.

BPMN, está planeada para dar soporte únicamente a aquellos procesos que sean aplicables a procesos de negocio. Esto significa, que cualquier otro tipo de modelado realizado por una organización con fines distintos a los del negocio, no estará en el ámbito de BPMN. A pesar de que BPMN muestra el flujo de datos y la asociación de artefactos de datos con las actividades, no es de ningún modo un diagrama de flujo de datos.

Lenguaje Unificado de Modelado.

"Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), 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. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como: procesos de negocio y funciones del sistema. Además, presenta aspectos concretos como: expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables." (36)

Es importante resaltar que UML, es un "lenguaje de modelado" para especificar, o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas, para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología, o proceso usar.

Dentro de las ventajas que proporciona este lenguaje de modelado, se pueden destacar las siguientes:

- Está apoyado por la Object Management Group (OMG, por sus siglas en inglés), como la notación estándar para el desarrollo de proyectos informáticos.
- Es útil para el desarrollo del modelaje visual de cualquier proyecto, no solo informático.
- Promueve la reutilización.
- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyendo así modelos precisos, no ambiguos y completos.

- Puede conectarse con lenguajes de programación (ingeniería directa e inversa).
- UML es independiente del proceso, aunque para utilizarlo óptimamente, se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

1.5.3 Servidor de aplicaciones.

Un servidor de aplicaciones, es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general, a través de Internet y utilizando el protocolo HyperText Transfer Protocolo o protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés). El servidor generalmente gestiona la mayor parte de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Como principales beneficios, con esta tecnología de servidores se tiene: la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

JBoss Server

JBoss es el servidor de aplicaciones más utilizado actualmente en todo el mundo. Es un servidor de aplicaciones J2EE de código abierto, implementado en Java puro. Al estar basado en Java, puede ser utilizado en cualquier sistema operativo, para el que esté disponible Java.

"JBoss, es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de *e-business*. Combinando una arquitectura orientada a servicios revolucionaria, con una licencia de código abierto. El mismo puede ser descargado, utilizado, incrustado y distribuido, sin restricciones por la licencia. Por este motivo, es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas. " (37)

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa.

- Orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java.

1.5.4 Sistema gestor de bases de datos.

Los sistemas de gestión de bases de datos (DBMS, por sus siglas en inglés), son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de los sistemas de gestión de bases de datos, es el de manejar de manera clara, sencilla y ordenada, un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

"Existen distintos objetivos que deben cumplir los SGBD, la abstracción de la información es uno de ellos, ya que los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos, también la consistencia, para que en aquellos casos en los que no se ha logrado eliminar la redundancia, sea necesario vigilar que la información que aparece repetida, se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea." (38)

Los sistemas de gestión de bases de datos proporcionan múltiples ventajas, como son:

- Simplifican la programación de equipos de consistencia.
- Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
- Organizan los datos con un impacto mínimo en el código de los programas.
- Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado, si los mismos son bien explotados por los desarrolladores.

PostgreSQL

PostgreSQL es un potente Sistema de Gestión de Bases de Datos Objeto-Relacionales de código abierto. Tiene más de quince años de activo proceso de desarrollo a nivel mundial y una arquitectura probada, que se ha ganado una sólida reputación de fiabilidad e integridad de los datos. Es multiplataforma y funciona en los principales sistemas operativos, como: Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. También apoya el almacenamiento de grandes objetos binarios, imágenes, sonidos o vídeos.

"Características de PostgreSQL:

- Objeto-relacionales: PostgreSQL en cada tabla, define una clase que permite implementar la herencia entre tablas o clases, funciones y operadores polimórficos.
- Implementación del estándar SQL92/SQL99.
- Código abierto: Debido a la licencia liberal, PostgreSQL puede ser usado, modificado y distribuido por todo el mundo de forma gratuita para cualquier fin, ya sea de datos, comerciales o académicas.
- Múltiples-cliente API: Soporta el desarrollo de aplicaciones cliente en varios lenguajes.
- Tipos de datos: Integer, string, numeric, boolean, char, varchar, date, interval, y timestamp, tipo de datos booleanos y tipos de datos diseñados específicamente para hacer frente a las direcciones de red.
- Extensibilidad: Es una de las características más importantes de PostgreSQL ya que puede ser ampliado, se pueden añadir nuevos tipos de datos, nuevas funciones y operadores, e incluso nuevos lenguajes de procedimiento y de cliente.
- Integridad Referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos." (39)

1.5.5 Herramientas de desarrollo.

Las herramientas de desarrollo, son aquellos programas o aplicaciones, que tengan cierta importancia en el desarrollo de un programa. Pueden ser de vital importancia, como un ensamblador, un compilador o un editor, o de importancia secundaria, como un entorno de desarrollo integrado (IDE, por sus siglas en inglés). Estas son fundamentalmente editores de código, que además, pueden servir, para depurar y facilitar las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación.

Herramientas CASE

Las herramientas CASE son diversas aplicaciones informáticas, destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas, en términos de tiempo y dinero. Estas herramientas 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.

Visual Paradigm

"Visual Paradigm para UML es una herramienta UML profesional, que soporta el ciclo de vida completo del desarrollo de software. El lenguaje de modelado UML, ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE, también proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML." (40)

Con esta herramienta se puede generar código para los lenguajes PHP, JAVA y C#, y para los gestores de base de datos DB2, Informix, SQL Server, MySQL, Oracle y PostgreSQL.

Beneficios de Visual Paradigm para UML:

- Persistencia de forma fácil.
- Amplia cobertura para bases de datos.
- Base de datos de ingeniería inversa.
- VP-UML no solo es una aplicación independiente, se puede integrar a los principales Integrated Development Environments (IDEs): Eclipse/WebSphere ®, Borland.

1.5.6 Ambiente de desarrollo integrado.

Eclipse

"Eclipse es un entorno de desarrollo integrado (IDE, por sus siglas en inglés), que facilita las tareas de edición, compilación y ejecución de programas durante su fase de desarrollo. Aunque Eclipse, pretende

ser un entorno versátil soportando varios lenguajes de programación, es con el lenguaje Java con el que mejor se integra y con el que ha ganado su popularidad. ” (41)

Es además, una aplicación gratuita y de código abierto, disponible en la red para su descarga, e incluida ya en muchas distribuciones de Linux. Proporciona el entorno de desarrollo solamente, siendo necesario además, para el caso del lenguaje Java, disponer del Java Development Kit (JDK, por sus siglas en inglés), para poder compilar y ejecutar las aplicaciones desarrolladas.

1.6 Descripción del Flujo de Trabajo denominado Prueba.

El Flujo de Trabajo Prueba es una actividad en la cual, un sistema o un componente, es ejecutado bajo unas condiciones o requerimientos especificados. Los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

La prueba de software, es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

Los roles del Flujo de Trabajo Prueba son:

- Administrador de Prueba: Es el responsable del éxito de la prueba. Este rol involucra al defensor de prueba y calidad, planificación y administración de recursos y la resolución de problemas que impiden las pruebas.
- Analista de Prueba: Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de la prueba y el resultado en cada ciclo de prueba, evaluando la calidad total experimentada, como un resultado de las actividades de prueba. Este rol tiene la responsabilidad, para representar apropiadamente las necesidades de los stakeholder o usuarios, que no tienen representación regular y directa en el proyecto.
- Diseñador de prueba: Es el responsable de definir el método de prueba y asegurar su implementación exitosa.
- Probador: Es el responsable durante las actividades principales de las pruebas, el cual incluye la conducción de las pruebas necesarias y el registro de los resultados.

1.6.1 Métodos utilizados en el Flujo de Trabajo Prueba.

Método de Caja Blanca.

"En las pruebas de caja blanca se comprueban los caminos lógicos del software proponiendo casos de prueba, además, estas permiten que se ejerciten conjuntos específicos de condiciones y bucles. Mediante estas pruebas se puede examinar el estado del programa en varios puntos, para determinar si el mismo es real y si coincide con el resultado esperado." (42)

Las pruebas de caja blanca, requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o del código.

Mediante los métodos de prueba de caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

- Se ejerciten todas las decisiones lógicas, en sus vertientes verdaderas y falsas.
- Se ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos, para asegurar su validez.

Método de Caja Negra.

"Las pruebas de caja negra se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantenga." (43)

Las pruebas de caja negra permiten:

- Verificar las especificaciones funcionales, además, estas no consideran la estructura interna del programa.
- Se realizan sin el conocimiento interno del producto.
- No validan funciones ocultas, por ejemplo: funciones implementadas pero no descritas en las especificaciones funcionales del diseño, por tanto, los errores asociados a ellas no serán encontrados.

Las pruebas normalmente se efectúan con los llamados datos de prueba, que es un conjunto seleccionado de datos típicos, a los que puede verse sometido el sistema, los módulos o los bloques de

código. También se escogen: datos que llevan a condiciones límites al software, a fin de probar su tolerancia y robustez, datos de utilidad para mediciones de rendimiento, datos que propician condiciones eventuales o particulares poco comunes y a las que el software normalmente no estará sometido. Los datos de prueba, no necesariamente son ficticios o creados, pero normalmente, si lo son los de poca probabilidad de ocurrencia.

Existen además, otros métodos, los cuales permiten realizarles pruebas al software, tales como:

- **La prueba del Camino Básico:** Esta prueba permite al diseñador de casos de prueba, obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida, como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico, garantizan que durante la prueba se ejecute por lo menos una vez, cada sentencia del programa.
- **La prueba de Condición:** Esta prueba es un método de diseño de casos de prueba, que ejercita las condiciones lógicas, contenidas en el módulo de un programa.
- **La prueba de Flujo de Datos:** Se seleccionan caminos de prueba de un programa, de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- **La prueba de Bucles:** Es una técnica de prueba de caja blanca, que se centra exclusivamente en la validez de las construcciones de bucles.

En el presente capítulo se realizó una descripción del proyecto médico Renacer Contigo, así como del sistema que se encarga de lograr la informatización del mismo: SENDN. Se mostraron además, las características con las que cuenta dicho sistema, así como los módulos por los cuales está compuesto. Se definieron los principales objetivos del Módulo Neurología y se realizó una breve reseña de la importancia que tiene la informatización de los procesos realizados en la Neurología. Para un mejor entendimiento de usuarios inexpertos, se plasmó un sistema categorial, el cual explica términos o conceptos, tanto médicos, como técnicos.

Se realizó un análisis del RUP arribándose a la conclusión, que la utilización del mismo como metodología de desarrollo de software en la construcción del Módulo Neurología, permitirá que las funcionalidades con

las que cuenta la aplicación sean operativas, cumpliendo además, con los requerimientos de calidad establecidos para el sistema; todo ello mediante un proceso continuo de pruebas y retroalimentación, gracias a que es una metodología completa y bien documentada, lo que representa una alta probabilidad de éxito en la obtención del módulo. Una de las cualidades más destacables, es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así, los costos de implantación en un equipo de desarrollo; por lo cual, para el desarrollo de la aplicación se decidió la utilización de esta metodología.

En el capítulo se describieron además, las actividades, características y trabajadores, que se involucran en los flujos de implementación y prueba. Se describieron y analizaron las técnicas y plataformas de desarrollo, que serán usadas en la construcción de la aplicación, así como las herramientas y tecnologías definidas por el Departamento de Gestión Hospitalaria.

CAPÍTULO 2: DESCRIPCIÓN DE LA ARQUITECTURA DE SOFTWARE DEL SISTEMA DE EVALUACION DEL NEURODESARROLLO EN NIÑOS.

Un elemento crítico y muy importante, que se debe considerar en una arquitectura de software y en lo que precisamente está basado el diseño, son los requerimientos no funcionales del sistema, específicamente, los atributos de calidad, establecidos para el mismo; atributos como: usabilidad, fiabilidad, eficiencia, soporte, restricciones de diseño, requerimientos de rendimiento, hardware y software, etcétera.

De manera concreta, al diseñar una arquitectura de software, se deben crear y representar componentes que interactúen entre ellos y tengan asignadas tareas específicas, además, de organizarlos de forma tal, que se logren los requerimientos establecidos.

En el diseño de la arquitectura de software, es donde recae toda la creatividad, experiencia y creación, de la propuesta de solución que más se adecue a las necesidades de los clientes. Por tal razón, en este capítulo se pretende analizar los requerimientos no funcionales del software, los patrones de arquitectura y diseño establecidos, la especificación de los términos de seguridad, las estrategias de codificación, los estándares y estilos a utilizar en la construcción del software; y además, se especifican los términos de seguridad con los que contará la aplicación.

2.1 Requerimientos No Funcionales del Software.

Los requerimientos no funcionales de un software, son propiedades o cualidades que el producto debe poseer. Estas muestran las características que hacen al producto atractivo, usable, rápido y confiable. A diferencia de los requisitos funcionales, estos no modifican o alteran la funcionalidad del producto. Los requisitos no funcionales son especificados por escrito y deben posibilitar su verificación y prueba, además, sus especificaciones deben ser lo más abstractas y concisas posibles, siendo descritos como una característica del sistema. Los requerimientos no funcionales surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware, o a factores externos como: los reglamentos de seguridad, las políticas de privacidad, entre otras.

Estos requisitos que se muestran a continuación, fueron definidos por el Departamento de Gestión Hospitalaria (alasHIS), para ser adoptados por SENDN, en la implementación de todos los módulos que lo conforman.

RNF Usabilidad.

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

- Para alcanzar el nivel elemental, asociado al dominio del sistema y el uso eficiente del mismo, serán necesarios veinte días de preparación, obteniendo la categoría de Usuarios Normales.
- Para alcanzar el nivel avanzado, asociado al dominio del sistema y el uso eficiente del mismo, serán necesarios treinta días de preparación, obteniendo la categoría de Usuarios Avanzados.

RNF Seguridad.

El requisito no funcional de seguridad es uno de los más difíciles, ya que provocará los mayores riesgos si no se maneja correctamente. La seguridad de un sistema abarca el ambiente en el que se usará el mismo. Por lo que se tiene que contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso al sistema y las regulaciones legales que afectan o determinan el uso y que serán tenidas en cuenta si se incumple.

La seguridad puede ser tratada en tres aspectos diferentes: Confidencialidad, Integridad y Disponibilidad; por lo que se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos solo a los niveles establecidos, de acuerdo con la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario, o por el administrador del sistema. Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento. El sistema proporcionará un registro de actividades de cada usuario, además, permitirá la recuperación de la información de la base de datos, a partir de los respaldos o salvas realizadas.

RNF Fiabilidad.

En los servidores de los hospitales y en el Centro de Datos Nacional del Minsap, se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos. Se garantizarán además, políticas de respaldo a toda la información, evitando pérdidas en caso de desastres ajenos al sistema.

Los estudios y otros datos que por su tamaño no se puedan replicar hacia el Centro de Datos, se almacenarán localmente en los hospitales. Las informaciones médicas relacionadas con los pacientes y

que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas para evitar accesos, o modificaciones no autorizadas.

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

El sistema implementará un control de cambios a determinados campos de información (seleccionados por su importancia), de forma tal, que sea posible determinar cuáles han sido las actualizaciones que se le han realizado.

RNF Eficiencia.

El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos. Además, el sistema minimizará el volumen de datos en las peticiones y optimizará el uso de recursos críticos, como es el caso de la memoria. Para ello, se potenciará como regla, guardar en la memoria caché, datos y recursos de alta demanda. El sistema respetará buenas prácticas de programación, para incrementar el rendimiento en operaciones costosas para la máquina virtual.

RNF Soporte.

