Universidad de las Ciencias Informáticas

Facultad 7

Implementación de los procesos: realizar evaluaciones y crear resumen del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños

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

Autoras: Yisel Morera Rivera

Patricia Gómez Mazorra

Tutores: Ing. Runer Céspedes Aldana

Ing. Solainy Fajardo Araujo

La Habana, 2011

"Año 53 de la Revolución"

DATOS DE CONTACTO

Ing. Runer Céspedes Aldana. Graduado 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 profesor, ha impartido las materias Programación II y IV. Actualmente imparte la asignatura de Programación II, de la cual es responsable en la Facultad 7 de esta universidad. En la vinculación a la producción pertenece al Departamento de Sistemas Especializados en Salud (SES) del Centro de Informática Médica (CESIM), donde se desempeña como jefe del proyecto informático Prótesis.

Ing. Solainy Fajardo Araujo. Graduada de Ingeniería en Ciencias Informáticas en la UCI en el 2009. Instructora recién graduada en adiestramiento. Durante su trabajo como profesora ha impartido la asignatura de Matemática Discreta I. Actualmente, imparte Matemática Discreta II y se desempeña como tutora de estudiantes en la asignatura de Práctica Profesional. En la vinculación a la producción, pertenece al Departamento SES del CESIM donde se desempeña como analista principal.

RESUMEN

En el Hospital Pediátrico William Soler se lleva a cabo un proyecto de atención y evaluación del neurodesarrollo en niños llamado Renacer Contigo, compuesto por cinco especialidades: Neurología, Nutrición, Fisiatría, Psicología y Logopedia. Las pruebas aplicadas son muy extensas, se registran de forma manual y requieren de mucho tiempo, lo cual impide la atención de mayor cantidad de pacientes el mismo día.

La Universidad de las Ciencias Informáticas, específicamente la Facultad 7, se ha vinculado con este hospital, con el objetivo de informatizar todos los procesos llevados a cabo en la aplicación de dicho proyecto, a través de la elaboración del Sistema de Evaluación del Neurodesarrollo en Niños. Desde el curso 2009-2010 se diseñó el Módulo Logopedia de este sistema informático, constituyendo el mismo una solución no funcional.

El presente trabajo tiene como objetivo implementar los procesos realizar evaluaciones y crear resumen, apoyados en los flujos de trabajo Implementación y Prueba. Para el desarrollo del Módulo se emplean como herramientas: el Gestor de Bases de Datos PostgreSQL 8.3, el lenguaje de programación Java, la metodología de desarrollo de software el Proceso Unificado de Desarrollo. Se hace uso del Lenguaje Unificado de Modelado y Visual Paradigm para la creación de los artefactos, el framework Jboss Seam para la lógica del negocio y el patrón Modelo-Vista-Controlador en la arquitectura.

El despliegue de esta aplicación informática, permitirá agilizar la realización de las pruebas y el resumen de la consulta, logrando una mayor eficiencia en la gestión de la información.

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), es uno de los elementos característicos de la sociedad actual; lo que ha provocado una explosión en la capacidad de procesar y almacenar grandes volúmenes de información. Con el desarrollo de estas tecnologías la forma en que operan las organizaciones modernas ha cambiado, pues a través de su uso, se logran importantes mejoras, automatizan los procesos operativos, suministran una plataforma de información necesaria para la toma de decisiones en todas las esferas y constituyen el núcleo central de una transformación multidimensional, que experimenta la economía y la sociedad a nivel mundial.

La capacidad de las TIC para reducir muchos obstáculos tradicionales, posibilita el uso del potencial de estas tecnologías en beneficio de millones de personas en todo el mundo. Estas han sido conceptualizadas como la integración y convergencia de la computación, las telecomunicaciones y las técnicas para el procesamiento de datos. Se han introducido en los más disímiles campos, como el de la medicina, dando lugar a la Informática Médica. Su utilización permite la optimización de todos los procesos con la finalidad de prestar mejores servicios de salud, contribuyendo de esta forma, a lograr mejoras en la salud individual y colectiva y una mejor calidad de vida de la población.

La informatización de la sociedad se define en Cuba como el proceso de utilización ordenada y masiva de las TIC, para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. Este proceso busca lograr mayor eficacia y eficiencia, que permitan una mayor generación de riquezas y hagan sustentable el aumento sistemático de la calidad de vida de los cubanos. La Revolución cubana se encuentra en un proceso continuo de cambios, pues como pronunciara Fidel Castro en su discurso del 1ro de mayo del 2000:“Revolución es sentido del momento histórico; es cambiar todo lo que debe ser cambiado…; es emanciparnos por nosotros mismos y con nuestros propios esfuerzos (…)” (1)

Esta transformación se realiza en todas las ramas de la sociedad y el Ministerio de Salud Pública (Minsap), es uno de los mayores beneficiados, por lo que se está llevando a cabo un proceso de automatización y mejoras tecnológicas con el objetivo de elevar la calidad de todos los servicios de salud en el país. Solo después del triunfo de la Revolución y la constitución de una Sociedad Socialista, en la que nada es más importante que el hombre y su realización social, se organizan con efectividad los servicios médicos

especializados para el pueblo; ya que como expresara el Comandante en Jefe: “Una profunda revolución en los servicios de salud tendrá lugar en nuestra Patria”. (2)

La Universidad de las Ciencias Informáticas (UCI) juega un papel fundamental en este proceso, pues constituye un eslabón primordial en el desarrollo de la Industria Cubana del Software. El Centro de Informática Médica (CESIM) perteneciente a la Facultad 7 de esta universidad, es el encargado de apoyar la elaboración de sistemas informáticos que agilizarán los procesos desarrollados en diferentes unidades del Sistema Nacional de Salud (SNS).

En Cuba, el SNS cuenta con varias instituciones especializadas que juegan un papel fundamental en el desarrollo y calidad de vida de los pacientes que atienden, así surgen las Unidades de Cuidados Intensivos Pediátricos (UCIP), con el objetivo de dar respuesta asistencial eficiente a las urgencias pediátricas, siendo el servicio hospitalario dedicado a la asistencia intensiva integral y continuada al niño críticamente enfermo y como expone el máximo líder de la Revolución: “Debemos pensar en los niños de hoy, que son el pueblo de mañana. Hay que cuidarlos y velar por ellos como los pilares con que se funda toda obra verdaderamente hermosa y útil”. (3)

En el Hospital Pediátrico William Soler no solo se brinda una atención especializada al niño grave, sino que se trabaja para lograr una calidad de vida óptima en los afectados, mediante el programa “Renacer Contigo”. Este programa tiene como metas: evaluar la calidad del neurodesarrollo de los niños de cero a cinco años de edad, egresados de las Unidades de Terapia Intensiva Polivalente y Neonatal; determinar grupos de tratamiento según la edad neurológica del paciente; valorar la efectividad del Programa de Atención Temprana y dar respuesta a las necesidades transitorias o permanentes que presentan los mismos.

Durante la aplicación de este programa, la evaluación se realizará por un equipo interdisciplinario compuesto por especialistas en: fisiatría, neurología, nutrición, neurofisiología, psicología y logopedia, quienes determinan la presencia o no, de afecciones del neurodesarrollo tras el egreso de terapia intensiva.

La Logopedia es la rama que trata la comunicación oral, la cual se divide en tres niveles: lenguaje, habla y voz. En cada nivel existen patologías propias, las que son tratadas en las consultas del programa mediante pruebas de evaluación del neurodesarrollo.

Las pruebas de evaluación del neurodesarrollo aplicadas por los logopedas son muy extensas y las mismas

se registran de forma manual, por lo que dificulta la comparación de diagnósticos emitidos en diferentes consultas y propicia la pérdida o deterioro de la información. Otras problemáticas que se confrontan, son los retrasos en la obtención de información y en la generación de datos estadísticos. Además, el evaluador debe consultar tablas matemáticas demasiado voluminosas y complejas, lo que requiere de mucho tiempo, e impiden atender a gran cantidad de niños el mismo día.

Con el objetivo de solucionar esta problemática, surge el Sistema de Evaluación del Neurodesarrollo en Niños (SENDN), el mismo se encarga de informatizar los procesos anteriormente expuestos. Este sistema se ha desarrollado desde el curso 2009-2010, obteniéndose como resultado el diseño del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños; pero este diseño no es funcional, por lo que actualmente se hace necesario dar continuidad al desarrollo de los procesos: realizar evaluaciones y crear resumen.

Teniendo en cuenta la situación problemática anteriormente referenciada y atendiendo a las necesidades actuales relacionadas con esta temática, se identifica el siguiente **problema a resolver**: ¿cómo hacer funcional el diseño de los procesos: realizar evaluaciones y crear resumen del Módulo de Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños?

Se plantea como **objeto de estudio**: proceso de desarrollo de software del Módulo Logopedia del SENDN.