- **Monitoreo de funcionamiento:** Permitirá la administración remota, el monitoreo del funcionamiento del sistema en los centros hospitalarios y la detección de fallas de comunicación.
- **Respaldo y recuperación de base de datos:** Se podrán realizar copias de seguridad de la base de datos, hacia otro dispositivo de almacenamiento externo, además, recuperará la base de datos, a partir de los respaldos realizados.
- **Auditoría:** Se mantendrá un chequeo de las operaciones y acceso de los usuarios al sistema; para esto, debe existir un registro de trazas que almacene todas las transacciones realizadas en el sistema, indicando para cada caso como mínimo: usuario que realizó la transacción, tipo de operación que se realizó, fecha y hora en que se realizó e información contenida en el registro modificado.

- **Configuración de parámetros:** Permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

RNF Réplica.

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

RNF Restricciones de diseño.

El sistema estará dividido en las siguientes capas:

Capas físicas:

- Cliente: Computadora con cualquier tecnología o sistema operativo, que cuente con un navegador actualizado y que siga los estándares web (se recomienda IE 7 o superior o Firefox 3.x).
- Servidor de Aplicaciones: Servidor con cualquier tecnología o sistema operativo, que soporte el Java Runtime Environment (JRE) y al JBoss AS 4.2. Estas mismas condiciones se aplican para los servidores de aplicación del Centro de Datos.
- Servidor de Base de Datos: Servidor con cualquier tecnología o sistema operativo, que soporte a PostgreSQL Server 8.2 en los servidores de base de datos de cada hospital, y para los servidores de base de datos del Centro de Datos.

Capas lógicas:

- Presentación: Contiene todas las vistas y la lógica de la presentación. El flujo web se maneja de forma declarativa y basándose en definiciones de procesos del negocio.
- Negocio: Mantiene el estado de las conversaciones y procesos del negocio, que concurrentemente pueden estar siendo ejecutados por cada usuario. En los casos en que algún objeto del negocio tenga una interfaz externa, siendo accesible la misma desde sistemas legados, o directamente del cliente, se garantiza la seguridad a nivel de objeto y métodos.
- Acceso a Datos: Contiene las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente, en la implementación del motor de persistencia Hibernate.

RNF Requisitos para la documentación de usuarios en línea y ayuda del sistema.

Se posibilitará el uso de ayudas dinámicas y tutoriales en línea, sobre el funcionamiento del sistema.

Interfaces de usuario:

- Las ventanas del sistema contendrán claros y bien estructurados los datos, además, permitirán la interpretación correcta de la información.
- La entrada de datos incorrecta será detectada claramente, e informada al usuario.
- Todos los textos y mensajes en pantalla aparecerán en idioma español.
- Se incorporarán asistentes que faciliten el uso del sistema por los usuarios, en procesos con determinado nivel de complejidad, para minimizar la posibilidad de errores.
- El diseño de la interfaz del sistema responderá a la ejecución de acciones, de una manera rápida, minimizando los pasos a dar en cada proceso.

RNF Requerimientos de rendimiento.

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

RNF Requerimientos de hardware.

- **Estaciones de trabajo:** En la solución se incluyen estaciones de trabajo, para las consultas del Sistema de Información Hospitalaria (alasHIS), las cuales necesitan de una capacidad, que soporte un sistema operativo con un navegador actualizado y que siga los estándares web; para ellos se recomienda: IE 7, Firefox 2, o versiones superiores. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz, con sistema operativo Linux.
- **Servidores:** La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad, residencia de la información y aplicaciones bajo esquemas seguros y confiables. Los servidores de bases de datos de datos que soportará son: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual – Core, 4GB de memoria,

2x72GB de disco y sistema operativo Linux. Como servidores de aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual – Core, 4GB de memoria, 2x72GB de disco y sistema operativo Linux. Y los servidores de intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual – Core, 2 GB de memoria, 2x72GB de disco y sistema operativo Linux.

RNF Requerimientos de software.

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

2.2 Descripción de la arquitectura, fundamentación.

2.2.1 Patrones de arquitectura y diseño.

"Los patrones de diseño proporcionan un esquema, para refinar los subsistemas o componentes de un sistema software y las relaciones entre ellos. Describe estructuras recurrentes, para comunicar los componentes que resuelven un problema de diseño, en un contexto particular. Son patrones de un nivel de abstracción menor, que los patrones de arquitectura. Están por lo tanto, más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema." (44)

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón, debe poseer ciertas características. Una de ellas, es que debe haber comprobado su efectividad resolviendo problemas similares, en ocasiones anteriores.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones, a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.

- Facilitar el aprendizaje de las nuevas generaciones de diseñadores, condensando conocimiento ya existente.

"Los patrones arquitectónicos, o patrones de arquitectura, son patrones de software que ofrecen soluciones a problemas de arquitectura de software. Dan una descripción de los elementos y del tipo de relación que tienen, junto con un conjunto de restricciones sobre cómo pueden ser usados. Expresan un paradigma fundamental para estructurar u organizar un sistema. Proporcionan un conjunto de subsistemas o módulos predefinidos, con reglas y guías para organizar las relaciones entre ellos." (45)

Para el desarrollo de las funcionalidades, se propone la utilización del patrón de diseño Modelo-Vista-Controlador y el Patrón en capas.

Modelo-Vista-Controlador

El patrón Modelo-Vista-Controlador (MVC), separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código es el que provee los datos dinámicos a la página. A continuación se detallan cada una de las capas, que conforman al MVC.

- Modelo: En esta capa se realiza una representación específica de la información, con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador, facilitando de esta manera las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.
- Vista: En esta capa se presenta el modelo en un formato adecuado para interactuar, lo cual sería la interfaz de usuario.
- Controlador: Esta capa responde a eventos, usualmente acciones del usuario, e invoca peticiones a la modelo y a la vista.

El uso de este patrón es favorable, ya que al separar la presentación de la programación (o lógica de negocio), la aplicación es más fácil de modificar en el futuro, el resultado es más claro, y el reparto de tareas dentro del equipo de trabajo, es mucho más fácil; la depuración de la aplicación es más sencilla y puede utilizarse un marco de trabajo (o *framework*) bien testeado.

Para el desarrollo de la aplicación, se seleccionó el uso del patrón, ya que muestra las siguientes ventajas:

- La aplicación está implementada modularmente.
- Sus vistas muestran información actualizada todo el tiempo.
- El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
- Si se desea hacer una modificación al modelo del dominio, como aumentar métodos, solo debe modificarse el modelo y las interfaces con las vistas.
- Las modificaciones a las vistas no afectan en absoluto a los otros módulos de la aplicación.
- El patrón MVC está demostrando ser un patrón de diseño bien elaborado, pues las aplicaciones que lo implementan presentan una extensibilidad y un mantenimiento único, en comparación con otras aplicaciones basadas en otros patrones.

Patrón en capas.

El patrón de arquitectura por capas, es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software, ubicados de la misma forma que las capas de un pastel, donde cada capa, descansa sobre la inferior. En este esquema, la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior.

"Los beneficios de trabajar un sistema por capas son:

- Se puede entender una capa como un todo, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- Se minimizan dependencias entre capas.
- Las capas posibilitan la estandarización de servicios.
- Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel." (46)

Esta división muchas veces se hace en tres capas: la capa de presentación, capa de negocio y la capa de datos.

Capa de Presentación: Referente a la interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando, o tan complejo, como una aplicación basada en formas. Su principal responsabilidad es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados.

Capa de las Reglas de Negocio: Esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos, basados en la información dada por el usuario, datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos. Se puede diseñar la lógica de la capa de negocio para uso directo, por parte de componentes de presentación o su encapsulamiento.

Capa de Datos: Esta capa contiene la lógica de comunicación con otros sistemas, que llevan a cabo tareas por la aplicación. Estos pueden ser monitores transaccionales, otras aplicaciones, sistemas de mensajerías, etcétera. Para el caso de aplicaciones empresariales, generalmente está representado por una base de datos, que es responsable del almacenamiento persistente de la información. Esta capa debe abstraer completamente a las capas superiores del dialecto utilizado, para comunicarse con los repositorios de datos.

2.3 Estrategias de integración.

Las estrategias de integración o reutilización, se apoyan en el uso de componentes de software, vistos como colecciones de código reutilizables, que facilitan el desarrollo de las aplicaciones, los cuales, se pueden encontrar en el mismo sistema que se está desarrollando, o en sistemas externos, que incluyan la solución que se desea desarrollar. Estas estrategias, reducen los costes de diseño, codificación y comprobación.

El Módulo Neurología, implementa funcionalidades para la aplicación del test, el cual requiere de la integración para la adquisición de componentes del Módulo Expediente y Turnera.

Esta integración del Módulo Neurología con Expediente y Turnera, ocurre a través de los listados de los pacientes, para poder desarrollar algunas de las funcionalidades que se implementan en Neurología, entre las que se encuentran:

- Datos personales del paciente.
- Listado de pacientes a atender.

Para obtener los datos personales del paciente y la lista de pacientes a atender, se hace necesaria la conexión con el Sistema de Gestión Hospitalaria (alasHIS). El HIS cuenta con dos esquemas: el HC_local y el SENDN. En el esquema HC_local, que gestiona entre otros aspectos, los datos personales del paciente en su tabla Hoja Frontal, se relaciona a través de su identificador con la tabla mode_Expediente, perteneciente al esquema SENDN. Se necesita además, implementar en los controladores del sistema, la interacción del Módulo Neurología y el Módulo Expediente; para ello el esquema SENDN cuenta con una tabla llamada modneu_evaluación, donde todos los datos son manipulados desde el componente mode_Expediente, el cual se relaciona a través de su identificador con dicha tabla del esquema SENDN, perteneciente al Módulo Neurología. De esta forma, se realiza la integración entre los módulos, permitiendo la reutilización de componentes y así, aumentar la calidad de las funcionalidades del sistema.

2.4 Especificación de los términos de seguridad.

La seguridad informática se enfoca en la protección de la infraestructura computacional y todo lo relacionado con esta, incluyendo la información contenida. Para ello, existen una serie de estándares, protocolos, métodos, reglas, herramientas y leyes, concebidas para minimizar los posibles riesgos a la infraestructura o a la información. La seguridad informática comprende software, bases de datos, metadatos, archivos y todo lo que la organización valore y signifique un riesgo, si ésta llega a manos de otras personas. Este tipo de información, se conoce como información privilegiada o confidencial.

La seguridad de un sistema es de vital importancia, ya que de ello depende la integridad y la confiabilidad de la información, además, un sistema sin seguridad, es un sistema vulnerable a todo tipo de ataques.

En el caso del sistema en cuestión, los métodos y técnicas de seguridad fueron definidos por el proyecto alasHIS. Para esto definieron diferentes tipos de seguridad:

- Acceso al sistema.
- Registro de trazas.
- Administración de seguridad.
- Configuración de funcionalidades.

Las cuales presentan las siguientes características:

Acceso al sistema

Cuando el usuario necesita acceder al sistema, este solicita el nombre del usuario y la contraseña. El usuario introduce los datos solicitados, el sistema verifica que los datos introducidos sean válidos, si es así, el usuario accede al módulo. El sistema muestra como opciones del menú, las funcionalidades a las que tiene permiso para acceder el usuario en el módulo, lo que garantiza el acceso de los mismos, solo a los niveles establecidos, de acuerdo con la función que realizan. El sistema permite: cerrar sesión y salir del módulo.

Registrar trazas

Cuando el usuario realiza una acción sobre el sistema, la cual puede ser: inicio o cierre de sesión, acceso al módulo, modificación a un atributo de una entidad, o cualquier otra operación sobre el sistema, este registra una traza en la base de datos.

Administrar seguridad

El sistema brinda la posibilidad de asignar o denegar permiso a roles y usuarios.

Configurar funcionalidades

El sistema brinda la posibilidad de configurar las funcionalidades del módulo.

2.5 Vista de despliegue.

Parte importante del Flujo de Trabajo Implementación, es el diagrama de despliegue. Este diagrama es del tipo UML, que se emplea para modelar el hardware utilizado, en las implementaciones de sistemas y las relaciones entre sus componentes.

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución, en instancias de nodos conectados por enlaces de comunicación. Un nodo, es un recurso de ejecución tal como un computador, un dispositivo o una memoria.

A continuación se muestra el diagrama de despliegue.

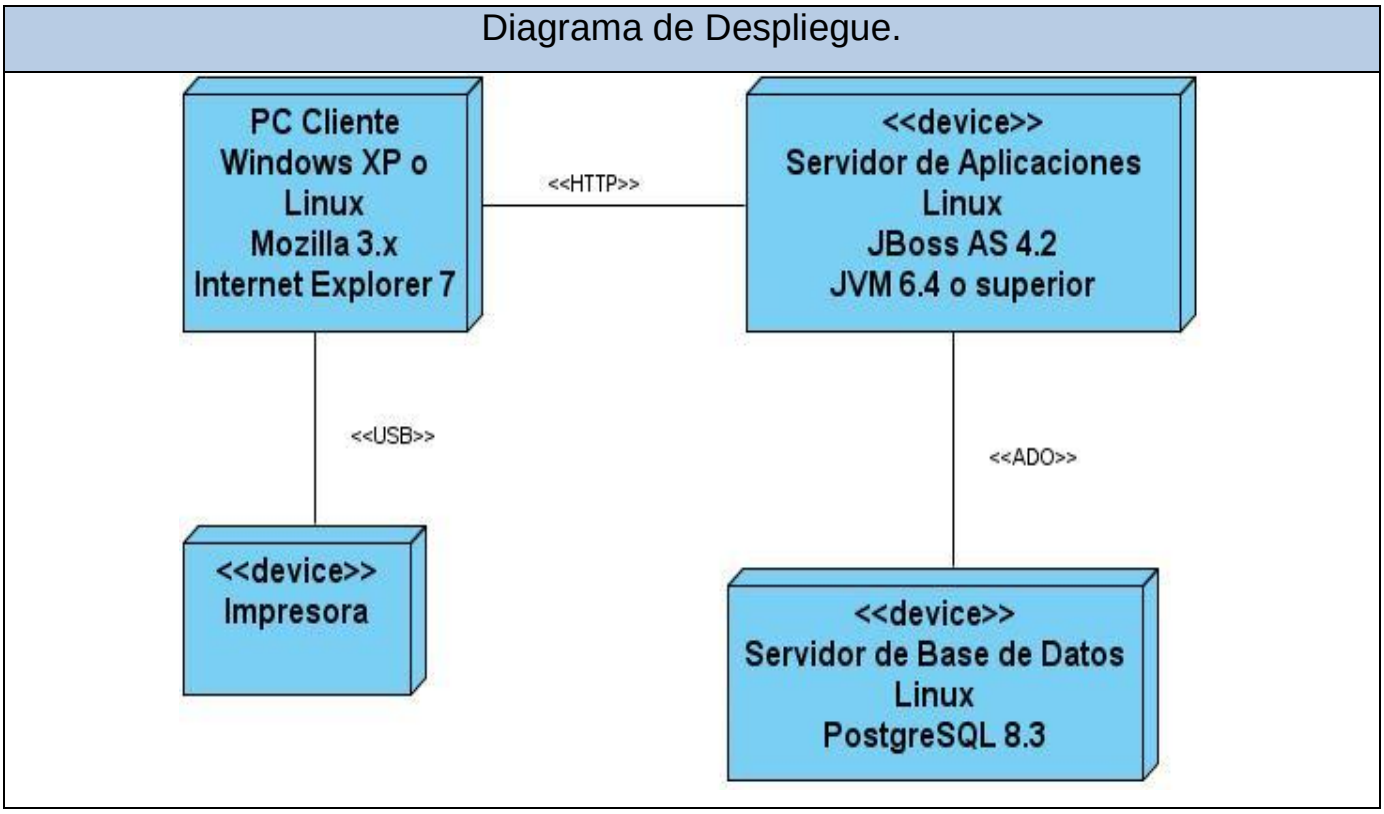

Figura 1: Diagrama de Despliegue.