Como **campo de acción** se ha considerado: implementación y pruebas de los procesos: realizar evaluaciones y crear resumen del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños.

En calidad de **objetivo general**, para brindar solución al problema planteado se establece: implementar los procesos realizar evaluaciones y crear resumen del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños, a partir del diseño propuesto por el analista.

Para dar cumplimiento al objetivo general se proponen las siguientes **tareas de investigación:**

- Analizar la metodología, plataforma, tecnologías, librerías y herramientas, definidas por el Departamento de Gestión Hospitalaria del Centro de Informática Médica.
- Estudiar el Proceso Unificado de Desarrollo de Software.

- Describir la arquitectura propuesta por el Departamento de Gestión Hospitalaria del Centro de Informática Médica.
- Valorar críticamente y refinar el diseño propuesto por el analista.
- Generar los artefactos correspondientes a los flujos de trabajo: “Implementación” y “Pruebas”.
- Implementar las funcionalidades de los procesos: realizar evaluaciones, obtener resultados de consulta y crear resumen, definidos en el diseño del Módulo Logopedia, del Sistema de Evaluación del Neurodesarrollo en Niños.
- Validar las funcionalidades del sistema, a través de la realización de los casos de prueba.

Con el despliegue de la aplicación se obtendrán los siguientes beneficios:

- Se minimizará el tiempo de consulta, por lo que se podrán atender más pacientes en el mismo día.
- El paciente será el centro de atención durante la realización de la consulta.
- Facilitará que el programa de evaluación y atención temprana, se pueda realizar en todos los hospitales pediátricos de Cuba.
- La información será almacenada de forma segura y podrá ser consultada con mayor rapidez y eficiencia.
- La propuesta que se realiza representa un importante valor añadido para los servicios de salud que se prestan a los ciudadanos, al existir una mayor eficacia en la gestión de la información generada en las consultas.
- Permitirá un avance en la organización de la información necesaria para garantizar el seguimiento del estado de salud de cada paciente.
- Posibilitará un ahorro notable de recursos materiales.

El trabajo se ha estructurado en cuatro capítulos, como a continuación se describen:

Capítulo 1: Fundamentación teórica del procesos de desarrollo de software del Módulo de Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños. En este capítulo se realiza una descripción del SENDN y de su Módulo de Logopedia, así como un análisis de los procesos que se llevan a cabo en el mismo. Se describe el Proceso Unificado de Desarrollo de software. Se efectúa un análisis de

los flujos de trabajo desarrollados por el analista en el período 2009-2010. Se analizan los flujos de trabajo de Implementación y Prueba. Se enuncian las tecnologías, metodologías y herramientas definidas por el Departamento de Gestión Hospitalaria del Centro de Informática Médica, que serán utilizadas para en el desarrollo del módulo de Logopedia del SENDN.

Capítulo 2: Descripción de la arquitectura de software del Sistema de Evaluación del Neurodesarrollo en Niños. En este capítulo se detallan los requisitos no funcionales y de la arquitectura; además de las estrategias de codificación, los estándares y estilos a utilizar en el desarrollo de la aplicación. Se explica cómo lograr la seguridad del SENDN.

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

Capítulo 4: Validación de la solución propuesta. En este capítulo se diseñan los casos de pruebas y se detallan las características de los mismos. Se analiza el procedimiento de ejecución de las pruebas de caja negra y se puntualiza el método a utilizar en el Módulo Logopedia del SENDN.

CAPÍTULO 1. Fundamentación teórica del proceso de desarrollo de software del Módulo de Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños.

En el presente capítulo se describe el marco conceptual que será empleado en la investigación para lograr un mayor entendimiento de la misma. Se realiza una reseña del Sistema de Evaluación del Neurodesarrollo en Niños, y la necesidad de su creación. Se detallan los procesos realizados en el Módulo Logopedia. Se analiza la metodología de desarrollo de software utilizada para la elaboración del sistema, analizando las actividades, trabajadores, artefactos, fases y flujos de trabajo realizados por el analista que anteceden a los de implementación y prueba. Se exponen y explican los lenguajes de programación, herramientas, y tecnologías a utilizar, en el desarrollo del sistema.

1.1 Marco Conceptual.

Lenguaje: conjunto de sonidos y palabras con que se expresa el pensamiento. Forma de comunicar el pensamiento.

Discapacidad: es toda restricción o ausencia, (debido a una deficiencia), de la capacidad de realizar una actividad en la forma, o dentro del nivel normal, para su edad, sexo y condiciones socioculturales. Representa la objetivación de una deficiencia y refleja alteraciones en el ámbito de la persona.

Neurodesarrollo: adquisición de funciones dependientes del sistema nervioso, que implican un incremento de estructuras orgánicas y funcionales a través de un proceso de maduración.

Equipo interdisciplinario: grupo formado por profesionales de distintas disciplinas, en el que existe un espacio formal para compartir la información, las decisiones se toman a partir de la misma y se tienen objetivos comunes.

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, sustituyendo la carga de una vida inútil por la alegría de una existencia útil y transformando los sentimientos de agresividad, indiferencia, o rechazo, en solidaridad, colaboración y esperanza.

Intervención: tomar parte en un asunto.

Necesidad transitoria: afectación que posee un individuo, que de ser tratada adecuadamente, puede desaparecer sin dejar secuelas.

Necesidad permanente: afectación que posee un individuo, que permanecerá a lo largo de su vida.

Nivel de edad basal: primer nivel en el cual, el niño obtenga una puntuación de "dos" en todos los ítems, o el nivel por debajo del cual, obtenga "cero o uno", en uno solo de los ítems y "dos", en el resto.

Nivel de edad superior: nivel en el cual el niño puntee "cero o uno" en todos los ítems, o "dos" en un ítem y "cero o uno", en el resto de los ítems.

Prueba de Pesquizaje de Desarrollo del Lenguaje (PPDL): examen que puede ser aplicado a niños cuya edad cronológica sea de 0 a 36 meses. Evalúa el retardo en el desarrollo del lenguaje.

Peabody Picture Vocabulary (PPV, por sus siglas en inglés): examen que puede ser aplicado a niños cuya edad cronológica sea mayor a los 30 meses. Evalúa el vocabulario receptivo.

Inventario de las Primeras Palabras (IPP): examen que puede ser aplicado a niños cuya edad cronológica sea de 12 a 36 meses. Evalúa el desarrollo del vocabulario.

1.2 Descripción del Sistema de Evaluación del Neurodesarrollo en Niños.

El Grupo Interdisciplinario de Atención Temprana del Hospital William Soler, ha investigado los resultados de la evaluación e intervención del neurodesarrollo del niño egresado de terapia intensiva, reportado como crítico durante su estadía, con ánimos de disminuir las discapacidades en la infancia; pues el niño sometido a una enfermedad grave es propenso a que se comprometa su calidad de vida. El resultado de la investigación realizada, será materializado en el Programa Renacer Contigo. La información y las acciones relacionadas con su aplicación son complejas, por lo que se decide automatizarlas, conformando de esta manera, el Sistema de Evaluación del Neurodesarrollo en Niños.

El sistema cuenta con siete módulos:

- Expediente y Turnera: encargado de gestionar los turnos asignados a los pacientes.
- Fisiatría: se especializa en el manejo de la información relacionada con la exploración neurológica, la indagación de la esfera motora y la aplicación del Peabody Motor fino y grueso.
- Neurología: permite el procesamiento de los resultados generados en la aplicación del examen físico neurológico y la elaboración del expediente del paciente.
- Nutrición: posibilita a la especialista el procesamiento de la información concebida a través del examen físico, además de facilitar el análisis de la evolución antropométrica y dietética.

- Neurofisiología: procesa la información extraída de los estudios neurofisiológicos.
- Psicología: facilita la evaluación del desarrollo psicomotor, y la aplicación del test de Bayley y el test de Therman Merrill.
- Logopedia: genera resultados a partir de las pruebas aplicadas, asigna clasificaciones según el nivel de discapacidad de los pacientes, y permite la creación de la Historia Clínica Logofoniátrica.

1.3 Descripción del Módulo de Logopedia del SENDN.

"La especialidad de Logopedia, constituye la rama de la medicina que trata la etiología, la patogenia, la sintomatología, la evolución, el diagnóstico, el tratamiento y la profilaxis, de las afecciones de la comunicación verbovocal". (4)

Durante la realización de la consulta, si el paciente llega por primera vez, se le realiza la Historia Clínica logopédica que incluye:

ExamenPsico-físico: analiza la anatomía y fisiología de los órganos del lenguaje y la audición. Se lleva a cabo un estudio de la voz, la respiración-tono, la amplitud ante el sonido y la comunicación familiar.

Examen Logofoniátrico: Mide los niveles de comunicación oral:

- Nivel I: Lenguaje. A través del interrogatorio se comprobará: Codificación o Decodificación.
- Nivel II: Habla.

  Lenguaje expresivo.

  Lenguaje receptivo.
- Nivel III: Voz (timbre, tono, intensidad, resonancia).
- Examen Laringoscópico.

Con el propósito de obtener la edad de vocabulario y de medir el nivel de desarrollo del niño se aplicarán las escalas siguientes:

Prueba de Pesquisaje del Desarrollo del Lenguaje: permite evaluar a todos los niños desde su nacimiento hasta los 36 meses de edad (3 años); además, cualquier niño con dificultades cuyo desarrollo del lenguaje esté por debajo de los 36 meses, en un tiempo de tres minutos como máximo. Consta de 41 ítems, divididos en tres áreas: auditivo expresivo, auditivo receptivo y visual. Cada ítem está representado en la hoja de

respuesta por una barra horizontal. Las partes sombreadas corresponden a los valores percentiles para la edad de la emergencia de cada ítem: sin sombra, percentiles 25 y 5; rayado: percentiles 50 y 75 y negro: percentiles 75 y 90.Criterio de éxito: vence o no vence. Hay éxito en el PPDL solo si se "pasa" en las tres áreas. (Ver Anexo 1)

Peabody Picture Vocabulary Test: propuesto por Lloyd Dunn en 1965, que evalúa el vocabulario receptivo. En esta propuesta, se utiliza la versión desarrollada por el Centro de Neurociencias de Cuba y la Facultad de Psicología. Consiste en 150 láminas con 4 figuras cada una y una palabra, para la cual el sujeto debe seleccionar la figura adecuada, obteniéndose un Coeficiente de Inteligencia que puede variar desde Retraso Mental no clasificado, hasta normal alto. Es válido para niños de edades comprendidas entre 2 y 17 años y brinda información acerca de la edad de vocabulario, el Coeficiente de Inteligencia y el Percentil de Ejecución. Ofrece muchas ventajas, pues no es necesario que los sujetos sepan leer, no son obligatorias las respuestas orales ni escritas, a la vez que el rango de edad que cubre, es bastante amplio. (Ver Anexo 2).

El *resumen de la consulta,* se realiza luego de aplicados los exámenes. En él, se plasman los resultados de los exámenes complementarios indicados anteriormente, los obtenidos durante la aplicación de las pruebas, así como la impresión diagnóstica, la cual es discutida por todo el equipo interdisciplinario de especialistas.

Este módulo en el SENDN, gestiona toda la información relacionada con esta especialidad. Su desarrollo inició en el período 2009-2010, con la realización de los flujos Modelamiento del Negocio, donde se identificaron los procesos de negocio: realizar consulta de logopedia y realizar reconsulta de logopedia; se determinaron los requerimientos funcionales con los que el sistema deberá cumplir:

- **RF1:** Realizar Evaluación Logofoniátrica.
- **RF2:** Visualizar detalles Evaluación Logofoniátrica.
- **RF3:** Modificar Evaluación Logofoniátrica.
- **RF 4:** Buscar pacientes atendidos Logopedia.
- **RF 5:** Realizar Resumen Logopedia.
- **RF 6:** Visualizar Detalles Resumen Logopedia.
- **RF7:** Modificar Resumen Logopedia.

- **RF8:** Visualizar detalles test PPDL.
- **RF9:** Realizar Test PPDL.
- **RF10:** Realizar Test PPV.
- **RF11:** Visualizar detalles test PPV.
- **RF12:** Buscar Resultado Consulta Logopedia.
- **RF13:** Visualizar Consulta Logopedia.
- **RF14:** Buscar Resumen Logopedia.
- **RF15:** Visualizar Resumen Logopedia.

Se realizó el Modelo de Clases del diseño, este posee una gran importancia pues describe la realización de los casos de usos, y sirve como una abstracción del Modelo de Implementación y el código fuente. Es usado como una entrada inicial en las actividades de implementación y prueba.

En el presente trabajo de diploma, se desarrollarán los flujos de trabajo de *Implementación* y *Prueba* como continuidad al ciclo de vida del software.

1.4 Metodología de desarrollo de Software.

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevas aplicaciones de software. En un proyecto de desarrollo, define quién debe hacer qué, cuándo y cómo hacerlo. Es un proceso, puede seguir uno o varios modelos de ciclo de vida, indica cómo hay que obtener los distintos productos parciales y finales en el desarrollo de un software.

Está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo. Se basan en una combinación de los modelos de procesos genéricos (cascada, incremental y evolutivo).

En el desarrollo del presente trabajo se utiliza el Proceso Unificado de Modelado(RUP, por sus siglas en inglés) como metodología, pues se encuentra predefinida por el Departamento de Gestión Hospitalaria,

debido que el SENDN forma parte del Sistema de Información Hospitalaria (HIS, por sus siglas en inglés) y adopta toda la arquitectura y herramientas especificadas por este.

El Proceso Unificado de Modelado, se refiere a un proceso de desarrollo de software. “No es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. Unifica los mejores elementos de las metodologías que le precedieron”. (5)

El ciclo de vida de RUP se caracteriza por ser:

- **Dirigido por casos de uso:** en RUP los Casos de Uso, no son solo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso, constituyen un elemento integrador y una guía de trabajo. 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.
- **Centrado en la arquitectura:** “presta especial atención al establecimiento temprano de una buena arquitectura que no se encuentre fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento. Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura, debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura, como Casos de Uso, deban evolucionar en paralelo durante todo el proceso de desarrollo de software”. (6)(sic)
- **Iterativo e incremental**: el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo que el equilibrio entre Casos de Uso y arquitectura, se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales), del cual se obtiene un incremento que produce un crecimiento en el producto.

RUP agrupa las actividades en grupos lógicos, definiéndose nueve flujos de trabajo principales (los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo), basándose en el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés).

- Modelamiento del negocio: describe los procesos de negocio, identificando quiénes participan y 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 de los componentes en ellos y la estructura de capas de la aplicación.
- Prueba: busca los defectos a lo largo del ciclo de vida del software.
- Instalación: produce el despliegue del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- Administración del proyecto: involucra actividades con las que se busca elaborar 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/actualización concurrente de elementos, control de versiones, etc.
- 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.

RUP divide el proceso de desarrollo en cuatro fases, dentro de las cuales, se realizan varias iteraciones en número variable y en las que se hace un mayor o menor hincapié en los distintas actividades.

Durante la fase de inicio, las iteraciones ponen mayor énfasis en las actividades de modelado del negocio y de requisitos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de la implementación orientado a la línea de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración, se seleccionan algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se ejecutan tantas iteraciones, hasta que termine la implementación de la nueva versión del producto.

En la fase de transición, se pretende garantizar que se posee un producto preparado para su entrega a la comunidad de usuarios.

De forma general, el objetivo fundamental de RUP es producir software de alta calidad, que cumpla con los requerimientos de los usuarios dentro de la planificación y presupuesto establecidos.

Como RUP es un proceso, en su modelación define como sus principales elementos: los roles, que responden a la pregunta ¿quién?; las actividades que responden a la pregunta ¿cómo?; los artefactos, que responden a la pregunta ¿qué? y los flujos de trabajo, que responde a la pregunta ¿cuándo?

Descripción del Flujo de Trabajo de Implementación.

En este flujo de trabajo, se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y otros. Se deben realizar las pruebas de unidad: cada implementador es responsable de probar cada una de las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración se ejecutarán las siguientes actividades:

- Planear 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 analizan los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.

Los propósitos fundamentales de este flujo son:

- Planificar las integraciones de sistema necesarias en cada iteración.
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.
- Implementar las clases y subsistemas encontrados durante el diseño.

- Probar los componentes individualmente, y a continuación integrarlos, compilándolos y enlazándolos en uno o más ejecutables.

Intervienen los siguientes trabajadores:

- Arquitecto de Software (genera el Modelo de Implementación).
- Implementador (crea los artefactos: elementos de implementación, subsistema de implementación y elementos de pruebas).
- Integrador (compone el artefacto: Plan de Integración).
- Revisor Técnico (revisa los artefactos generados en este flujo).

En este flujo se crea un artefacto de gran importancia en el ciclo de vida del software, el *Modelo de Implementación*, que consiste en una visión general de lo que tiene que ser implementado, un apartado para cada iteración con los componentes y subsistemas a implementar durante esa iteración, así como los resultados software, que se han de obtener y las pruebas que se han de realizar sobre ellos.

También se elaboran:

- Diagrama de Componentes: incluye a los componentes y archivos que se utilizan para ensamblar y hacer disponible el sistema físico.
- Diagrama de Despliegue: contiene los nodos que forman la topología hardware, sobre la que se ejecuta el sistema y la distribución de las partes del sistema en ellos.