"Los estereotipos del diagrama de componentes permiten precisar la naturaleza del equipo, estos estereotipos son:

- Dispositivos.
- Procesadores.
- Memoria." (47)

2.5 Estrategias de codificación. Estándares y estilos a utilizar.

Un estándar de codificación comprende todos los aspectos de la generación de código, de tal manera, que sea prudente, práctico y entendible, para todos los programadores. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación, es de gran importancia para la calidad del software y para la obtención de un buen rendimiento.

Para el desarrollo del sistema presentado, se utiliza un estándar de codificación, garantizando la homogeneidad del estilo de programación, durante la implementación del sistema.
A continuación se describen algunos de los elementos que conforman el estándar de codificación.

- Idioma: Todas las palabras deben ser en español, pero sin acentuarse. Se debe lograr una estructura uniforme para los bloques de código, así como para los diferentes niveles de anidamiento.
- Inicio y fin de bloque: Se deben dejar dos espacios en blanco desde la instrucción anterior, para el inicio y fin de bloque {}, asimismo, para el caso de las instrucciones if, else, for, while, do while, switch, foreach.
- Aspectos generales: Evitar el uso del tabulador, ya que puede variar según la computadora o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito, deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay solo una instrucción. No se debe colocar ({) en la línea de un código cualquiera, ya que requiere de una línea propia.

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

- Ubicación de comentarios: Estos se ubican al inicio de cada clase o función, y al final de cada bloque de código, especificando el objetivo de la misma, así como el tipo de dato y el objetivo de cada parámetro.
- Líneas en blanco: Estas se emplean antes y después de métodos, clases y estructuras.
- Espacios en blanco: Se recomienda entre operadores lógicos y aritméticos, para lograr una mayor legibilidad en el código.

Aspectos generales:

No se debe comentar cada línea de código. Si el comentario se aplica a un grupo de instrucciones, debe estar seguido de una línea en blanco. Cuando se comente una sola instrucción, se suprime la línea en blanco, o se escribe a continuación de la instrucción. No se usa un espacio en blanco después del corchete abierto y antes del cerrado de un arreglo, después del paréntesis abierto y antes del cerrado, o antes de un punto y coma.

En cuanto al uso de variables y contantes se tuvo en cuenta los siguientes elementos:

- Apariencia de variables: El nombre de las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto, se empleará la notación *Camel Casing*.
- Apariencia de constantes: Se declaran con todas sus letras en mayúscula.
- Aspectos generales: Los nombres de las variables y constantes, deben tener nombres que con solo leerlas se conozca el propósito.

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

- Apariencia de clases y objetos: Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto, se empleará notación *Pascal Casing*. Para el caso de las instancias, se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
- Apariencia de atributos: En caso de que sea un nombre compuesto, se empleará notación *Camel Casing*.
- Apariencia de las funciones: El nombre de las funciones se pondrá con verbos que denoten la acción que realizan y se empleará la notación Pascal Casing. Las funciones que obtienen algún dato, emplean el prefijo get y si fijan algún valor, se emplea el prefijo set.
- Declaración de parámetros en funciones: Se declaran agrupados por tipos, definiendo primero los string y luego los numéricos, además, se agrupan teniendo en cuenta los valores por defecto.
- Aspectos generales: El nombre que se emplea para las clases, objetos, atributos y funciones, debe permitir que con solo leerlo se conozca el propósito de los mismos.

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

- Apariencia de la base de datos: El nombre debe comenzar con el prefijo bd, a continuación un *underscoard,* y luego el nombre completamente en minúscula. Los nombres serán cortos y descriptivos.
- Apariencia de las tablas: El nombre comienza con el prefijo tb, seguido de underscore, y luego se escriben todas las letras en minúscula, en caso de ser un nombre compuesto, se utiliza underscore para separarlo.
- Tablas que representen relaciones: El nombre empleado para estas tablas comienza con el prefijo tr, seguido de underscore y la concatenación del nombre de las dos tablas que la generaron, separados por underscore, todo esto en minúscula.
- Tablas que representen nomencladores: El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn, seguido de underscoard, debe ser corto, descriptivo y todo en minúscula.
- Apariencia de los campos: El nombre de los campos se escribe con todas las letras en minúscula, para evitar problemas con el *Case Sensitive* del gestor.
- Nombre de los campos: Los campos identificadores comienzan con el identificador id, seguido de underscore y posteriormente el nombre del campo.
- Aspectos generales: El nombre empleado para la base de datos, tablas y campos, con solo leerlos se debe conoce el propósito de los mismos.

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

- Apariencia de los controles: El nombre que se le asigna a los controles, comienza con las primeras letras en minúscula, las cuales, identifican el tipo de dato al que se refiere. Para los nombres compuestos se emplea la notación *Camel Casing*.

Capítulo2: Descripción de la arquitectura de Software

En el presente capítulo se abordó la importancia de representar de forma explícita la arquitectura, en la realización y desarrollo del Módulo Neurología. Estas representaciones permiten elevar el nivel de abstracción, facilitando la comprensión de los sistemas de software complejos. Aumentan las posibilidades de reutilizar tanto la arquitectura, como los componentes que aparecen en ella. Y por último, si la notación utilizada tiene una base formal adecuada, es posible analizar la arquitectura del sistema, determinando cuáles son sus propiedades aún antes de construirlo. Todas estas características, posibilitan un mejor entendimiento en la comprensión y construcción de la aplicación.

En la descripción de la arquitectura del diseño propuesto, el punto fundamental fue la seguridad del módulo, debido a que en la actualidad, los sistemas y subsistemas basados en software, cada vez son más utilizados, por lo cual, la seguridad de los mismos se hace indispensable. Se tuvo en cuenta a la hora de asegurar el correcto funcionamiento de la aplicación y de esta forma evitar accidentes, los requisitos no funcionales, ya que los mismos son muy difíciles de determinar. Hay que tener en cuenta, que los errores ocasionados por los requisitos no funcionales, son los más difíciles y caros de resolver; por lo cual, es recomendable para el correcto funcionamiento, establecer restricciones en el producto que está siendo desarrollado. Si los requisitos no funcionales, no son plasmados, o incorporados al diseño correctamente, pueden constituir un problema grave para el módulo en desarrollo, debido a que afectan la seguridad del mismo.

De esta forma se puede concluir, que los requisitos no funcionales, así como la necesidad de implantar un buen sistema de seguridad, constituyen aspectos imprescindibles a tener en cuenta, en el desarrollo del Módulo Neurología.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

En el presente capítulo se realiza el análisis y descripción de la solución propuesta, para lo que se definen claramente los diagramas de clases y de secuencia, que son las entradas para el modelo de implementación, el modelo de datos y los diagramas de componentes; estos últimos, con el objetivo de mostrar las dependencias, entre las partes del código del sistema propuesto. Además, se describen las clases y entidades asociadas a cada diagrama representado.

3.1 Valoración crítica del diseño propuesto por el analista. Refinamiento del diseño.

Una de las ventajas que trae consigo la definición de un buen diseño, es el hecho de posibilitar la implementación del sistema sin equivocaciones. Además, el diseño describe las clases, la organización de las mismas en paquetes y subsistemas, encontrándose entre sus artefactos, el modelo de diseño.
A la hora de realizar una valoración del diseño, el refinamiento es sin dudas, un paso imprescindible. En él se describen los pasos a seguir y si el diseño no cumple con los requerimientos propuestos, la aplicación no podrá ser desarrollada exitosamente, acarreando errores, e invirtiendo tiempo extra en corregirlos.

Al Módulo Neurología se le realizó un minucioso análisis del diseño, arribándose a la conclusión de que algunos artefactos no cumplían con los requerimientos, necesarios para la implementación del sistema. Debido a esto, se le realizó un refinamiento al diagrama de clases del diseño, al cual se le incorporaron relaciones, métodos y atributos, pues los existentes no contaban con los requerimientos necesarios, para desarrollar correctamente cada una de las funcionalidades; teniendo que transformar conjuntamente los diagramas de secuencia. También se le realizó el refinamiento al diagrama de clases persistentes, agregándole nuevos atributos y relaciones, lo cual trajo consigo que se generara nuevamente el modelo de datos y el script de la base de datos. Se cambiaron además, los prototipos no funcionales, al no cumplir con los componentes necesarios para la realización de los casos de uso. Estos prototipos, también presentaban errores en el código y no contaban con las pautas de diseño definidas por el analista.

En la etapa de implementación, se hace necesario un buen diseño de clases basado en patrones, el cual posibilita el aumento de la reutilización del código. El framework Seam, hace posible la persistencia de los

datos, a través del uso de Hibernate, el cual permite abstraerse del gestor de base de datos que se utilice, además, esta ofrece muchas ventajas, al servir de puente entre el mundo relacional y la programación orientada a objetos, mediante el mapeo de las entidades del sistema. Por otra parte, es la interfaz principal del JPA, por lo cual es el responsable de llevar a cabo las operaciones: insertar, eliminar, modificar y listar.

Patrones de diseño.

Para el diseño del sistema se recurrió a los patrones *GRASP*, los cuales se utilizan para la asignación de responsabilidades, obteniéndose como resultado un bajo acoplamiento en el sistema. Fue necesario además, la utilización de los patrones Experto y Creador, pertenecientes a los patrones *GRASP,* lográndose de esta manera la conservación del encapsulamiento de la información.

En el diseño obtenido, se evidencia también el cumplimiento de los patrones de Bajo Acoplamiento y Alta Cohesión, los cuales se encargan, de que las dependencias entre clases sean mínimas y que cada elemento lleve a cabo una única labor dentro de la aplicación. Respondiendo a la arquitectura definida, se llevó a cabo, el modelo de diseño, cuyos criterios de empaquetamiento quedaron definidos por procesos y clases.

Cada proceso que se ha definido en el sistema, se ha graficado en su paquete individual, utilizando las clases que se encuentran en el paquete del repositorio, las cuales a su vez, se encuentran separadas en tres subpaquetes:

- **Sesiones:** Está compuesto por las clases controladoras autogeneradas, personalizadas y las controladoras propias del proceso. Siendo las autogeneradas, las que son creadas por el entorno de desarrollo, las personalizadas las que fueron modificadas y las controladoras, son aquellas que fueron creadas específicamente para el proceso.
- **Entidades:** Está integrado por el conjunto de entidades autogeneradas, que fueron obtenidas mediante el proceso de ingeniería inversa, utilizando el mapeo objeto – relacional de Hibernate. Además, contiene las entidades personalizadas, que son las que han sido modificadas.
- **Vistas:** Está compuesto por el conjunto de clases que representan la interfaz de usuario.

3.1.1 Diagrama de paquetes.

El modelo de diseño consiste en una colaboración entre clases, que pueden ser agregadas en paquetes y subsistemas. Mediante este modelo, se hace un refinamiento del proceso de análisis anteriormente realizado. Para ello, se tienen en cuenta los requisitos no funcionales del sistema, ya que el principal propósito del modelado del diseño, es el de crear un plano del modelo de implementación. También, se define la arquitectura del sistema. Los casos de uso son realizados por las clases del diseño y sus objetos, a partir de los cuales se forma el diagrama de clases del diseño.

"Para lograr un mejor entendimiento del modelo de diseño, se describen a continuación los términos empleados en el mismo:

- **Clase de diseño:** Es una abstracción de una clase o construcción, en la implementación del sistema.
- **Diagramas de clases de diseño:** Exponen un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema. Permiten visualizar, especificar y documentar modelos estructurales. Estos forman parte de las realizaciones de casos de uso.
- **Diagramas de interacción:** Modelan los aspectos dinámicos de un sistema, muestran gráficamente cómo los objetos se comunican entre sí, a fin de cumplir con los requerimientos." (48)

Dentro de los diagramas de interacción, se definen los diagramas de secuencia y colaboración. Generalmente se realizan los diagramas de colaboración en el análisis, y los de secuencia en el diseño.

- **Diagrama de secuencia:** Es un diagrama de interacción que destaca la ordenación temporal de los mensajes.

Para la elaboración del modelo de diseño, se define una estructura de paquetes que permite dividir el sistema en fragmentos manejables, para su futura implementación. Se emplea el criterio de empaquetamiento por procesos, siguiendo la estructura de procesos definidos en el sistema.

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

3.1.2 Diagrama de clases del diseño.

Dentro de los elementos que componen el modelo de diseño, se destacan los diagramas de clases del diseño y los diagramas de interacción. Los diagramas de diseño, describen gráficamente las características de las clases de software y de las interfaces en un sistema. Mientras que los diagramas de secuencia, muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso.

A continuación se muestran dos diagramas del diseño propuesto, el diagrama de clases del diseño y el diagrama de secuencia, pertenecientes al caso de uso: Crear Examen Físico Neurológico, los cuales se muestran en las figuras 2 y 3, respectivamente.

Diagrama de Clases del Diseño. Crear Examen Físico Neurológico

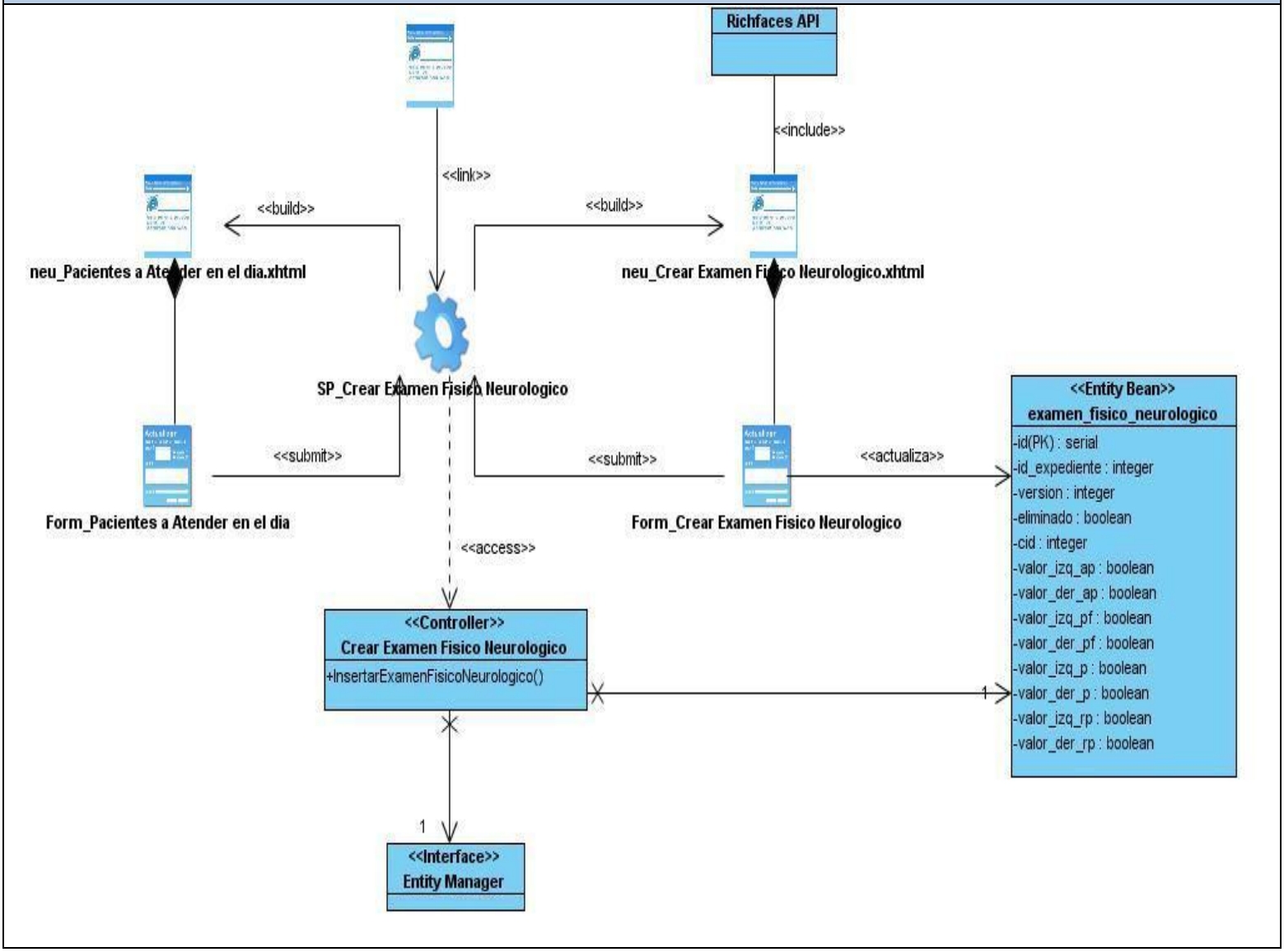

Figura 2: Diagrama de Clases del Diseño.

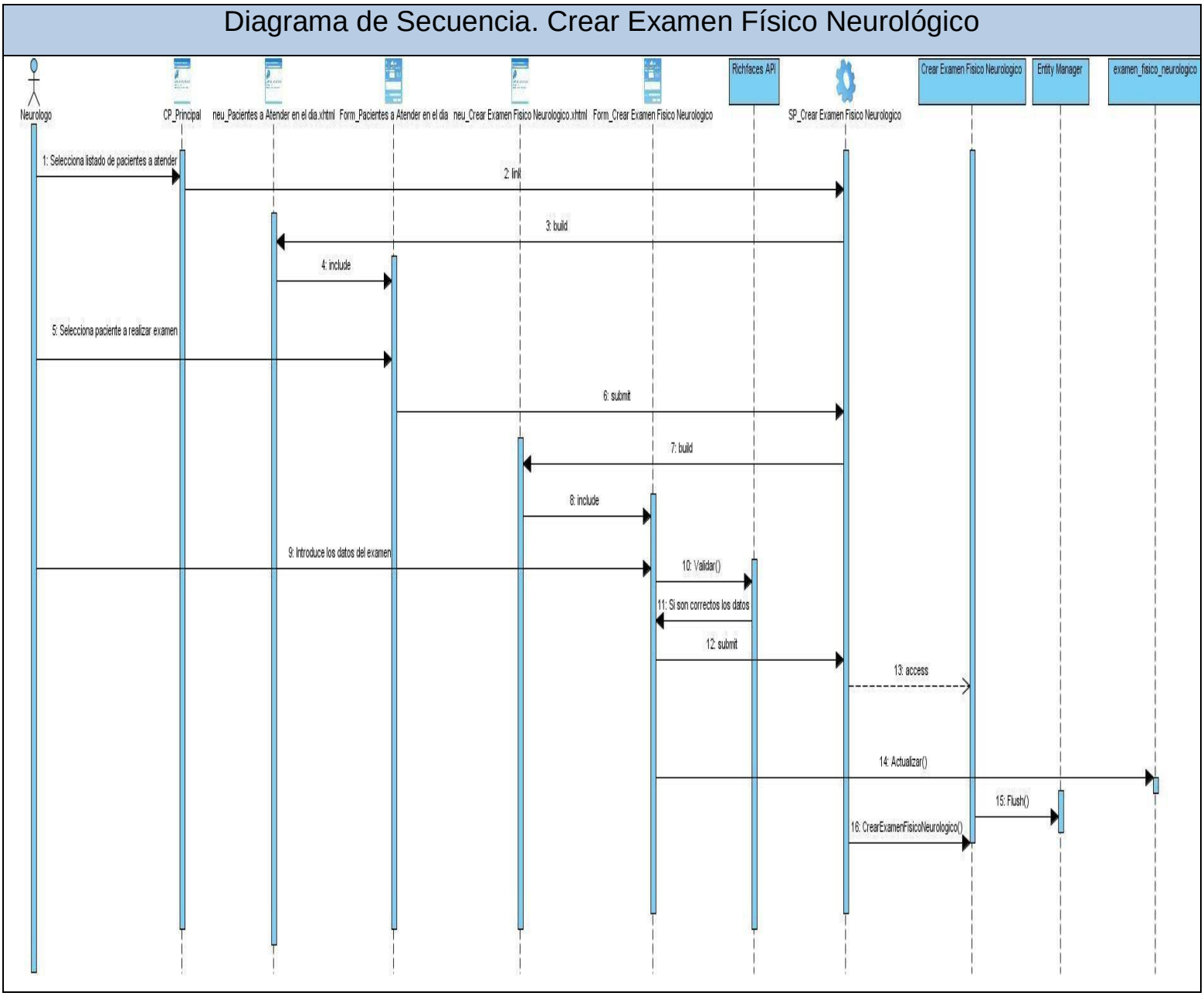

Figura 3: Diagrama de Secuencia.

3.2 Descripción de las nuevas clases u operaciones necesarias.

Al ocurrir cambios en la base de datos, y al no contar con todas las funcionalidades necesarias para desarrollar el Módulo Neurología, se hizo necesaria la incorporación de nuevas clases y atributos. Estas nuevas clases permiten un mejor desarrollo y dotan a la aplicación con una mayor robustez. A continuación se describen las clases controladoras, las cuales se pueden observar en las siguientes tablas:

Tabla 1: Gestionar Angulo Poplíteo Clasificación.

Nombre: Gestionar Ángulo Poplíteo Clasificación.	
Tipo de clase: Controladora	
Atributo	**Tipo**
id	int
existe	bool
nomAnguloPopitleoClasificacion	ModneuNAnguloPopitleoClasificacion
Para cada responsabilidad:	
Nombre:	configurarNomencladorAnguloPopitleoClasificacion
Descripción:	Inserta, elimina, modifica y lista, los elementos de AnguloPopitleoClasificacion, que no es más que una clasificación del nomenclador original AnguloPopitleo, por lo cual, el primero depende del segundo para ejecutarse.

Tabla 2: Gestionar Ángulos Aductores Clasificación.

Nombre: Gestionar Ángulos Aductores Clasificación.	
Tipo de clase: Controladora	
Atributo	**Tipo**
id	int
existe	bool
nomAnguloAductoresClasificacion	ModneuNAnguloAductoresClasificacion
Para cada responsabilidad:	

Nombre:	configurarNomencladorAnguloAductoresClasificacion
Descripción:	Inserta, elimina, modifica y lista, los elementos de AnguloAductoresClasificacion, que no es más que una clasificación del nomenclador original AngulosAductores, por lo cual, el primero depende del segundo para ejecutarse.

Tabla 3: Gestionar Electroencefalograma.

Nombre: Gestionar Electroencefalograma.	
Tipo de clase: Controladora	
Atributo	**Tipo**
id	int
existe	bool
nomEEG	ModneuNEeg
Para cada responsabilidad:	
Nombre:	configurarNomencladorEEG
Descripción:	Inserta, elimina, modifica y lista, los elementos del Electroencefalograma (EEG). Luego de realizar el examen general, es necesario realizar el examen complementario (EEG), por lo cual se creó, esta nueva clase.

Tabla 4: Gestionar Potenciales Evocados Auditivos de Tallo Cerebral.

Nombre: Gestionar Potenciales Evocados Auditivos de Tallo Cerebral.	
Tipo de clase: Controladora	
Atributo	**Tipo**
id	int
existe	bool
nomPeatc	ModneuNPeatc
Para cada responsabilidad:	
Nombre:	configurarNomencladorPeatc
Descripción:	Inserta, elimina, modifica y lista, los elementos de Potenciales Evocados Auditivos de

	Tallo Cerebral (Peatc). Luego de realizar el examen general, es necesario realizar el examen complementario (Peatc), por lo cual se creó, esta nueva clase.

Tabla 5: Gestionar Potenciales Evocados Somato-Sensoriales.

Nombre: Gestionar Potenciales Evocados Somato-Sensoriales.	
Tipo de clase: Controladora	
Atributo	**Tipo**
id	int
existe	bool
nomPess	ModneuNPess
Para cada responsabilidad:	
Nombre:	configurarNomencladorPess
Descripción:	Inserta, elimina, modifica y lista, los elementos del Potenciales Evocados Somato-Sensoriales. (Pess). Luego de realizar el examen general, es necesario realizar el examen complementario (Pess), por lo cual se creó, esta nueva clase.

Tabla 6: Gestionar Potenciales Evocados Visuales.

Nombre: Gestionar Potenciales Evocados Visuales.	
Tipo de clase: Controladora	
Atributo	**Tipo**
id	int
existe	bool
nomPev	ModneuNPev
Para cada responsabilidad:	
Nombre:	configurarNomencladorPev
Descripción:	Inserta, elimina, modifica y lista, los elementos del nomenclador. Luego de realizar el examen general, se realiza el examen complementario (Pev).

3.3 Modelo de datos.

Los modelos de datos, proporcionan cierto nivel de abstracción de datos, al ocultar las características sobre el almacenamiento físico, que la mayoría de los usuarios no necesita conocer. "Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos. El mismo, ayuda a entender y nombrar la información, evita la redundancia, asegura la corrección, validación y la completitud de los datos, además, su organización refleja la política del negocio. " (49)

Los atributos se representan, colocando su nombre dentro del rectángulo de la entidad. Además, pueden ser clasificados en: obligatorios, opcionales, claves foráneas y claves primarias. La representación gráfica de la clave primaria, es un signo de suma y la de la foránea, es un símbolo de número.

Las relaciones describen la interacción entre dos o más entidades. Se representan mediante una línea que une las dos entidades implicadas y tienen principalmente dos características: cardinalidad y obligatoriedad.

La cardinalidad, se refiere al número de ocurrencias de una entidad respecto a la otra. "La obligatoriedad en el modelo de datos, determina que ante la existencia de una entidad, deberá o podrá haber ocurrencias de otras entidades relacionadas. Esto se define a través de una línea continua, en caso de ser obligatoria la ocurrencia, o con una línea discontinua en caso de no serlo. " (50)

Uno de los aportes que brindan los modelos de datos, es el hecho de ofrecer la base conceptual para llevar a cabo el diseño de los sistemas, que trabajan con bases de datos. Además, cabe destacar que poseen un conjunto de operaciones básicas, para la realización de consultas y actualización de los datos.

Con la descripción de los componentes del modelo de datos, se tendrá una mejor comprensión del diagrama que se muestra en la figura 4.

Modelo de Datos.

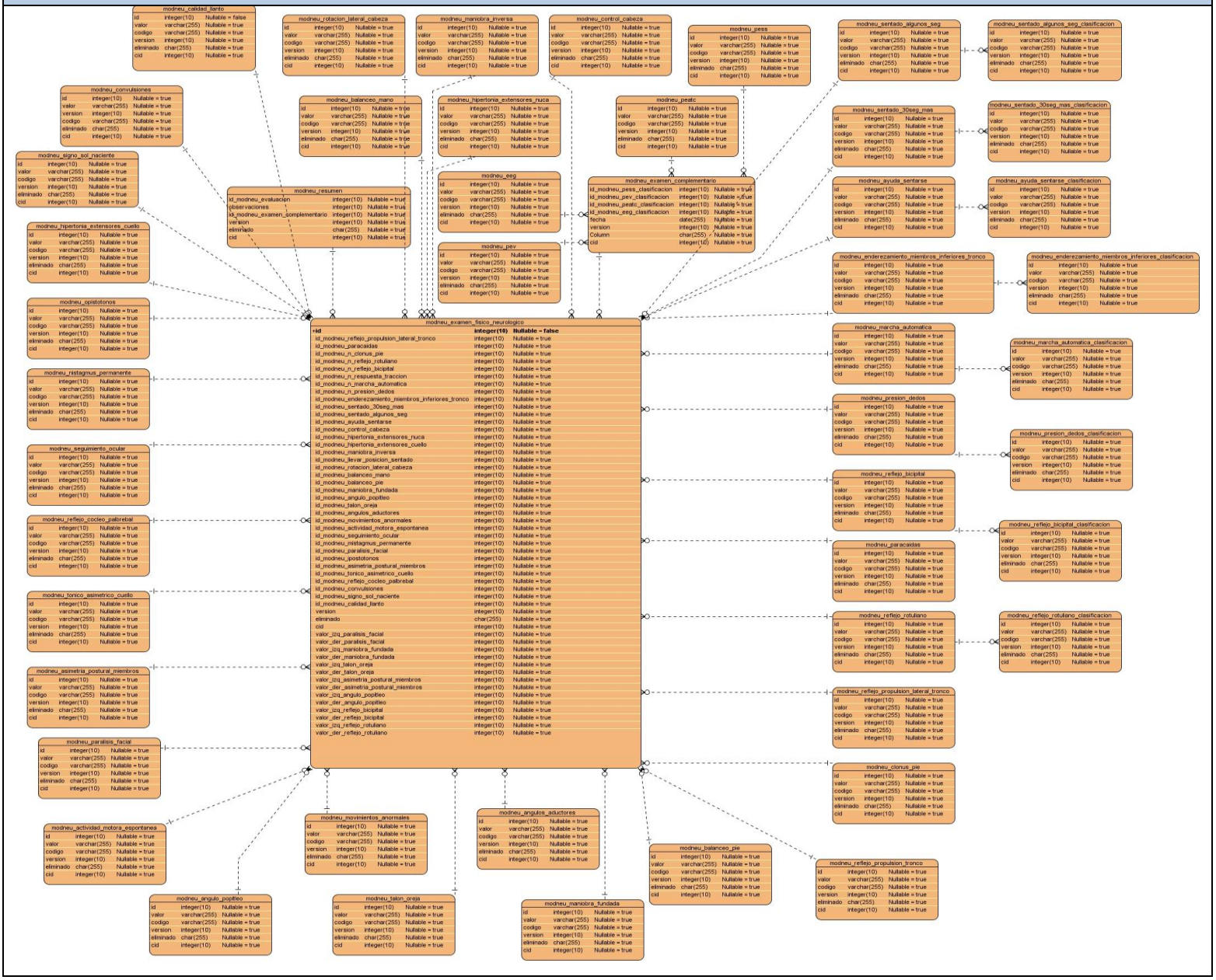

Figura 4: Modelo de Datos.

3.4 Valoración de las técnicas de validación.

Normalmente al realizarse el desarrollo de cualquier aplicación, se llevan a cabo varias validaciones en los datos que son introducidos por los usuarios del sistema. Este es un proceso para confirmar que los valores que se especifican en los objetos de datos, son compatibles con las restricciones dentro de un esquema del conjunto, al igual que las reglas establecidas para su aplicación. La validación de los datos, antes de enviar actualizaciones a la base de datos subyacente, es una buena práctica que reduce los errores y la cantidad potencial de acciones de ida y vuelta, entre una aplicación y la base de datos.

"Para confirmar que son válidos los datos que se escriben en un conjunto, se puede construir un conjunto de comprobaciones de validación, en el propio conjunto de datos. Este conjunto puede comprobar los datos independientemente de cómo se esté realizando la actualización, ya sea directamente, mediante los controles de un formulario, desde dentro de un componente, o de alguna otra manera. Dado que el conjunto de datos forma parte de la aplicación, es lógico construir una validación específica de la aplicación." (51)

3.5 Descripción de las tablas de la base de datos.

Para una mejor comprensión de la estructura de la base de datos con la que cuenta la aplicación, se realiza a continuación la descripción de algunas de sus tablas, en donde se detallan los atributos, relaciones y qué función cumple cada una de las tablas en la implementación del sistema. Esto se puede observar en las siguientes tablas:

Tabla 7: modneu_n_actividad_motora_espontanea.