Esta es una de las fases más técnicas del sistema, en la cual hay una interacción y comunicación permanente entre el equipo de analistas, diseñadores y el equipo de programadores. La participación del usuario es mínima, estando limitada a participar en las pruebas de programas o módulos.
Un aspecto muy importante a considerar, es la definición de un estándar de codificación de los programas, esto comprende: la estructuración del programa, funciones, definición de clases, métodos, definición de variables globales, definición de variables locales, tablas internas, tablas temporales, uso de encabezados en los programas, documentación interna de programas, etc. En este flujo de trabajo, se desempeña el rol de implementador.

Descripción del Flujo de Trabajo Prueba.

El desarrollo del software implica una serie de actividades de producción en las que las posibilidades de existencia de fallos humanos aumentan. Los errores pueden aparecer desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea e imperfecta; así como en los posteriores pasos del diseño y desarrollo.

"Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente". (7) Constituye 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.

Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad, pues los exámenes realizados y los resultados de los mismos son proporcionados por el sistema propuesto en la investigación; se procesan datos de pacientes reales y en caso de estar errados, se estaría atentando contra la vida de algún ser humano.

Durante este flujo de trabajo se realizan las siguientes actividades:

- Definir la misión de la evaluación.
- Verificar el enfoque de prueba.
- Validar la estabilidad del build.
- Probar y evaluar.
- Lograr la misión aceptable.
- Mejorar la calidad de las pruebas.

Intervienen los siguientes trabajadores:

- Administrador de Prueba: es el responsable del éxito de la prueba. Este rol involucra la gestión de la calidad, planificación, 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 y evaluando la calidad total experimentada como un resultado de las actividades. Este rol lleva la responsabilidad para representar apropiadamente las necesidades de los stakeholder 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. El rol incluye la identificación de técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas.
- Probador: es el responsable durante las actividades principales de las pruebas, el cual incluye la conducción de las pruebas necesarias y el registro del resultado de la prueba.

En el desarrollo de este flujo se desempeña el rol de probador y se aplicarán las pruebas de caja negra para comprobar la operatividad de las funcionalidades. No se realizará la liberación del producto, se efectuarán solamente pruebas internas al software.

1.5 Herramientas y Tecnologías.

En el proceso de desarrollo e implementación de las funcionalidades para el Sistema de Evaluación del Neurodesarrollo en Niños, se utilizan diversas herramientas que apoyarán su desarrollo rápido, seguro y eficiente. Estas herramientas fueron definidas previamente por el Departamento de Gestión Hospitalaria del Centro de Informática Médica, a continuación se realiza su descripción:

Java Platform Enterprise Edition (JavaEE 5)," es un estándar del sector para desarrollar aplicaciones Java portátiles, robustas, escalables y seguras para el servidor". (8)

"La plataforma Java constituye un conjunto de especificaciones que facilitan el desarrollo y despliegue de aplicaciones empresariales multicapa. Ofrece un conjunto de especificaciones y técnicas que proporcionan soluciones completas, seguras, estables y escalables para el desarrollo, despliegue y gestión, de aplicaciones de múltiples niveles de funcionalidad basadas en servidores. Se reduce el costo y complejidad de desarrollo, lo cual resulta en servicios que se pueden desplegar y extender fácilmente". (9)

Java EE posee una arquitectura distribuida y una definición estandarizada de componentes, contenedores y servicios, que permiten crear y desplegar aplicaciones distribuidas, en una arquitectura multicapa.

El referido estándar, incluye una suite de especificaciones para la Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés), tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc. y define cómo coordinarlos. Estas incluyen: Enterprise JavaBeans, Servlets, Portlets (siguiendo la especificación de Portlets Java), Java Server Pages y varias tecnologías de servicios web, que reducen el tiempo de desarrollo y la complejidad, al tiempo que mejoran el rendimiento de las aplicaciones.

Java Runtime Environment (JRE, por sus siglas en inglés), constituye un conjunto de herramientas necesarias para la ejecución del código y el correcto funcionamiento de cualquier aplicación que haya sido desarrollada a partir del lenguaje Java.

El entorno en tiempo de ejecución de Java, está conformado por una Máquina Virtual de Java o Java Virtual Machine (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.

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

La Máquina Virtual de Java ejecuta el código resultante de la compilación del código fuente y se encarga de traducir el bytecode en instrucciones nativas de la plataforma destino. Esto permite que una misma aplicación Java, pueda ser ejecutada en una gran variedad de sistemas con arquitecturas distintas, siempre que posea una implementación adecuada de la JVM.

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, denominado Java Development Kit (JDK, por sus siglas en inglés), que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java. Es compatible con la mayoría de los navegadores web, incluyendo a Internet Explorer, Mozilla Firefox y Netscape Navigator, debido a la extensión que se instala con el paquete.

Java Server Faces (JSF, por sus siglas en inglés) es un framework para aplicaciones web, basadas en Java 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. Es una tecnología que se ejecuta del lado del servidor. La interfaz de usuario es tratada como un conjunto de componentes. Permite desarrollar rápidamente aplicaciones de negocio dinámicas, creando páginas para las vistas muy sencillas.

JSF incluye varios elementos como:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario, incluyendo los elementos estándares del Lenguaje de Marcado de Hipertexto (HTML, por sus siglas en inglés), para representar un formulario.
- Dos librerías de etiquetas personalizadas para Java Server Pages, que permiten expresar una interfaz Java Server Faces dentro de una página JSP.

"RichFaces, es un framework de código abierto y una librería de componentes visuales que añade capacidad Ajax dentro de aplicaciones JSF existentes sin recurrir a Java Script. RichFaces incluye ciclo de vida, validaciones, conversores y la gestión de recursos estáticos y dinámicos. Los componentes de RichFaces están construidos con soporte Ajax y un alto grado de personalización del "look-and-feel", que puede ser fácilmente incorporado dentro de las aplicaciones JSF". (10) Es un proyecto open source activo y con una comunidad también activa.

RichFaces permite:

- Crear rápidamente vistas complejas, basándose en la caja de componentes, posee una librería Interfaz de usuario (UI, por sus siglas en inglés), que contiene componentes para agregar características de interfaz de usuario a aplicaciones JSF.
- Escribir componentes propios con función soportada por Ajax.
- Proporcionar un avanzado soporte a la gestión de diferentes recursos: imágenes, código JavaScript y hojas de estilo en cascada.
- Generar casos de prueba para los componentes que se están creando (actions, listeners, etc.).

Los componentes de la interfaz de usuario de RichFaces, vienen preparados para su uso fuera del paquete, así, los desarrolladores ahorrarán tiempo y podrán disponer de las ventajas mencionadas para la creación de aplicaciones Web. Como resultado, la experiencia puede ser más rápida y fácil de obtener. Permite definir (por medio de etiquetas de JSF), diferentes partes de una página JSF que se desee actualizar con una solicitud Ajax, proporcionando de esta forma, varias opciones para enviar peticiones Ajax al servidor.

Ajax4JSF, es un framework de código abierto que integra funcionalidades AJAX con el ciclo de vida JSF. Presenta mejoras sobre los propios beneficios del framework JSF incluyendo el ciclo de vida, validaciones, facilidades de conversión y el manejo de recursos estáticos y dinámicos. Permite definir un evento en una página que invoca una petición AJAX y luego, las áreas de la página deberían sincronizarse con el Árbol de Componentes JSF, después de que la petición Ajax cambie los datos en el servidor. Permite la realización de peticiones automáticas al servidor y el control de cualquier evento que realice el usuario. Brinda a la aplicación JSF, un contenido mucho más profesional con muy poco esfuerzo y aporta al programador una gran ventaja: la utilización de la funcionalidad AJAX dentro de páginas JSF, sin la necesidad de crear nuevo código JavaScript.

Java Server Facelets, es un framework para plantillas (templates), centrado en la tecnología JSF (Java Server Faces), por lo cual se integran de manera muy fácil.

Este framework incluye muchas características siendo las más importantes:

- Tiempo de desarrollo cero de los tags para UIComponents.
- Facilidad en la creación del templating para los componentes y páginas.
- Habilidad de separar los UI Components en diferentes archivos.
- Un buen sistema de reporte de errores.
- Soporte completo a Expression Language (EL, por sus siglas en inglés).
- Validación de EL en tiempo de construcción.
- No es necesaria la configuración del lenguaje de marcado extensible (XML, por sus siglas en inglés).
- Trabaja con cualquier Render Kit.

Facelets es una tecnología centrada en la creación de árboles de componentes y se relaciona estrechamente con el complejo ciclo de vida JSF.

Las principales ventajas de Facelets son:

- No depende de un contenedor Web.
- Integrar JSP con JSF trae problemas, además, no se puede usar Java Server Pages Standard Tag Library (JSTL, por sus siglas en inglés), con JSF.

- Provee un proceso de compilación más rápido que JSP.
- Provee templating, lo cual implica reutilización de código, simplificación de desarrollo y facilidad en el mantenimiento de grandes aplicaciones.
- Permite crear componentes ligeros sin necesidad de crear los tags de los UI Components.
- Soporta Unified Expression Language, incluyendo soporte para funciones EL y validación de EL en tiempo de compilación.
- Fácil creación de funciones y librerías de componentes.