Nombre: modneu_n_actividad_motora_espontanea		
Descripción: Tabla para gestionar los datos pertenecientes a los pacientes atendidos. Esta tabla solo contiene los datos relacionados con la actividad motora del paciente.		
Atributo	**Tipo**	**Descripción**
id serial	NOT NULL	Identificador del nomenclador dentro de las tablas.
valor	character varying	Es el valor por el cual se reconoce a cada nomenclador dentro de las tablas.
código	character varying	Es el código por el cual se reconoce al nomenclador y se utiliza de forma internacional, quiere decir que no varía si cambia el idioma.
eliminado	boolean	Permite saber si el área ha sido eliminada.
versión	integer	Indica con qué versión de la entidad se está trabajando
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 8: modneu_eeg_clasificacion.

Nombre: modneu_eeg_clasificacion		
Descripción: Tabla para gestionar los datos pertenecientes a los pacientes atendidos. Esta tabla solo contiene los datos relacionados con el electroencefalograma y es una clasificación del nomenclador original EEG.		
Atributo	**Tipo**	**Descripción**
id serial	NOT NULL	Identificador del nomenclador dentro de las tablas.
id_modneu_n_eeg	integer	Identificador del examen complementario realizado, referente al nomenclador EEG.
id_modneu_n_eeg_valor	integer	Identificador del valor que toma el EEG dentro de las tablas.
valor	character varying	Es el valor por el cual se reconoce a cada nomenclador dentro de las tablas.
código	character varying	Es el código por el cual se reconoce al nomenclador y se utiliza de forma internacional, quiere decir que no varía si cambia el idioma.
eliminado	boolean	Permite saber si el área ha sido eliminada.
versión	integer	Indica con qué versión de la entidad se está trabajando
cid	integer	Identificador de modificaciones registradas en la bitácora.

Tabla 9: modneu_enderezamiento_miembros_inferiores_tronco_clasificacion.

Nombre: modneu_enderezamiento_miembros_inferiores_tronco_clasificacion		
Descripción: Tabla para gestionar los datos pertenecientes a los pacientes atendidos. Esta tabla solo contiene los datos relacionados con el enderezamiento de los miembros inferiores y el tronco y es una clasificación del nomenclador original Enderezamiento Miembros Inferiores Tronco.		
Atributo	**Tipo**	**Descripción**
id_modneu_n_enderezamiento_miembros_inferiores_tronco	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador enderezamiento miembros inferiores tronco.
valor	character varying	Es el valor por el cual se reconoce a cada nomenclador dentro de las tablas.
código	character varying	Es el código por el cual se reconoce al nomenclador y se utiliza de forma internacional, quiere decir que no varía si cambia el idioma.
eliminado	boolean	Permite saber si el área ha sido eliminada.
versión	integer	Indica con qué versión de la entidad se está trabajando
cid	integer	Identificador de modificaciones registradas en la bitácora.
id	integer NOT NULL DEFAULT	Identificador del nomenclador dentro de las tablas.

Tabla 10: modneu_evaluacion.

Nombre: modneu_evaluacion		
Descripción: Tabla para mostrar todo el contenido de la evaluación realizada al paciente.		
Atributo	**Tipo**	**Descripción**
id serial	NOT NULL	Identificador del nomenclador dentro de las tablas.
id_expediente	integer NOT NULL	Identificador del expediente del paciente dentro de las tablas.
fecha	date NOT NULL	Fecha en que se realizó la evaluación.
eliminado	boolean	Permite saber si el área ha sido eliminada.
versión	integer	Indica con qué versión de la entidad se está trabajando
cid	Integer DEFAULT (-1)	Identificador de modificaciones registradas en la bitácora en la posición anterior.

Tabla 11: modneu_examen_fisico_neurologico.

Nombre: modneu_examen_fisico_neurologico		
Descripción: Tabla para gestionar los datos pertenecientes a los pacientes atendidos. Esta tabla contiene los datos relacionados con todo el proceso del examen físico neurológico.		
Atributo	**Tipo**	**Descripción**
id serial	NOT NULL	Identificador del nomenclador dentro de las tablas.
id_modneu_reflejo_propulsion_lateral_tronco_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador reflejo propulsión lateral tronco.

id_modneu_n_clonus_pie integer	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador clonus pie.
id_modneu_reflejo_rotuliano_ clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador reflejo rotuliano.
id_modneu_reflejo_bicipital_cl asificacion integer	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador reflejo bicipital.
id_modneu_respuesta_traccio n_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador respuesta tracción.
id_modneu_prension_dedos_ clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador presión dedos.
id_modneu_marcha_automati ca_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador marcha automática.
id_modneu_sentado_30seg_ mas_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador posición sentado 30 segundos.
id_modneu_sentado_algunos _seg_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador posición sentado algunos segundos.
id_modneu_ayuda_sentarse_ clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador ayuda sentarse.
id_modneu_control_cabeza_c	integer NOT NULL	Identificador de los exámenes

lasificacion		realizados, referentes al nomenclador control cabeza.
id_modneu_n_hipertonia_extensores_nuca	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador hipertonía extensores nuca.
id_modneu_n_maniobra_inversa	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador maniobra inversa.
id_modneu_n_rotacion_lateral_cabeza integer	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador rotación lateral cabeza.
id_modneu_n_balanceo_mano	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador balanceo mano.
id_modneu_n_balanceo_pie	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador balanceo del pie.
id_modneu_angulo_popitleo_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador ángulo poplíteo.
id_modneu_angulo_aductores_clasificacion	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador ángulos aductores.
id_modneu_n_movimientos_anormales	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador movimientos anormales.
id_modneu_n_actividad_moto	integer NOT NULL	Identificador de los exámenes

ra_espontanea		realizados, referentes al nomenclador actividad motora espontanea.
id_modneu_n_asimetria_postural_miembros	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador asimetría postural miembros.
id_modneu_n_seguimiento_ocular	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador seguimiento ocular.
id_modneu_n_convulsiones	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador convulsiones.
id_modneu_n_calidad_llanto	integer NOT NULL	Identificador de los exámenes realizados, referentes al nomenclador calidad de llanto.
versión	integer	Indica con qué versión de la entidad se está trabajando
eliminado	boolean	Permite saber si el área o la tupla han sido eliminadas.
cid	integer	Identificador de modificaciones registradas en la bitácora.
valor_izq_paralisis_facial	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador parálisis facial izquierda.
valor_der_paralisis_facial	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador parálisis facial derecha.

valor_izq_asimetria_postural_ miembros	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador asimetría postural miembros izquierda.
valor_der_asimetria_postural_ miembros	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador asimetría postural miembros derecha.
valor_izq_angulo_popliteo	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador ángulo poplíteo izquierda.
valor_der_angulo_popitleo	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador ángulo poplíteo derecha.
valor_izq_reflejo_bicipital	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador reflejo bicipital izquierda.
valor_der_reflejo_bicipital	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador reflejo bicipital derecha.
valor_izq_reflejo_rotuliano	boolean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador reflejo rotuliano izquierda.
valor_der_reflejo_rotuliano	boleean	Devuelve un valor que puede ser verdadero o falso respecto al

		nomenclador reflejo rotuliano derecha.
valor_izq_reflejo_propulsion_lateral_tronco	boleean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador reflejo propulsión izquierda.
valor_der_reflejo_propulsion_lateral_tronco	boleean	Devuelve un valor que puede ser verdadero o falso respecto al nomenclador reflejo propulsión derecha.
id_modneu_enderezamiento_miembros_inferiores_tronco_clasificacion	integer	Identificador de los exámenes realizados, referentes al nomenclador enderezamiento miembros inferiores tronco.

3.6 Vista de implementación.

La vista de implementación muestra el empaquetado físico de las partes reutilizables del sistema, en unidades sustituibles llamadas componentes, que no son más que piezas reutilizables de alto nivel, a partir de las cuales, se pueden construir diferentes sistemas. Esta vista tiene como propósito definir la organización del código, planificar las integraciones del sistema, necesarias en cada iteración, e implementar las clases y subsistemas definidos durante el diseño.

Es además, aquella vista que describe la organización del sistema en capas y subsistemas de implementación, describiendo la forma en la que se realiza la asignación de paquetes y clases existentes en la vista lógica, a los paquetes y módulos pertenecientes a la implementación, contándose entre uno de sus artefactos, al diagrama de componentes.

3.6.1 Diagrama de componentes.

Dentro de la vista de implementación se desarrollan diferentes artefactos, el diagrama de componentes es uno de ellos. "Este diagrama muestra la relación entre los componentes del software, sus dependencias, comunicaciones, localización y otras condiciones. Los diagramas de componentes son usados para estructurar los componentes en los sistemas del software. Ellos examinan y controlan las dependencias entre componentes, o interfaces de los componentes. Un componente, representa una parte modular, desplegable y reutilizable de un sistema." (52) En la figura 5 se expone el modelo de componentes definido en la solución propuesta.

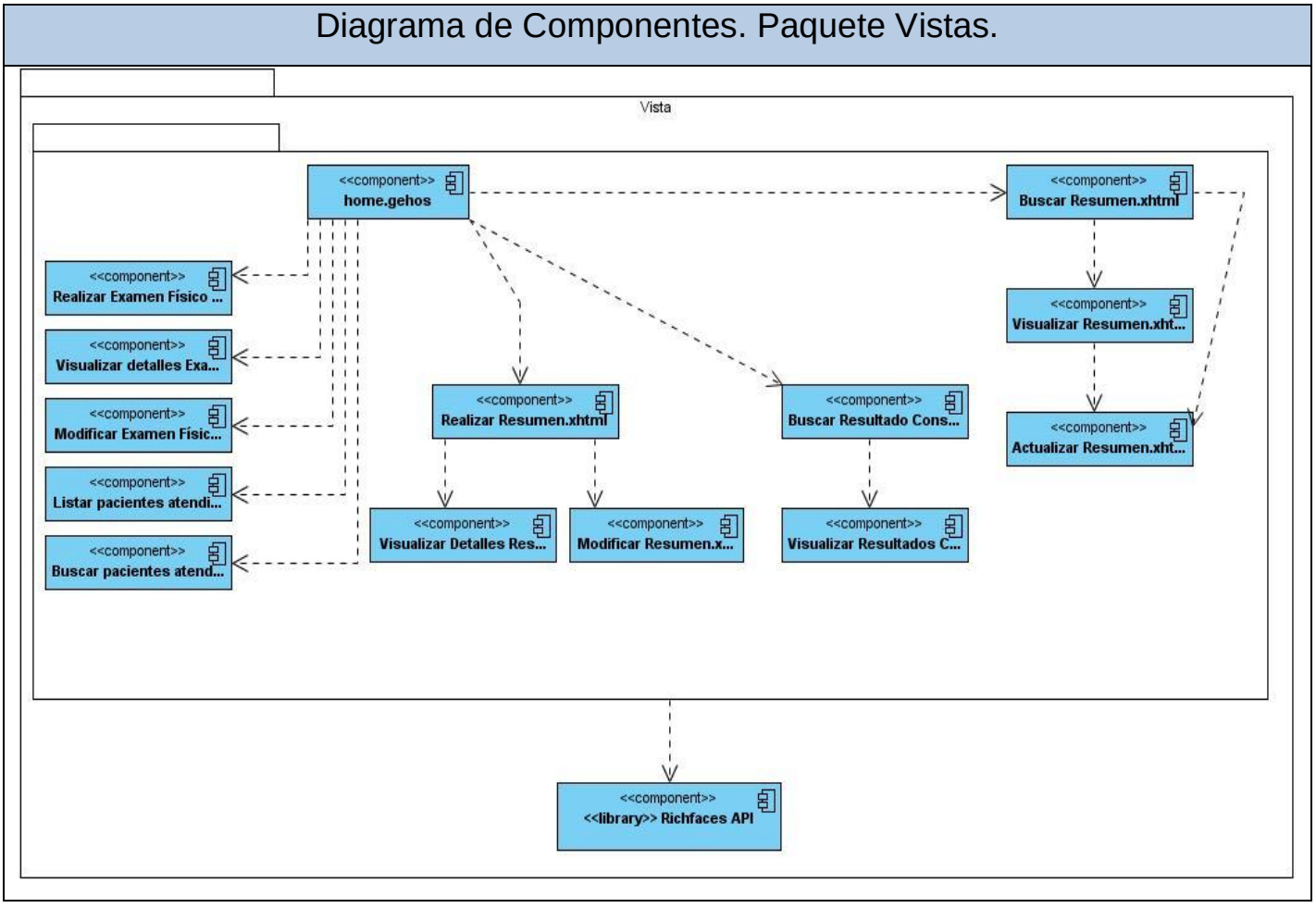

Figura 5: Diagrama de Componentes. Paquete Vistas.

Los diagramas de componentes se utilizan además, para describir la vista de implementación estática de un determinado sistema, ya que presenta un nivel más alto de abstracción que un diagrama de clases; usualmente, un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, y eventualmente, un componente puede comprender una gran porción de un sistema.

En la creación de este diagrama se tienen en cuenta los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en la implementación. Esto se muestra en las figuras 6 y 7 que representan los diagramas de paquetes de las controladoras y las modelos, respectivamente.

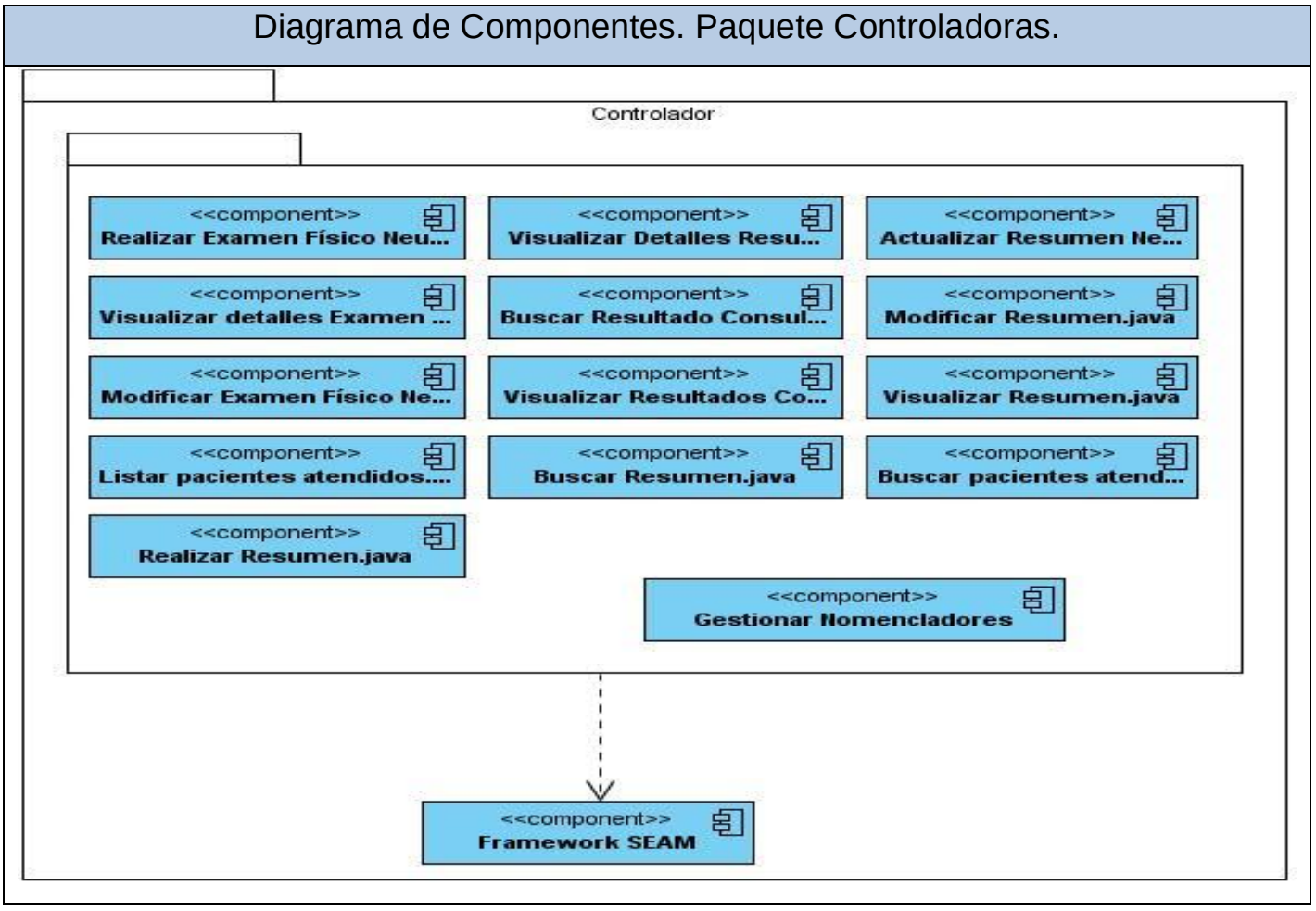

Figura 6: Diagrama de Componentes. Paquete Controladoras.