El Lenguaje de Marcado de Hipertexto Extensible(XHTML, por sus siglas en inglés), es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a este último lenguaje ante su limitación de uso, con las cada vez más abundantes herramientas basadas en el Lenguaje de Marcado Extensible (XML, por sus siglas en inglés).

XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar y describir datos, con la de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium, para 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 desea escribir el autor.
- Como es XML se pueden utilizar fácilmente con herramientas creadas para procesamiento de documentos XML genéricos (editores, XSLT, etc.).

JBoss Seam, es un framework Open source, que integra y unifica los distintos estándares de la plataforma Java EE 5.0, permitiendo el trabajo con todos ellos, siguiendo el mismo modelo de programación. Ha sido diseñado intentando simplificar al máximo el desarrollo de aplicaciones, basando el diseño en Plain Old Java Objects (POJOs, por sus siglas en inglés). Estos componentes se usan desde la capa de persistencia hasta la de presentación, poniendo todas las capas en comunicación directa.

El núcleo principal de Seam, está formado por las especificaciones Enterprise JavaBeans 3 (EJB3) y Java Server Faces.

Seam elimina la barrera existente entre estas tecnologías, permitiendo usar EJBs directamente como "backingbeans" de JSF y lanzar, o escuchar eventos web, a través de en un solo framework bien acoplado y basado en estándares, ampliamente utilizados y probados (escalables, portables y reusables).

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), que permiten establecer estas relaciones. Hibernate es software libre.

“Las características de esta tecnología son:

- Soluciona 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).
- Es flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente.
- Tiene la funcionalidad de crear la base de datos a partir de la información disponible.
- Ofrece un lenguaje de consulta de datos llamado Hibernate Query Language (HQL, por sus siglas en inglés), al mismo tiempo que una API, para construir las consultas (conocida como "criteria").
- Puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations, que implementa el estándar Java Persistence Aplicattion, que es parte de esta plataforma”. (11)

Ofrece un lenguaje para realizar consultas a la base de datos. Este lenguaje es similar a SQL, y se denomina Hibernate Query Lenguaje (HQL, por sus siglas en inglés). El HQL permite utilizar un lenguaje intermedio dependiendo de la base de datos que se utilice, y el dialecto será traducido al SQL dependiente de la base de datos, de manera transparente y automática y simplificando el código.

Hibernate proporciona grandes beneficios como es la independencia de la base de datos, bajo acoplamiento entre negocio y persistencia, y un desarrollo rápido. Se puede cubrir de manera sencilla y rápida el 80 - 90% de la persistencia de la aplicación. Esto permite centrar los esfuerzos en optimizar las consultas.

Enterprise JavaBeans (EJB, por sus siglas en inglés), arquitectura para un sistema transaccional de objetos distribuidos, basado en componentes que permite construir aplicaciones portables, reusables y escalables. “Una de las metas de la arquitectura EJB, es poder escribir de manera fácil aplicaciones de negocio orientadas a objetos y distribuidas, basadas en el lenguaje de programación JAVA. Las API que lo conforman, son parte del estándar de construcción de aplicaciones empresariales J2EE de Sun Microsystems (ahora JEE 5.0). Su especificación, detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor que son, precisamente, los EJB”. (12)

Los EJB proporcionan un modelo distribuido y estándar de componentes que se ejecutan en el servidor. El objetivo de los EJB es dotar al programador de un modelo, que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad), para centrarse en el desarrollo de la lógica de negocio en sí. En cuanto a la persistencia de entidades, define su propio manejo de persistencia, y la posibilidad de emplear anotaciones en ORM, EJB QL y sentencias SQL nativas, además de integración con Hibernate.

El **Java Persistence API** (JPA, por sus siglas en inglés), “es una especificación de Sun Microsystems para la persistencia de objetos Java a cualquier base de datos relacional. Esta API fue desarrollada para la plataforma JEE e incluida en el estándar de EJB 3.0”. (13)

Este framework permite a los desarrolladores, organizar datos relacionales en aplicaciones Java. Establece una conexión a la base de datos configurable haciendo uso la implementación del Entity Manager a través de un container provider. Transforma los objetos de nivel medio, usando anotaciones y administra la sesión con la base de datos usando el soporte del servidor de aplicaciones o del container provider para la administración de transacciones.

“La persistencia consiste de cuatro áreas:

- JPA (javax.persistence).
- El lenguaje de consultas.
- La Java Persistence Criteria API.

- El metadata de mapeo objeto/relacional". (14)

Java, es un lenguaje de programación simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. Trabaja con sus datos como objetos y con interfaces a esos objetos. "Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Realiza verificaciones en busca de problemas, tanto en tiempo de compilación, como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo". (15)

"Una característica distintiva de Java, es su capacidad multiplataforma. Esto lo consigue, no solo a nivel de código fuente, sino también a nivel de código compilado. Java podrá ejecutarse en cualquier sistema operativo que tenga una máquina virtual Java compatible". (16)

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), "es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software". (17)Mediante UML 2.0 es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software, previo al proceso intensivo de escribir código. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que es un lenguaje, cuenta con reglas para combinar tales elementos. UML no es un proceso, es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Las ventajas que proporciona este lenguaje de modelado son las siguientes:

- Permite modelar sistemas, utilizando técnicas orientadas a objetos (OO).
- 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).
- Permite documentar todos los artefactos de un proceso de desarrollo.
- 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.
- El tiempo invertido en el desarrollo de la arquitectura se minimiza.

- La detección y resolución de errores se agiliza siempre y cuando, se haga uso de herramientas adecuadas de diagnóstico y depuración.
- La trazabilidad y documentación del proyecto se realiza de una forma ordenada y guiada por los casos de uso.

Un **servidor de aplicaciones**, es un software que permite el procesamiento de datos de una aplicación 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). Las principales ventajas de la tecnología de los servidores de aplicación son: la centralización y la disminución de la complejidad del desarrollo de aplicaciones. Es un producto basado en un componente que se encuentra en el plano medio de la arquitectura central de un servidor. Proporciona servicios de 'middleware', es decir, trabaja como un intermediario para la seguridad y el mantenimiento, además de proveer acceso a los datos.

Jboss, es el servidor de aplicaciones de uso libre y código abierto más desarrollado del mercado. Por ser una plataforma certificada Java EE, soporta todas las funcionalidades de Java EE 1.4, incluyendo servicios adicionales como clustering, caching y persistencia. Jboss es ideal para aplicaciones Java y aplicaciones basadas en la web. También soporta Enterprise Java Beans (EJB) 3.0, y esto hace que el desarrollo de las aplicaciones sean mucho más simples. Dentro de sus características más destacadas se encuentran los servicios de Middleware para Java, cumple los estándares, es confiable a nivel de empresa, integrable y orientado a arquitectura de servicios. Posee una flexibilidad consistente. Se ha convertido en uno de los servidores de aplicaciones más usados y con la comunidad más amplia en el mercado.

Un **Sistema Gestor de Bases de Datos** (SGBD), es una herramienta que permite a los usuarios crear y mantener una base de datos, por lo que es un software de propósito general, que facilita el proceso de definir, construir y manipular, los datos almacenados. Estos sistemas se especializan en la validación de datos para evitar redundancias y errores que afecten la integridad y seguridad de la información registrada. Además, garantizan el control centralizado de la información de manera sistemática y única, así como el acceso a la base de datos, y la integración y sincronización de ella, para el uso de múltiples usuarios.

PostgreSQL, es el sistema de gestión de base de datos relacional, orientada a objetos de código abierto más avanzado del mundo. PostgreSQL, es un potente sistema de base de datos objeto/relacional. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de los datos, y corrección. Funciona en todos los principales sistemas

operativos, incluyendo Linux, UNIX y Windows. Es altamente escalable, tanto en la enorme cantidad de datos que puede administrar, como en el número de usuarios concurrentes que puede soportar.

“Posee numerosas ventajas entre ellas se pueden mencionar:

- Instalación ilimitada.
- Mejor soporte que los proveedores comerciales.
- Estabilidad y confiabilidad legendarias.
- Extensible pues el código fuente está disponible para todos sin costo.
- Multiplataforma.
- Diseñado para ambientes de alto volumen.
- Herramientas gráficas de diseño y administración de bases de datos. Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora , Data Architect).
- Altamente extensible, pues soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Soporte SQL Comprensivo, soporta la especificación SQL99, e incluye características avanzadas tales como las uniones (joins) SQL92.
- Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- Usa la tecnología PostgreSQL MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), para evitar bloqueos innecesarios.
- Usa una arquitectura proceso-por-usuario cliente/servidor (Cliente/Servidor).
- Usa Write Ahead Logging (WAL) para incrementar la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos, garantizando que existirá un registro de las transacciones a partir del cual se puede restaurar la base de datos”. (18)

"Las **herramientas Computer Aided Software Engineering** (CASE, por sus siglas en inglés), Ingeniería de Software Asistida por Ordenador, son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas, en términos de tiempo y de dinero". (19) Estas herramientas, permitirán organizar y manejar la información de un proyecto informático, permitiéndoles a los participantes del mismo, que los sistemas se tornen más flexibles y comprensibles. Presentan diversas ventajas como la facilidad para la revisión de aplicaciones, el soporte para el desarrollo de prototipos de sistemas, la generación de código, la mejora en la habilidad para satisfacer los requerimientos del usuario.

Visual Paradigm, es una herramienta multiplataforma de modelado visual UML y una herramienta CASE muy potente y fácil de utilizar. "Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML, ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación". (20)

La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas, la generación automática de informes en formato PDF, Word o HTML; generación de máquinas de estados, y la integración con Visio y Rational Rose.

Un **entorno de desarrollo integrado** o Integrated Development Environment (IDE, por sus siglas en inglés), es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien, puede dar cabida a varios de estos. Como elementos básicos, un IDE cuenta con en un editor de código, un compilador/intérprete y un depurador.

"Eclipse, es principalmente una plataforma de programación, usada para crear entornos integrados de desarrollo. Eclipse, fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para Visual Age. Es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro, que fomenta una comunidad de código abierto". (21) "Los usuarios pueden ampliar sus capacidades instalando los plugins escritos para el marco del software del eclipse, tal como cajas de herramientas del desarrollo para otros lenguajes de programación". (22) Es multiplataforma y sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También brinda soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, JavaScript, HTML, PHP o Python, etc. Tiene

características potentes como el completamiento de código, permite mostrar desde varias perspectivas el entorno de cada framework, permitiendo la integración de sus funcionalidades y el trabajo en equipo. Eclipse necesita tener instalado en el sistema una máquina virtual Java, preferiblemente JRE (Java Runtime Environment).

La elaboración y análisis del presente capítulo ha permitido concluir que la aplicación del Proceso Unificado de Desarrollo de Software al ciclo de vida del sistema, constituyó una práctica eficiente que permite la obtención exitosa de los artefactos generados durante el desarrollo del software. El análisis de los flujos de trabajo desarrollados por el analista posibilitó una mejor comprensión de los procesos que se desarrollan en el Hospital Pediátrico William Soler, así como, las funcionalidades y requisitos funcionales con los que debe cumplir el sistema, por lo que se verificó que se encontraban definidos de la forma más óptima. El estudio de las ventajas y desventajas de las herramientas a utilizar, confirmó reiteradamente que eran las más indicadas para el desarrollo del trabajo, pues son tecnologías libres que ofrecen los beneficios necesarios para la implementación de las soluciones de informática médica. La descripción de las actividades, artefactos y trabajadores involucrados en el flujo de Implementación, evidenció la importancia de cada uno de ellos en la construcción del sistema.

CAPÍTULO 2. Descripción de la arquitectura de software del Sistema de Evaluación del Neurodesarrollo en Niños.

Este capítulo tiene como objetivos, definir y describir los requerimientos no funcionales del sistema; analizar y fundamentar la arquitectura de software del Módulo de Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños; detallar las estrategias de integración con el Sistema de Gestión Hospitalaria; describir la seguridad propuesta para el sistema y especificar los estándares de diseño y codificación a utilizar en la realización de la implementación.

2.1. Requisitos No Funcionales.

"Las necesidades actuales de toda organización, demandan la construcción de grandes y complejos sistemas de software que requieren de la combinación de diferentes tecnologías y plataformas de hardware y software, para alcanzar un funcionamiento acorde con dichas necesidades. Lo anterior, exige a los profesionales dedicados al desarrollo de sistemas informáticos, poner especial atención y cuidado al diseño de la arquitectura, bajo la cual estará soportado el funcionamiento de sus sistemas". (23)

Si una arquitectura de software se encuentra deficiente en su concepción o diseño, existirán grandes posibilidades de construir un sistema que no alcanzará el total de los requerimientos establecidos. El desarrollo de la arquitectura de software es una de las etapas fundamentales, pues es aquí, donde los profesionales aportan todos sus conocimientos, creatividad y experiencia, para crear la mejor propuesta de solución que se dará al cliente que cumpla con los requerimientos funcionales y no funcionales establecidos para el sistema en desarrollo; de ahí la importancia de una buena definición de la misma.

Los Requerimientos No Funcionales (RNF), son cualidades o propiedades que el sistema debe cumplir para facilitar su uso y darle valor agregado a las funcionalidades que le brinda al usuario. Estos requisitos son de gran significación en la aceptación del software, debido a que representan las ventajas más visibles al usuario y repercuten en el óptimo funcionamiento y mantenimiento del sistema.

Los Requisitos No Funcionales definidos para el sistema en cuestión se encontraban predefinidos por el ALAS HIS. Durante el desarrollo de la investigación se realizará un mayor énfasis en los de eficiencia, usabilidad, restricciones de diseño y los requerimientos de rendimiento, a los que se les proporcionará una solución efectiva en el transcurso del flujo de trabajo Implementación.

RNF Usabilidad.

“La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y atractivo para el usuario, en condiciones específicas de uso”. (24)

Estos requerimientos describen los niveles apropiados de usabilidad, dados los usuarios finales del producto. Los requerimientos de usabilidad se derivan de la combinación de lo que el cliente está tratando de lograr con el producto y lo que los usuarios finales esperan del mismo.

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

- Para alcanzar un nivel elemental asociado al dominio del sistema y el uso eficiente del mismo, serán necesarios 20 días de preparación, obteniendo la categoría de usuarios normales.
- Para alcanzar un nivel avanzado asociado al dominio del sistema y el uso eficiente del mismo, serán necesarios 30 días de preparación, obteniendo la categoría de usuarios avanzados.
- Para brindar comodidad a la hora de acceder a las diferentes funcionalidades que proporciona la aplicación mediante teclas de acceso rápido.

RNF Fiabilidad.

“La fiabilidad es la característica de los sistemas informáticos por la que se mide el tiempo de funcionamiento sin fallos”. (25)

En la investigación se propone garantizar la misma de diversas formas:

- En los servidores de los hospitales se deberá proporcionar una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos. Se plantea el empleo de políticas de respaldo a toda la información, para evitar pérdidas en caso de desastres ajenos al sistema.
- El sistema soportará el uso de firmas digitales para la transferencia de información cuya certificación sea imprescindible, para validar el uso de la misma.

- Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos, independientemente de su inexistencia para el sistema. Permitirá la recuperación de la información de la base de datos, a partir de los respaldos, o salvas realizadas.

RNF Eficiencia.

Imponen condiciones a los Requerimientos Funcionales. Por ejemplo, para una acción específica pueden definirse parámetros tales como: velocidad de procesamiento o cálculo, rendimiento, disponibilidad, precisión, tiempo de respuesta, tiempo de recuperación y aprovechamiento de los recursos.

- El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos.
- El sistema minimizará el volumen de datos en las peticiones y además, optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla, guardar en la memoria caché datos y recursos de alta demanda.
- El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos. Se deberá usar siempre que sea posible, el patrón Singleton (instancia única), destruir referencias que ya no estén siendo usadas, y optimizar el trabajo con cadenas, entre otras buenas prácticas, que ayudan a mejorar el rendimiento.

RNF Soporte.

Abarcan todas las acciones que se llevarán a cabo una vez que se ha terminado el desarrollo del software, con motivos de asistir a los clientes de este, logrando así su mejora progresiva y evolución en el tiempo. Pueden incluir: pruebas, extensibilidad, adaptabilidad, mantenimiento, configuración, compatibilidad, servicios, instalación.

- Las notificaciones de las deficiencias detectadas en la aplicación desplegada deberán realizarse por escrito.
- Una vez notificada por la entidad médica, la deficiencia detectada en la aplicación desplegada, el equipo de desarrollo deberá solucionarla en un período de 7 días.

- La capacitación y entrenamiento del profesional de salud para el uso del sistema, se realizará en un período de 6 meses.

RFN Réplica.

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

RNF Restricciones de diseño.

Este tipo de requerimiento, especifica o restringe la codificación o construcción de un sistema; son restricciones que han sido ordenadas y deben ser cumplidas estrictamente.

Ejemplos de ellas son:

- Estándares requeridos.
- Lenguajes de programación a ser usados para la implementación.
- Uso obligatorio de ciertas herramientas de desarrollo.
- Restricciones en la arquitectura o el diseño.
- Bibliotecas de clases.

El sistema estará dividido en las siguientes capas:

Capa física:

Cliente: computadora con cualquier tecnología o sistema operativo, que cuente con un navegador actualizado y que siga los estándares web (se recomienda IE 6 o superior o Firefox 2.x).