Diagrama de Componentes. Paquete Modelos.

Figura 7: Diagrama de Componentes. Paquete Modelos.

Capítulo 3: Descripción y Análisis de la Solución Propuesta

En el desarrollo de este capítulo se analizó y describió la solución propuesta. Para la implementación del Módulo Neurología se tenía definida una arquitectura y un diseño, en esta fase del proyecto se le da continuidad al mismo, realizando el diseño orientado a paquetes de prueba, lo que trae consigo, que la aplicación cuente con un correcto funcionamiento, ya que se siguieron los requerimientos necesarios en la implementación y que son los mismos que se tienen en el diseño propuesto.

Se mostraron además, los diagramas de clases del diseño, de secuencia, el modelo de datos y los diagramas de componentes, permitiendo todos ellos, un mejor entendimiento de la manera en la que está estructurado el sistema, constituyendo un eslabón fundamental en la implementación. También se describen las tablas de la base de datos, en donde se guardan todos los datos recogidos por el sistema y se describen las nuevas clases que fueron agregadas en la misma, con el objetivo de mantener una buena documentación de todos los cambios realizados.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

El desarrollo de un producto de software, implica la realización de una serie de actividades encaminadas a encontrar errores. Es por ello que se deben incorporar acciones, que evalúen la calidad del producto que se está desarrollando. Dentro del proceso de desarrollo, en específico el flujo de trabajo Prueba, es mediante el cual se valida, que las suposiciones hechas en el diseño y los requerimientos, se estén cumpliendo satisfactoriamente, por lo que se encarga de verificar que el producto funcione como se diseñó y que se cumplan los requerimientos.

En el presente capítulo se tiene como objetivo presentar el Modelo de Prueba, diseñado para comprobar la funcionalidad y efectividad con la que trabaja la aplicación. Se realiza además, una caracterización de las pruebas de caja negra, ya que estas son las que se le aplicarán al sistema, lo cual incluirá una descripción de los casos de prueba, asociados a las funcionalidades implementadas.

4.1 Pruebas de caja negra.

Las pruebas de software constituyen uno de los flujos de mayor importancia dentro de la Ingeniería de Software, ya que estos procesos permiten verificar la calidad de los sistemas, al poder identificar fallos en la aplicación, ya sean de usabilidad, o de implementación. Un buen diseño de prueba, muestra las diferentes clases de errores que puedan surgir al ejecutar el sistema, realizándose en menos tiempo y requiriendo poco esfuerzo.

Las técnicas de evaluación dinámica o prueba, proporcionan distintos criterios para generar casos de prueba, que provoquen fallos en los programas. Estas técnicas se agrupan en:

- Técnicas estructurales o de caja blanca: Se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario, conocer la lógica del programa.
- Técnicas funcionales o de caja negra: Realizan pruebas sobre la interfaz del programa, entendiendo por interfaz, las entradas y las salidas de dicho programa.

En la Figura 8 se representa gráficamente la filosofía de las pruebas de caja blanca y caja negra.

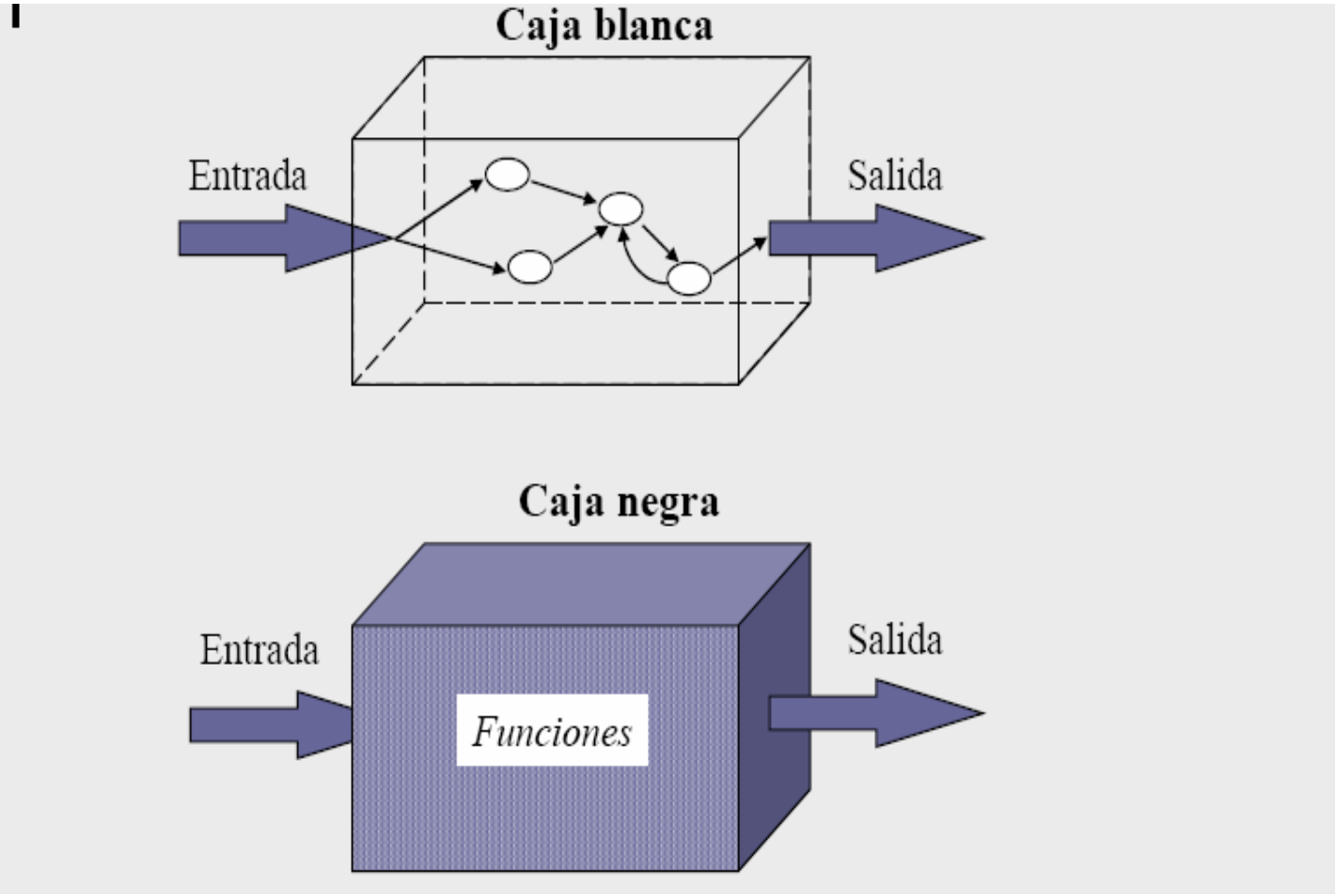

Figura 8: Representación de pruebas de caja blanca y caja negra.

Como se puede observar, las pruebas de caja blanca necesitan conocer los detalles procedimentales del código, mientras que las de caja negra únicamente necesitan saber el objetivo, o funcionalidad que el código ha de proporcionar.

Las pruebas de caja negra son aquellas que se centran en revisar las funcionalidades del sistema, es decir, se lleva a cabo una búsqueda de diferentes tipos de errores, los cuales no son visibles al utilizarse pruebas de caja blanca. Estas pruebas son completamente indiferentes al comportamiento interno y la estructura del programa. Con ellas se pretende demostrar, que las funcionalidades del sistema son

operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que se mantiene la integridad de la información externa.
Entre los diferentes errores que los métodos de caja negra son capaces de detectar, se encuentran:

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

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas se encuentran:

- Técnica de partición de equivalencia.
- Técnica del análisis de valores límite.
- Técnica de grafos de causa-efecto.

Las técnicas o pruebas de caja negra se basan fundamentalmente, en las entradas y salidas de datos que se producen en un sistema, sin importar el funcionamiento del código interno, es decir, lo más importante será la forma en la que el sistema interactúe con el medio, mostrando respuestas correctas en cada caso.

Técnica de particiones de equivalencia.

"La partición de equivalencia es un método de prueba de caja negra, que divide el campo de entrada de un programa en clases de datos, de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba, que descubran diferentes tipos de errores, reduciendo así, el número total de casos de prueba que hay que desarrollar." (53)

Este método intenta dividir el dominio de entrada de un programa, en un número finito de clases de equivalencia, de tal modo que se pueda asumir razonablemente, que una prueba realizada con un valor representativo de cada clase, es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir, que si el caso de prueba correspondiente a una clase de equivalencia, detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error y

viceversa, si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia, encuentren errores.
El diseño de casos de prueba según esta técnica consta de dos pasos:

- Identificar las clases de equivalencia.
- Identificar los casos de prueba.

Técnica del análisis de valores límite.
Los casos de prueba que exploran las condiciones límite, producen mejor resultado que aquellos que no lo hacen. Las condiciones límite, son aquellas que se encuentran en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite, como técnica de prueba. Esta técnica, permite elegir los casos de prueba que ejerciten esos valores. Este análisis complementa la técnica de partición de equivalencia, de manera que en lugar de seleccionar cualquier caso de prueba de las clases válidas e inválidas, se eligen los casos de prueba en los extremos, y en lugar de centrarse solo en el dominio de entrada, los casos de prueba se diseñan también considerando el dominio de salida.

"Las pautas para desarrollar casos de prueba con esta técnica son:

- Si una condición de entrada especifica un rango de valores, se diseñarán casos de prueba para los dos límites del rango, y otros dos casos para situaciones justo por debajo y por encima de los extremos.
- Si una condición de entrada especifica un número de valores, se diseñan dos casos de prueba para los valores mínimo y máximo, además, se realizarán otros dos casos de prueba para valores justo por encima del máximo y por debajo del mínimo.
- Si la entrada o salida de un programa es un conjunto ordenado, habrá que prestar atención al primer y último elemento del conjunto." (54)

Técnica de grafos de causa-efecto

"El uso de grafos de causa - efecto es una técnica de casos de prueba, que proporciona una concisa representación de las condiciones lógicas y sus correspondientes acciones. La técnica sigue cuatro pasos:

- Se listan para un módulo las causas (condiciones de entrada) y los efectos (acciones), asignando un identificador a cada uno de ellos.
- Se desarrolla un grafo causa – efecto.
- Se convierte el grafo en una tabla de decisión.
- Se convierten las reglas de la tabla de decisión en casos de prueba." (55)

Si se ha usado una tabla de decisión como herramienta de diseño, los grafos causa - efecto ya no son necesarios. En este método se crea un grafo de objetos importantes y sus relaciones, y se diseñan además, una serie de pruebas que cubran el grafo, de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

La técnica que ha sido escogida para realizar estas pruebas en el sistema, es la técnica de partición de equivalencia. Esta prueba permite dividir el dominio de entrada de una aplicación en clases de datos, derivando a partir de los mismos, los casos de prueba, y representando cada una de estas clases de equivalencias, en un grupo de estados válidos e inválidos para las condiciones de entrada, por lo cual, esta técnica ofrece una amplia ventaja a la hora de detectar posibles errores en tiempo de ejecución.

4.2 Descripción de los casos de prueba.

Un caso de prueba no es más que un conjunto de condiciones, bajo las cuales se introducen datos, con el objetivo de obtener varios resultados, permitiendo determinar si se ha cumplido satisfactoriamente el desarrollo de las funcionalidades que se han estado probando. Se puede saber si un caso de prueba es aceptable, si el mismo presenta una alta probabilidad de detectar un error, que no haya sido encontrado hasta el momento.

A continuación se muestra uno de los casos de prueba, que fue diseñado para ser aplicado a las funcionalidades con las que cuenta el sistema. En este caso solo se mostrará el nomenclador: Gestionar Ayuda Sentarse, el cual se puede observar en las siguientes tablas:

Tabla 12: Caso de prueba: Gestionar Ayuda Sentarse.

Gestionar Ayuda Sentarse	Escenarios del Gestionar Ayuda Sentarse	Descripción de la funcionalidad
SC 1: Adicionar Nomenclador Ayuda Sentarse.	EC 1.1: Adicionar Nomenclador Ayuda Sentarse exitosamente.	Se agrega un nuevo Nomenclador Ayuda Sentarse con los datos correspondientes.
	EC 1.2: Existen campos incompletos.	No se agrega el Nomenclador Ayuda Sentarse porque existen campos incompletos.
	EC 1.3: Existen campos incorrectos.	No se agrega el Nomenclador Ayuda Sentarse porque existen campos incorrectos.
SC 2: Modificar Nomenclador Ayuda Sentarse.	EC 2.1: Modificar Nomenclador Ayuda Sentarse exitosamente.	Se modifica un nuevo Nomenclador Ayuda Sentarse con los datos correspondientes.
	EC 2.2: Existen campos incompletos.	No se modifica el Nomenclador Ayuda Sentarse porque existen campos incompletos.
	EC 2.3: Existen campos incorrectos.	No se modifica el Nomenclador Ayuda Sentarse porque existen campos incorrectos.
SC 3: Eliminar Nomenclador Ayuda Sentarse.	EC 3.1: Eliminar el Nomenclador Ayuda Sentarse exitosamente.	Se elimina el Nomenclador Ayuda Sentarse exitosamente.
	EC 3.2: No Eliminar Nomenclador Ayuda Sentarse.	No se elimina el Nomenclador Ayuda Sentarse.
SC 4: Buscar Nomenclador Ayuda Sentarse.	EC 4.1: Busca el Nomenclador Ayuda Sentarse exitosamente.	Realiza un listado del Nomenclador Ayuda Sentarse al cual se le aplicará el criterio de búsqueda, si no lo encuentra por los parámetros especificados, sale un mensaje informando: “No se encontró información que cumpla con los criterios de búsqueda”.

Tabla 13: Adicionar Nomenclador Ayuda Sentarse.

Escenario	Variable 1 *(Código)*	Variable 2 *(Valor)*	Respuesta del Sistema	Flujo Central
EC 1.1: Adicionar Nomenclador Ayuda Sentarse exitosamente.	V	V	El sistema adiciona un nuevo Nomenclador Ayuda Sentarse.	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Adicionar Nomenclador Ayuda Sentarse.
EC 1.2: Existen campos incompletos.	I	V	El sistema muestra un carácter especial (asterisco rojo sobre el campo de entrada incompleto).	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Adicionar Nomenclador Ayuda Sentarse.
	V	I		
EC 1.3: Existen campos incorrectos.	I	V	El sistema muestra un carácter especial (asterisco rojo sobre el campo de entrada incorrecto).	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Adicionar Nomenclador Ayuda Sentarse.
	V	I		

Tabla 14: Ver Datos Nomenclador Ayuda Sentarse.

Escenario	Variable 1 *(Código)*	Variable 2 *(Valor)*	Respuesta del Sistema	Flujo Central
EC 2.1: Ver datos del Nomenclador Ayuda Sentarse exitosamente.	NA	NA	Se muestra los datos del Nomenclador Ayuda Sentarse seleccionado.	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Ver.
EC 2.2: Salir del Nomenclador Ayuda Sentarse exitosamente.	NA	NA	Se visualiza el listado de los nomencladores de Ayuda Sentarse.	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Ver. • Botón Salir.

Tabla 15: Modificar Nomenclador Ayuda Sentarse.

Escenario	Variable 1 *(Código)*	Variable 2 *(Valor)*	Respuesta del Sistema	Flujo Central
EC 1.1: Modificar Nomenclador Ayuda Sentarse exitosamente.	V	V	El sistema modifica el Nomenclador Ayuda Sentarse.	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Modificar Nomenclador Ayuda Sentarse.
EC 1.2: Existen campos incompletos.	I	V	El sistema muestra un carácter especial (asterisco rojo sobre el campo de entrada incompleto).	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Modificar Nomenclador Ayuda Sentarse.
	V	I		
EC 1.3: Existen campos incorrectos.	I	V	El sistema muestra un carácter especial (asterisco rojo sobre el campo de entrada incorrecto).	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Modificar Nomenclador Ayuda Sentarse.
	V	I		

Tabla 16: Eliminar Nomenclador Ayuda Sentarse.

Escenario	Variable 1 *(Código)*	Variable 2 *(Valor)*	Respuesta del Sistema	Flujo Central
EC 4.1: Eliminar el Nomenclador Ayuda Sentarse exitosamente.	NA	NA	Se muestra el listado del Nomenclador Ayuda Sentarse.	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Eliminar. • Opción del mensaje: “Si”.
EC 4.2: No Eliminar Nomenclador Ayuda Sentarse.	NA	NA	Se muestra el listado del Nomenclador Ayuda Sentarse.	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Vínculo Eliminar. • Opción del mensaje: “No”.

Tabla 17: Buscar Nomenclador Ayuda Sentarse.

Escenario	Variable 1 *(Código)*	Variable 2 *(Valor)*	Respuesta del Sistema	Flujo Central
EC 5.1: Busca el Nomenclador Ayuda Sentarse exitosamente.	NA	NA	Realiza un listado del Nomenclador Ayuda Sentarse al cual se le aplicará el criterio de búsqueda, si no lo encuentra por los parámetros especificados, sale un mensaje informando: "No se encontró información que cumpla con los criterios de búsqueda".	• Menú Neurología. • Submenú Configuración. • Vínculo Ayuda Sentarse. • Botón Buscar.

Al aplicar las pruebas de calidad, estas mostraron una serie de inconformidades, las cuales no afectaban el funcionamiento operativo del sistema, pero sí atentaban contra la excelencia y la calidad con la que debe cumplir el mismo.

Durante la primera iteración de las pruebas, se detectaron múltiples errores en los nomencladores. Algunos permitían la entrada a la aplicación de caracteres extraños (entiéndase: *, -, +, @, #, ~, %, &, etc.). Para darle solución a esta inconformidad, se le realizaron validaciones al sistema, permitiendo solamente la entrada de datos en los campos de texto, en letras o números, y mostrando un error en caso contrario. Los nomencladores no cumplían con las pautas de diseño establecidas para la aplicación, por lo cual se le realizaron mediciones a todas las interfaces, estableciendo para cada una de ellas las pautas definidas, obteniéndose uniformidad y homogeneidad en todas las páginas de la aplicación.

En las interfaces gráficas las palabras salían distorsionadas y esto ocasionaba que la interacción del usuario con la aplicación fuera dificultosa, debido a que los comandos de datos de entrada, eran prácticamente ilegibles; esto representó otra de las inconformidades encontradas. Para ello se cambiaron todas las propiedades encargadas de mostrar estos datos, llegándose a resolver de esta manera el problema. Por último, se detectaron algunos errores de programación en los nomencladores, debido a que no modificaban correctamente los datos entrados, por lo que se realizó una revisión del código. Los errores fueron detectados y posteriormente corregidos, permitiendo el uso operativo y efectivo de la funcionalidad para modificar datos.

Las pruebas de calidad mostraron en total veinte inconformidades, todas fueron resueltas satisfactoriamente, por lo cual el Módulo Neurología cumple con los requerimientos operativos y de calidad.

En este capítulo se realizó una descripción de las pruebas de caja negra, arribándose a la conclusión que las mismas, constituyeron parte fundamental en la ejecución de pruebas en cada fase del desarrollo, lográndose de esta forma, un sistema con un alto nivel de calidad, por lo que tendrá un mayor grado de aceptación. Además, se describieron los casos de prueba, que son los artefactos que genera el Flujo de Trabajo Prueba, los cuales fueron diseñados para cada una de las funcionalidades implementadas; permitiendo la aplicación de estos casos de prueba, la obtención de un sistema capaz de cumplir con todos los requisitos. Es importante comprender que las pruebas nunca demuestran que un programa es correcto, siempre es posible que existan errores aún después de realizadas.

CONCLUSIONES

En correspondencia con los requerimientos del Módulo Neurología y los objetivos trazados en la investigación, se concluye:

- La utilización del RUP como metodología de desarrollo de software, permitió la obtención del Módulo Neurología, cumpliendo con los requerimientos definidos por el analista; todo ello mediante un proceso continuo de pruebas y retroalimentación, lo que posibilitó la obtención de una aplicación robusta, fiable y segura.
- La arquitectura, tecnologías y herramientas definidas, favorecieron el desarrollo de la aplicación web; mostrando una interfaz sencilla, robusta y flexible, que gestiona correctamente la información generada por los procesos analizados.
- La aplicación del patrón de diseño Modelo-Vista-Controlador, resultó conveniente, ya que permitió la organización de las funcionalidades en los tres componentes de diseño según sus responsabilidades, disminuyendo la complejidad en la implementación de los casos de uso.
- La utilización de las pautas de diseño en el proceso de desarrollo, contribuyó a alcanzar una uniformidad y homogeneidad visual, de todas las interfaces de la aplicación.
- La ejecución de las pruebas de caja negra, garantizaron la calidad necesaria para la futura explotación de las funcionalidades implementadas. Con estas pruebas se demostró que las funcionalidades eran operativas, que la entrada se aceptaba de forma adecuada, produciendo un resultado correcto, y que se mantenía además, la integridad de la información externa.

La implementación del Módulo Neurología contribuirá a mejorar los servicios y la gestión de la información en las instituciones hospitalarias, ya que aumentará la eficiencia de las acciones que tienen lugar en esta área; permitiendo la obtención de una solución informática dotada de la seguridad necesaria, para garantizar la confidencialidad, integridad y disponibilidad de la información de los pacientes.

RECOMENDACIONES

Las autoras recomiendan:

- Estandarizar los diagnósticos emitidos por los neurólogos para apoyar la generación de reportes estadísticos.
- Aplicarle al Módulo Neurología otras pruebas de calidad que permitan evaluar la aplicación desde una mejor perspectiva; teniendo en cuenta para ello la complejidad del código, lo que permitirá que se obtengan las mismas funcionalidades, pero implementadas de forma mucho más óptima y dotando al sistema de una mayor robustez.
- Que se valore por las autoridades competentes la extensión del uso de la aplicación, a todas aquellas instituciones hospitalarias que contengan el Programa de Atención Temprana.

REFERENCIAS BIBLIOGRÁFICAS.

1. [Online] [Cited: Enero 15, 2011.] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm. http://bitacoramedica.com/weblog/wpcontent/uploads/2009/09/Telefonica_TIC_Cap_II.pdf.

2. Dirección de Informática del Minsap. . [Online] [Cited: Enero 15, 2011.] http://www.di.sld.cu/foro/viewtopic.php?f=8&t=685.

3. [Online] [Cited: Enero 15, 2011.] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm.

4. [Online] [Cited: Febrero 11, 2011.] http://www.uci.cu/?q=node/48.

5. Federación Estatal de Asociaciones de Profesionales de Atención Temprana GAT. . [Online] [Cited: Marzo 5, 2011.] http://www.gat-atenciontemprana.org/1_AtencionTemprana/index.htm..

6. Estimulacion Temprana en la primera infancia. . [Online] [Cited: Marzo 5, 2011.] http://estimulaciontempranaenprimerainfancia.blogspot.com/2009/02/intervencion-temprana-en-ninos-con.html..

7. Estimualcion Temprana y Desarrollo Infantil. . [Online] [Cited: Marzo 5, 2011.] http://estimulacionydesarrollo.blogspot.com/..

8. **Mestre Villavicencio, Pedro.** *Proyecto Renacer Contigo. .* Ciudad de la Habana : s.n.

9. *Proyecto Renacer Contigo. .* Ciudad de la Habana : s.n.

10. **Rodríguez, Pedro Luis.** *Historia de la Neurología en Cuba. . .* Ciudad de la Habana. : s.n., 2008.

11. Pediatría. . [Online] [Cited: Febrero 11, 2011.] http://www.pediatraldia.cl/que_neurologia.htm.

12. Cosas de la Infancia. . [Online] [Cited: Febrero 10, 2011.] http://www.cosasdelainfancia.com/biblioteca-esti-t-g.htm.

13. Rev Cubana de Pediatría. . [Online] [Cited: Enero 10, 2011.] http://bvs.sld.cu/revistas/ped/vol68_2_96/ped11296.htm.

14. ¿Qué es la neurología? . [Online] [Cited: Enero 10, 2011.] http://svneurologia.org/pacientes/neurologia.htm.

15. **Medina Tornés, Yordanis.** *Una alternativa para modelamiento de negocio con RUP.*

16. Articulos sobre RUP. . [Online] [Cited: Febrero 11, 2011.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

17. **Gómez Gallego, Juan Pablo.** *Fundamentos de la metodología RUP.*

18. **Díaz, Luis Carlos. Carrillo, Angela. Alvarado, Deicy.** *Análisis y Diseño Orientado a Objetos.*

19. [Online] [Cited: Febrero 12, 2011.] http://www.slideshare.net/oscar8711/introduccin-a-rup-presentation..

20. [Online] [Cited: Febrero 11, 2011.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf..
21. [Online] [Cited: Enero 12, 2011.] http://sites.google.com/site/flaviodanesse/aprendiendo-java/patrones-de.
22. [Online] [Cited: Enero 12, 2011.] http://www.mitecnologico.com/Main/ArquitecturaDelSoftware.
23. [Online] [Cited: Enero 10, 2011.] http://www.mitecnologico.com/Main/ArquitecturaDelSoftware.
24. Una introducción a Java y la IDE NetBeans. . [Online] [Cited: Enero 12, 2011.] http://www.whyfloss.com/pages/conference/static/editions/res07/charla1.pdf..
25. Taringa! [Online] [Cited: Enero 12, 2011.] http://www.taringa.net/posts/downloads/6774448/soft-para-programar-en-java-_-ejercicios-hechos-por-mi___.html.
26. Java Server Faces 2: Un vistazo. [Online] [Cited: Enero 12, 2011.] http://coresware.com/web/java-server-faces-2-un-vistazo/.
27. Adictos al Trabajo. . [Online] [Cited: Enero 12, 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro..
28. Junta de Andalucia. [Online] [Cited: Enero 12, 2011.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces..
29. Introducción a Ajax4jsf. . [Online] [Cited: Enero 12, 2011.] http://www.adictosaltrabajo.com/tutoriales/Ajax4Jsf.php.
30. Desarrollo en Web, Facelets y JSF – Uso de Templates. . [Online] [Cited: Enero 14, 2011.] http://blogs.antartec.com/desarrolloweb/2008/12/facelets-y-jsf-uso-de-templates..
http://www.webtaller.com/construccion/lenguajes/html/lecciones/que-es-xhtml.php..
31. Fap-Devel. . [Online] [Cited: Enero 14, 2011.] http://code.google.com/p/fap-devel/wiki/JBossSeam..
32. Slideshare.net. . [Online] [Cited: Enero 14, 2011.] http://www.slideshare.net/ingeniods/persistencia-de-objetos-con-hibernate..
33. Junta de Andalucia. . [Online] [Cited: Enero 14, 2011.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Hibernate.
34. Curso de Java. [Online] [Cited: Enero 14, 2011.] http://www.scribd.com/doc/39973855/Curso-Java-J2EE-Completo.
35. Domotica. . [Online] [Cited: Enero 14, 2011.] http://www.domotica.us/JPA.

36. Curso de Java. . [Online] [Cited: Enero 14, 2011.] http://www.scribd.com/doc/39973855/Curso-Java-J2EE-Completo.
37. Taringa! . [Online] [Cited: Enero 14, 2011.] http://www.taringa.net/posts/info/6858927/UML_-Lenguaje-Unificado-de-Modelado.html..
38. Domotica. [Online] [Cited: Enero 14, 2011.] http://www.domotica.us/JBOSS.
39. SGBD: Sistemas Gestionadores de Base de Datos. . [Online] [Cited: Enero 15, 2011.] http://jorge613.wordpress.com/2010/05/27/sgbd-sistemas-gestionadores-de-base-de-datos/..
40. Curso Base de Datos PostgreSQL, SQL avanzado y PHP. . [Online] [Cited: Enero 15, 2011.] http://www.usabilidadweb.com.ar/postgre.php..
41. [Online] [Cited: Enero 15, 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/..
42. [Online] [Cited: Febrero 12, 2011.] http://www.buenastareas.com/ensayos/Pruebas-De-Software/1357326.html.
43. Introducción al entorno de desarrollo Eclipse. . [Online] [Cited: Enero 15, 2011.] http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf..
44. [Online] [Cited: Febrero 12, 2011.] http://www.buenastareas.com/temas/casos-de-pruebas-caja-blanca-software/40.
45. Maria Eugenia Arevalo's Blog. . [Online] [Cited: Enero 16, 2011.] http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/..
46. Programador PHP+AS. . [Online] [Cited: Enero 20, 2011.] http://www.sgmweb.es/modelo.asp..
47. TRABAJO DE INVESTIGACIÓN, ANALISIS Y DISEÑO DE SISTEMAS II. . [Online] [Cited: Enero 22, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=6&ved=0CCoQFjAF&url=http%3A%2..
48. Monografias. com. [Online] [Cited: Marzo 16, 2011.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml.
49. [Online] [Cited: Marzo 16, 2011.] http://www3.uji.es/~mmarques/f47/apun/node32.html.
50. **Marqués Andrés, M. M. .** Sistemas de Bases de Datos. [Online] [Cited: Marzo 16, 2011.] http://www3.uji.es/~mmarques/f47/apun/node32.html..
51. Información general sobre validación de datos. . [Online] [Cited: Marzo 18, 2011.] http://msdn.microsoft.com/es-es/library/kx9x2fsb%28VS.80%29.aspx.

52. [Online] [Cited: Marzo 16, 2011.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml#dpaquet.

53. **Mañas, José A..** *Prueba de Programas.*

54. Herramientas de prueba de software. [Online] [Cited: Abril 16, 2011.] http://herrorsoft.zxq.net/pruebacajanegra.html.

55. [Online] [Cited: Abril 16, 2011.] http://html.rincondelvago.com/ingenieria-de-software_4.html.

BIBLIOGRAFÍA.

1. [Online] [Cited: Enero 15, 2011.] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm.