Servidor de Aplicaciones: servidor con cualquier tecnología o sistema operativo que soporte el Java Runtime Environment (JRE) 1.5 o superior y al JBoss AS 4.2 o superior. 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 o superior en los servidores de base de datos de cada hospital, y Oracle 11g o superior, para los servidores de base de datos del Centro de Datos.

Capa lógica:

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 de 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 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.

Interfaz.

Interfaces de usuario:

- Las ventanas del sistema contendrán bien estructurados los datos, además de permitir 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, que lo guíen paso a paso 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 Rendimiento.

La eficiencia del sistema estará dada por el aprovechamiento de los recursos que se disponen en el modelo usuario. El sistema debe ser rápido y el tiempo de respuesta debe ser el mínimo posible.

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

RNF Seguridad.

Este es quizás el tipo de requerimiento más difícil, que provocará los mayores riesgos si no se maneja correctamente, debido a la cantidad de personas dedicadas a encontrar fallas de seguridad en las aplicaciones y a explotarlas con fines malintencionados. Por ello, se hace necesario establecer una serie de políticas de Seguridad tendentes a garantizar la continuidad del sistema en el caso que se produzcan incidencias, fallos, actuaciones malignas por parte de terceros, pérdidas accidentales o desastres que afecten los datos almacenados.

- La información estará protegida contra accesos no autorizados utilizando mecanismos de validación que puedan garantizar el cumplimiento de: usuario, contraseña y nivel de acceso, de manera que, cada uno pueda tener disponible solamente las opciones relacionadas con su actividad y tenga datos de acceso propios, garantizando así la confidencialidad.
- Se usarán mecanismos de encriptación para el envío de las contraseñas al servidor de base de datos.
- Las contraseñas podrán cambiarse únicamente por el propio usuario, o por el administrador del sistema.
- Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando solo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión.
- El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (log) de cada usuario en el sistema.

RNF Hardware.

Los requerimientos de hardware estarán en correspondencia con la plataforma específica que se utilice para la instalación del sistema, en cuanto a sistema operativo, servidor de aplicaciones y gestor de bases de datos.

Estaciones de trabajo:

En la solución se incluyen estaciones de trabajo para las consultas del Alas HIS, las que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, se recomienda Internet Explorer 7, Firefox 2, o versiones superiores. Por tal motivo 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 y residencia de la información y las aplicaciones bajo esquemas seguros y confiables. Servidores de Base de datos: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual-Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux. Servidores de Aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux. Servidores de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual-Core 2 GB de memoria y 2x72GB de disco o superior y sistema operativo Linux.

RNF Software.

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

2.2. Fundamentación de la descripción de la arquitectura.

“La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos, el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. (26) “Es una vista estructural de alto nivel, define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales. Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario, para guiar la construcción del software para un sistema de información”. (27)

Aporta además, una visión abstracta de alto nivel, postergando el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. Establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común, que permita alcanzar los objetivos y necesidades del sistema.

La importancia de representar de forma explícita la arquitectura de los sistemas de software es evidente. En primer lugar, estas representaciones elevan el nivel de abstracción, facilitando la comprensión de los sistemas de software complejos. En segundo lugar, hacen que aumenten las posibilidades de reutilizar tanto la arquitectura, como los componentes que aparecen en ella. 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.

2.2.1 Patrones de arquitectura y diseño.

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Los patrones pueden ser:

Patrones arquitecturales: aquellos que expresan un esquema organizativo estructural fundamental para sistemas software.

Patrones de diseño: son un conjunto de estrategias, o buenas prácticas, que facilitan el trabajo en muchas situaciones, a la hora de realizar una aplicación orientada a objetos.

Idiomas: patrones de bajo nivel específicos para un lenguaje de programación, o entorno concreto.

Para el desarrollo de las funcionalidades, se propone la utilización del patrón de arquitectura Modelo-Vista-Controlador.

2.2.1.1 Modelo-Vista-Controlador.

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. “El patrón Modelo-Vista-Controlador (MVC) se utiliza frecuentemente en aplicaciones web. La vista, es la página HTML; el código que provee de datos dinámicos a la página, el controlador; el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos”. (28) El controlador representa la Lógica de negocio y gestiona todos los eventos de entrada del usuario constituyendo el intermediario entre el Modelo y la Vista.

Para el desarrollo de la aplicación, se utiliza del patrón modelo-vista-controlador, pues se encontraba definido previamente por el Departamento de Gestión Hospitalaria y además su uso brinda muchas ventajas:

La aplicación está diseñada modularmente y sus vistas muestran información actualizada siempre.

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. La separación de capas como presentación, lógica de negocio y acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y fácilmente mantenibles, lo que al final, se deriva en un ahorro de tiempo en desarrollo. Se pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc. sin alterar la lógica de negocio.

Son más sencillas las labores de mejora como: agregar nuevas vistas, modificar los objetos de negocio, bien sea para mejorar el performance, o para migrar a otra tecnología. Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones se deben hacer en un solo lugar y no en varios, como sucedería, si se tuviese una mezcla de presentación e implementación de la lógica del negocio. Las vistas también son susceptibles de modificación, sin necesidad de provocar que todo el sistema se paralice.

Las tecnologías utilizadas se encuentran distribuidas por capas, las mismas fueron anteriormente presentadas en el capítulo 1.

2.3. Estrategias de integración.

El módulo de logopedia perteneciente al Sistema de Evaluación del Neurodesarrollo en Niños, se integrará al Sistema de Gestión Hospitalaria Alas HIS conformando un módulo del mismo.

Para la aplicación de los exámenes, se requiere de la integración para la adquisición de componentes del Módulo de Expediente y Turnera; la misma se concreta a través de los listados de los pacientes a atender en el día, según la programación de las consultas.

Para obtener los datos personales del paciente y la lista de pacientes a atender, se hace necesaria la relación del Módulo de Expediente y Turnera con el Alas HIS, a través de la tabla ModeExpediente. Dicha dependencia viene dada con el esquema HC_Local del Alas HIS, que gestiona entre otros aspectos, los datos personales del paciente en la tabla Hoja_Frontal, el identificador de esta tabla es el responsable de efectuar la relación.

2.4. Especificación de los términos de seguridad.

"La seguridad puede entenderse como aquellas reglas técnicas o actividades destinadas a prevenir, proteger y resguardar lo que es considerado como susceptible de robo, pérdida, o daño, ya sea de manera personal, grupal o empresarial". (29)

Los problemas de seguridad informática no pueden ser tratados aisladamente, ya que la seguridad de todo el sistema, es igual a la de su punto más débil. Por lo tanto, se apoyarán en técnicas desarrolladas para proteger los equipos informáticos individuales y conectados en una red frente a daños accidentales o intencionados. Estos daños incluyen el mal funcionamiento del hardware, la pérdida física de datos y el acceso a bases de datos, por personas no autorizadas.

La seguridad en el sistema, es de gran importancia, de ella depende la integridad, autenticidad y confiabilidad de su información. En el caso del sistema en cuestión, la seguridad es llevada a cabo por el proyecto Alas HIS. Se definieron diferentes tipos de seguridad: acceso al sistema, registro de trazas, administración de seguridad (vista lógica y vista física) y configuración de funcionalidades.

Iniciar/Cerrar sesión de trabajo.

Cuando el usuario necesita acceder al sistema, este solicita: nombre de usuario y contraseña. El usuario introduce los datos solicitados, el sistema verifica que los datos introducidos sean válidos, si es así, el usuario accede al módulo de Logopedia. El sistema muestra como opciones del menú, las funcionalidades a las que tiene permiso de 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. Una vez que el usuario introduce su contraseña es encriptada a través del algoritmo MD5. Este algoritmo calcula el hash de las claves de los usuarios; en la base de datos se almacena el resultado del MD5, y cuando el usuario desea entrar en el sistema se compara el hash MD5 de la clave introducida con el hash que hay guardado la base de datos. Si coinciden, es la misma clave y el usuario será autenticado.

Registrar trazas.

El usuario realiza una acción sobre el sistema, que 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 en las funcionalidades del módulo.

Configurar funcionalidades.

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

2.5. Vista de Despliegue.

"Un diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes de software que representan manifestaciones del código en tiempo de ejecución" **(30)**.

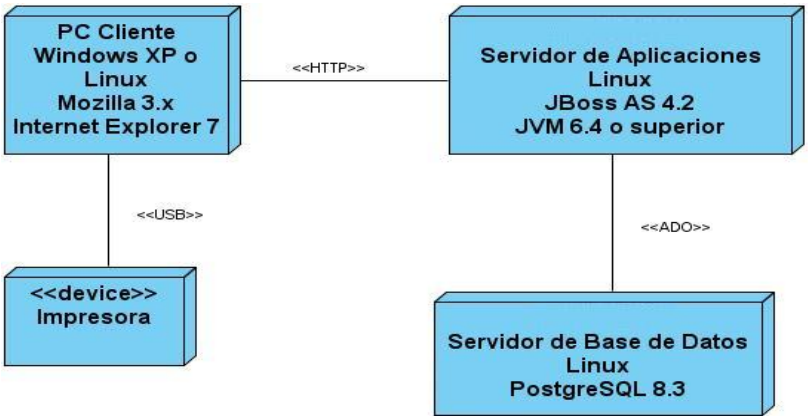


Figura 1: Diagrama de despliegue.