2. Kerdel-Vegas, Francisco. Aplicaciones de las TIC en el sector de la salud del futuro. [Online] Julio 2009. [Cited: Febrero 25, 2011.] http://bitacoramedica.com/weblog/wp-content/uploads/2009/09/Telefonica_TIC_Cap_II.pdf.

3. Dirección de Informática del Minsap. . [Online] [Cited: Enero 15, 2011.] http://www.di.sld.cu/foro/viewtopic.php?f=8&t=685.

4. [Online] [Cited: Enero 15, 2011.] http://bvs.sld.cu/revistas/spu/vol32_3_06/spu15306.htm.

5. [Online] [Cited: Febrero 11, 2011.] http://www.uci.cu/?q=node/48.

6. Federación Estatal de Asociaciones de Profesionales de Atención Temprana GAT. . [Online] [Cited: Marzo 5, 2011.] http://www.gat-atenciontemprana.org/1_AtencionTemprana/index.htm..

7. Estimulacion Temprana en la primera infancia. . [Online] [Cited: Marzo 5, 2011.] http://estimulaciontempranaenprimerainfancia.blogspot.com/2009/02/intervencion-temprana-en-ninos-con.html..

8. Estimualcion Temprana y Desarrollo Infantil. . [Online] [Cited: Marzo 5, 2011.] http://estimulacionydesarrollo.blogspot.com/..

9. Pedro Mestre. Proyecto Renacer Contigo. . Ciudad de la Habana : s.n.

10. Proyecto Renacer Contigo. . Ciudad de la Habana : s.n.

11. Pedro Luis Rodríguez. Historia de la Neurología en Cuba. . . Ciudad de la Habana. : s.n., 2008.

12. Pediatría. . [Online] [Cited: Febrero 11, 2011.] http://www.pediatraldia.cl/que_neurologia.htm.

13. Cosas de la Infancia. . [Online] [Cited: Febrero 10, 2011.] http://www.cosasdelainfancia.com/biblioteca-esti-t-g.htm.

14. REV CUBANA DE PEDIATRIA. . [Online] [Cited: Enero 10, 2011.] http://bvs.sld.cu/revistas/ped/vol68_2_96/ped11296.htm.

15. ¿Qué es la neurología? . [Online] [Cited: Enero 10, 2011.] http://svneurologia.org/pacientes/neurologia.htm.

16. Medina, Yordanis Tornés. Una alternativa para modelamiento de negocio con RUP.

17. Articulos sobre RUP. . [Online] [Cited: Febrero 11, 2011.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

18. Gallego, Juan Pablo Gómez. Fundamentos de la metodología RUP.

Bibliografía

19. Luis Carlos Díaz, Angela Carrillo, Deicy Alvarado. Análisis y Diseño Orientado a Objetos.

20. [Online] [Cited: Febrero 12, 2011.] http://www.slideshare.net/oscar8711/introduccin-a-rup-presentation..

21. [Online] [Cited: Febrero 11, 2011.] http://www.scribd.com/doc/44942492/Presentacion-Guia-RUP..

22. [Online] [Cited: Febrero 11, 2011.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf..

23. [Online] [Cited: Enero 12, 2011.] http://sites.google.com/site/flaviodanesse/aprendiendo-java/patrones-de.

24. [Online] [Cited: Enero 12, 2011.] http://www.mitecnologico.com/Main/ArquitecturaDelSoftware.

25. [Online] [Cited: Enero 10, 2011.] http://www.mitecnologico.com/Main/ArquitecturaDelSoftware.

26. Una introducción a Java y la IDE NetBeans. . [Online] [Cited: Enero 12, 2011.] http://www.whyfloss.com/pages/conference/static/editions/res07/charla1.pdf..

27. Taringa! [Online] [Cited: Enero 12, 2011.] http://www.taringa.net/posts/downloads/6774448/soft-para-programar-en-java-_-ejercicios-hechos-por-mi___.html.

28. Java Server Faces 2: Un vistazo. [Online] [Cited: Enero 12, 2011.] http://coresware.com/web/java-server-faces-2-un-vistazo/.

29. Adictos al Trabajo. . [Online] [Cited: Enero 12, 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro..

30. Junta de Andalucia. [Online] [Cited: Enero 12, 2011.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces..

31. Introducción a Ajax4jsf. . [Online] [Cited: Enero 12, 2011.] http://www.adictosaltrabajo.com/tutoriales/Ajax4Jsf.php.

32. Desarrollo en Web, Facelets y JSF – Uso de Templates. . [Online] [Cited: Enero 14, 2011.] http://blogs.antartec.com/desarrolloweb/2008/12/facelets-y-jsf-uso-de-templates..

33. Web Taller, Qué es XHTML. . [Online] [Cited: Enero 14, 2011.] http://www.webtaller.com/construccion/lenguajes/html/lecciones/que-es-xhtml.php..

34. Fap-Devel. . [Online] [Cited: Enero 14, 2011.] http://code.google.com/p/fap-devel/wiki/JBossSeam..

35. Slideshare.net. . [Online] [Cited: Enero 14, 2011.] http://www.slideshare.net/ingeniods/persistencia-de-objetos-con-hibernate..

36. Junta de Andalucia. . [Online] [Cited: Enero 14, 2011.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Hibernate.

Bibliografía

37. Curso de Java. [Online] [Cited: Enero 14, 2011.] http://www.scribd.com/doc/39973855/Curso-Java-J2EE-Completo.
38. Domotica. . [Online] [Cited: Enero 14, 2011.] http://www.domotica.us/JPA.
39. Curso de Java. . [Online] [Cited: Enero 14, 2011.] http://www.scribd.com/doc/39973855/Curso-Java-J2EE-Completo.
40. Taringa! . [Online] [Cited: Enero 14, 2011.] http://www.taringa.net/posts/info/6858927/UML_-Lenguaje-Unificado-de-Modelado.html..
41. Domotica. [Online] [Cited: Enero 14, 2011.] http://www.domotica.us/JBOSS.
42. SGBD: Sistemas Gestionadores de Base de Datos. . [Online] [Cited: Enero 15, 2011.] http://jorge613.wordpress.com/2010/05/27/sgbd-sistemas-gestionadores-de-base-de-datos/..
43. Curso Base de Datos PostgreSQL, SQL avanzado y PHP. . [Online] [Cited: Enero 15, 2011.] http://www.usabilidadweb.com.ar/postgre.php..
44. [Online] [Cited: Enero 15, 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/..
45. [Online] [Cited: Febrero 12, 2011.] http://www.buenastareas.com/ensayos/Pruebas-De-Software/1357326.html.
46. Introducción al entorno de desarrollo Eclipse. . [Online] [Cited: Enero 15, 2011.] http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf..
47. [Online] [Cited: Febrero 12, 2011.] http://www.buenastareas.com/temas/casos-de-pruebas-caja-blanca-software/40.
48. Maria Eugenia Arevalo's Blog. . [Online] [Cited: Enero 16, 2011.] http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/..
49. Programador PHP+AS. . [Online] [Cited: Enero 20, 2011.] http://www.sgmweb.es/modelo.asp..
50. TRABAJO DE INVESTIGACIÓN, ANALISIS Y DISEÑO DE SISTEMAS II. . [Online] [Cited: Enero 22, 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=6&ved=0CCoQFjAF&url=http%3A%2..
51. Monografias. com. [Online] [Cited: Marzo 16, 2011.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml.
52. [Online] [Cited: Marzo 16, 2011.] http://www3.uji.es/~mmarques/f47/apun/node32.html.

Bibliografía

53. Andrés, María Mercedes Marqués. . Arquitectura de los sistemas. . [Online] Diciembre 2, 2001. [Cited: Marzo 2, 2011.] http://www3.uji.es/~mmarques/f47/apun/node32.html..

54. Marqués Andrés, M. M. . Sistemas de Bases de Datos. [Online] [Cited: Marzo 16, 2011.] http://www3.uji.es/~mmarques/f47/apun/node32.html..

55. Información general sobre validación de datos. . [Online] [Cited: Marzo 18, 2011.] http://msdn.microsoft.com/es-es/library/kx9x2fsb%28VS.80%29.aspx.

56. [Online] [Cited: Marzo 16, 2011.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml#dpaquet.

57. Tutorial de Java-Arquitectura MVC. [Online] [Cited: Febrero 20, 2011.] http://sunsite.dcc.uchile.cl/java/docs/JavaTut/Apendice/arq_mvc.html.

58. Maldonado. Daniel M. Arquitectura de Programación en 3 Capas. . [Online] [Cited: Enero 20, 2011.] http://www.elcodigok.com.ar/2007/09/arquitectura-de-programacion-en-3-capas/ .

59. Plataforma J2EE. JBoss Seam Framework. [Online] [Cited: Enero 25, 2011.] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/ .

60. ¿Qué es JBoss SEAM? [Online] [Cited: Enero 27, 2011.] http://es.debugmodeon.com/articulo/que-es-jboss-seam.

61. Alvarez, Sara. Sistemas gestores de bases de datos. . [Online] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html .

62. Ramos, Ariel Delgado and María, Vidal Ledo. Informática en la salud pública cubana. . Ciudad de la Habana : s.n., 2006.

63. Pressman, Roger S. Un enfoque Práctico. .

64. Ivar Jacobson, Grady Booch y James Rumbaugh. El proceso Unificado de Desarrollo de Software.

GLOSARIO DE TÉRMINOS.

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

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

Artefacto: Conjunto de documentos, diagramas, archivos, etc., obtenidos o resultantes en cada flujo de trabajo y generados por los trabajadores.

Bean: Es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno. Se puede decir que existen con la finalidad de ahorrar tiempo al programar.

Bytecode: Es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador, cuyo contenido es el código objeto o código máquina.

Camel Casing: Especifica que la palabra de inicio del identificador comienza con minúscula. Si el identificador está compuesto por más de una palabra, entonces estás deben comenzar con mayúscula.

Case sensitive: Es una expresión usada en lenguaje informático, que se aplica a los textos en los que tiene alguna relevancia escribir un carácter en mayúsculas o minúsculas.

Caso de Uso: Es parte del análisis y describe lo que el sistema debe hacer desde el punto de vista del usuario. Es decir, describen un uso del sistema y cómo este interactúa con el usuario.

Confidencialidad: Es la propiedad de la información, por la que se garantiza que está accesible únicamente a personal autorizado a acceder a dicha información.

CRUD: Es el acrónimo de Crear, Obtener, Actualizar y Borrar (Create, Retrieve, Update y Delete, por sus siglas en inglés). Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un sistema de software.

Disponibilidad: Es un protocolo de diseño del sistema y su implementación asociada, que asegura un cierto grado absoluto de continuidad operacional, durante un período de medición dado. Se refiere a la habilidad de la comunidad de usuarios para acceder al sistema, someter nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder al sistema, se dice que está no disponible.

Driver: Es un software o programa, que sirve de intermediario entre un dispositivo de hardware y el sistema operativo. Su finalidad es la de permitir y extraer el máximo de las funcionalidades del dispositivo para el cual ha sido diseñado.

E-bussiness: Se refiere al conjunto de actividades y prácticas de gestión empresarial resultante de la incorporación a los negocios de las tecnologías de la información y la comunicación (TIC), general y particularmente de Internet, así como, a la nueva configuración descentralizada de las organizaciones y su adaptación a las características de la nueva economía.

Fichero: Es el soporte en el que se encuentran almacenados o registrados, los datos de carácter personal.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

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

Identación: Significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre, como sangrado, o sangría.

Integridad: Es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. La violación de la integridad se presenta cuando se accede a un programa o proceso (por accidente o con mala intención) se modifica o borra los datos importantes que son parte de la información, así mismo hace que su contenido permanezca inalterado, a menos que sea modificado por personal autorizado y esta modificación sea registrada, asegurando su precisión y confiabilidad.

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

Máquina Virtual: Es un software que emula a una computadora y puede ejecutar programas como si fuese una computadora real.

Modelo-Vista-Controlador: Es un patrón de arquitectura de software compuesto de tres componentes distintos: datos, interfaz de usuario, y lógica del negocio.

Paquete: Es una serie de programas que se distribuyen conjuntamente. Algunas de las razones suelen ser que el funcionamiento de cada uno complementa, o requiere de otros.

Pascal Casing: Especifica que la palabra de inicio del identificador comienza con mayúscula. Si el identificador está compuesto por más de una palabra, entonces estás deben comenzar con mayúscula.

Patrones GRASP: En diseño orientado a objetos, GRASP, son patrones generales de software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Se pueden destacar cinco patrones principales: experto, creador, alta cohesión, bajo acoplamiento, controlador.

Peso liviano: Inicialmente, los métodos ágiles fueron llamados métodos de "peso liviano".

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

Proceso: Un proceso puede ser definido como un conjunto de actividades interrelacionadas entre sí, que a partir de una o varias entradas de materiales o información, dan lugar a una o varias salidas también de materiales o información, con valor añadido.

Proceso de negocio: Es una colección de actividades que toma uno o varios tipos de entrada y crea una salida que tiene valor para el cliente.

Pruebas Unitarias: Consisten en probar o testear piezas de software pequeñas; a nivel de secciones, procedimientos, funciones y módulos; aquellas que tengan funcionalidades específicas. Dichas pruebas se utilizan para asegurar el correcto funcionamiento de secciones de código, mucho más reducidas que el conjunto, y que tienen funciones concretas con cierto grado de independencia. Es una manera de probar el correcto funcionamiento de un módulo de código.

Release: Una versión candidata a definitiva, o candidata para el lanzamiento, aunque más conocida por su nombre en inglés *release candidate*, comprende un producto final, preparado para publicarse como versión definitiva a menos que aparezcan errores que lo impidan. En esta fase el producto implementa todas las funciones del diseño y se encuentra libre de cualquier error que suponga un punto muerto en el desarrollo. Muchas empresas de desarrollo utilizan frecuentemente este término. Otros términos relacionados incluyen gamma, delta (y tal vez más letras griegas), para versiones que están prácticamente completas pero todavía en pruebas; y omega para versiones que se creen libres de errores y se hallan en

el proceso final de pruebas. Gamma, delta y omega son, respectivamente, la tercera, cuarta y última letras del alfabeto griego.

Rendimiento del software: Son las pruebas que se realizan, desde una perspectiva, para determinar lo rápido que realiza una tarea, un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza por mejorar el rendimiento, englobándose en el diseño y la arquitectura de un sistema, antes incluso del esfuerzo inicial de la codificación.

Sistema de Gestión de Base de Datos: Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Subproceso: Son partes bien definidas en un proceso. Su identificación puede resultar útil para aislar los problemas que pueden presentarse y posibilitar diferentes tratamientos dentro de un mismo proceso.

Subsistema: Es un sistema que se ejecuta sobre un sistema operativo, este puede ser un shell (intérprete de comandos) del sistema operativo primario o puede ser una máquina virtual.

Stubs: Es un artículo que contiene solo unas pocas frases del texto y además proporciona información útil, pero es demasiado corta para proporcionar una cobertura enciclopédica de un tema, pero tiene capacidad de expansión.

Testeado: Del inglés Test (Prueba).

Trabajador del Negocio: Un trabajador del negocio representa un rol que juega una persona (o grupo de personas), una máquina o un sistema automatizado; actuando en el negocio. Son los que realizan las actividades, interactuando con otros trabajadores del negocio y manipulando entidades.

Underscoard: Carácter ASCII representado como "_".

Usabilidad: Es la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos, con el fin de alcanzar un objetivo concreto. La usabilidad también puede referirse al estudio de los principios que hay tras la eficacia percibida de un objeto.

XUL: Acrónimo de ***X****ML-based* ***U****ser-interface* ***L****anguage*, lenguaje basado en XML para la interfaz de usuario, es la aplicación de XML a la descripción de la interfaz de usuario en el navegador Mozilla.