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

El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación; estos son reglas específicas de cada lenguaje de programación cuyo cumplimiento reduce de forma significativa el riesgo de que los desarrolladores introduzcan errores. La legibilidad del código fuente, repercute directamente en la comprensión del programador de un sistema de software.

La facilidad de cambio del código, es la facilidad con que la aplicación pueda modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Para la implementación del sistema propuesto se utilizaron varios estándares de codificación como son:

Idioma: se debe utilizar como idioma el español, las palabras no se acentuarán.

Indentación: “es usada para tener una mejor visibilidad en el diseño de un programa. Muestra las líneas que están subordinadas a otras líneas. Por ejemplo, todas las líneas que forman el cuerpo de un ciclo deberán estar indentadas con la instrucción principal del ciclo”. (31)

Objetivo: lograr una estructura uniforme para los bloques de código, así como para los diferentes niveles de anidamiento.

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

Aspectos generales:

El indentado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC y 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. Nunca colocar ({) en la línea de un código cualquiera, esto requiere una línea propia.

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

Objetivo**:** establecer un modo común para comentar el código de forma tal, que sea comprensible con solo leerlo una vez.

Ubicación de comentarios:

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

Líneas en blanco:

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

Espacios en blanco:

Entre operadores lógicos y aritméticos. Se recomienda usar espacios en blanco entre estos operadores, para lograr una mayor legibilidad en el código.

Ejemplo de código: paciente = nompaciente.

Aspectos generales:

Sobre el comentario:

- Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones, debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco, o se escribe a continuación de la instrucción.

Sobre los espacios en blanco:

No se debe usar espacio en blanco:

- Después del corchete abierto y antes del cerrado de un arreglo.
- Después del paréntesis abierto y antes del cerrado.
- Antes de un punto y coma.

Variables y constantes:

Apariencia de variables:

Las variables tendrán un prefijo para el tipo de datos en minúscula. El nombre de las variables debe comenzar con la primera letra en minúscula, la cual identificará el tipo de datos al que se refiere, en caso de que sea un nombre compuesto, se empleará notación CamellCasing**.

Ejemplo: sNombrePaciente.

Apariencia de constantes:

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

Aspectos generales:

Nombres de las variables y constantes. El nombre empleado, debe permitir que con solo leer se conozca el propósito de las mismas.

Clases y objetos:

Objetivo**:** Nombrar las clases e instancias de forma estándar para todas las aplicaciones.

Apariencia de clases y objetos:

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

Apariencia de atributos:

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

Apariencia de las funciones:

Primera letra en mayúscula. Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación PascalCasing*. Ejemplo: FunctionBuscarUnidad (). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set.

Declaración de parámetro en funciones:

Agrupados por tipos. Poner los string 1 numéricos 2, además, agrupar según valores por defecto. Los parámetros que se le pasan a las funciones, se recomienda sean declarados de forma tal, que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos.

Aspectos generales:

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

Bases de Datos, Tablas, Esquemas y Campos.

Apariencia de la Base de Datos:

Los nombres de las Bases de Datos deben comenzar con mayúscula y con underscoard entre palabras.

Ejemplo: Nueva_Linea_Base.

Apariencia de las vistas:

Las 2 primeras letras representan el módulo. Todas las letras en minúscula. El nombre a emplear para las vistas debe comenzar con el prefijo del módulo seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor.

Ejemplo: modl_buscar_paciente.

Apariencia de las tablas:

La primera palabra especifica el módulo. Todas las letras en minúscula. El nombre a emplear para las tablas debe comenzar con el prefijo del módulo seguido de underscoard y luego debe escribirse el nombre de la tabla, en caso de que sea un nombre compuesto se utilizará underscoard para separarlo.

Ejemplo: “modl_hoja_frontal”.

Tablas que representen Relaciones:

Todas las letras en minúscula. El nombre a emplear para estas tablas de relación debe comenzar con el nombre de la primera tabla seguido de underscoard, luego la palabra “in” y el nombre de la segunda tabla. Ejemplo: “paciente_in_consultas”.

Tablas que representen nomencladores:

Todas las letras en minúscula. El nombre a emplear para estas tablas debe comenzar con el nombre del módulo seguido de underscoard, luego la palabra “n” y el nombre del nomenclador. Ejemplo: “mod_n_sexo”.

Apariencia de los campos:

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

Nombre de los campos:

En caso de identificadores. Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo. Ejemplo: id_paciente.

Sentencias SQL:

Todas las letras en mayúscula. Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.

Aspectos generales:

Sobre las BD, vistas, tablas atributos y procedimientos. El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con solo leerlos se conozca el propósito de los mismos.

Controles.

Apariencia de los controles:

Los controles tendrán un prefijo para el tipo de datos en minúscula. El nombre que se le da a los controles debe comenzar con las primeras letras en minúscula, las cuales identificarán el tipo de datos al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Ejemplo: btnAceptar

Estándares y estilos a utilizar:

Notación PascalCasing: los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula.

Ejemplo: NotacionPascalCasing.

Notación CamellCasing: los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamelCasing.

La confección de este capítulo, proporcionó la adquisición de una base para verificar la eficacia del uso del patrón MVC, al diseñar la aplicación de forma modular. Los requerimientos no funcionales, son de vital importancia para que clientes y usuarios, puedan valorar las características no funcionales del producto. La

seguridad del sistema, constituye un aspecto crítico a la hora de almacenar la información confidencial de los pacientes, pues en la actualidad, existe un gran número de personas dedicadas a detectar y explotar posibles fallas en los sistemas. La vista de despliegue es fundamental para modelar el hardware utilizado en la implementación del sistema y las relaciones entre sus componentes. El estudio de los estándares y estilos de programación a utilizar es necesario para un mejor entendimiento y organización del código, estos se encontraban definidos previamente por el Departamento de Gestión Hospitalaria.

CAPÍTULO 3. Descripción y análisis de la solución propuesta.

Con la elaboración del capítulo actual, se persiguen los siguientes objetivos: describir las nuevas clases u operaciones necesarias para implementar los casos de usos, refinar el modelo de datos, conformar la vista de implementación con la intención de mostrar las organizaciones y dependencias lógicas entre componentes software y elaborar el diagrama de componentes que será utilizado para modelar la vista estática del sistema.

3.1. Valoración crítica del diseño propuesto por el analista.

El diseño, es el núcleo técnico de la ingeniería del software durante el cual, se desarrollan, revisan y documentan, los refinamientos progresivos de la estructura de datos, arquitectura, interfaces y datos procedimentales de los componentes del software. En la definición del mismo se generan artefactos como el modelo de clases del diseño, diagramas de interacción, modelo de despliegue y el prototipo no funcional del sistema; por lo que se define como la entrada principal al flujo de trabajo de implementación.

Al darle continuidad al ciclo de vida del software durante el flujo de trabajo de implementación, fue necesario realizar un análisis valorativo del diseño propuesto por el analista del módulo de logopedia del SENDN. Luego de este análisis se concluyó, que algunos artefactos no cumplían con los requerimientos para la realización exitosa de la implementación del módulo, como el diagrama de clases del diseño. A dicho diagrama hubo que incorporarle nuevos métodos, atributos y relaciones, pues los existentes no cumplían con los requerimientos necesarios para la solución del problema; teniendo que transformar así, los diagramas de secuencia. Además, se hizo necesaria la modificación del diagrama de clases persistentes, añadiendo atributos, clases y relaciones, generando nuevamente el modelo de datos y el script para la base de datos.

Los prototipos nos funcionales elaborados no poseían todos los componentes obligatorios para la realización de los casos de usos, además presentaban errores en el código y la ausencia de las pautas de diseño definidas por el Alas HIS, por lo que se procedió a su restructuración y perfeccionamiento.

3.2. Descripción de las nuevas clases u operaciones necesarias.

En el refinamiento realizado al diseño propuesto por el analista, se definió la incorporación o modificación de atributos y métodos, como se muestra en la Tabla 1:

Tabla 1: Descripción de la clase Modl_HistoriaClinica.

Nombre: Modl_HistoriaClinica.	
Tipo de clase: controladora.	
Atributos	**Tipos**
scraneo	String
lListar_craneo	List<String>
sDesc_craneo	String
sFacie	String

Nombre: ModlModificarHistoriaClinica.	
lListar_facie	List<String>
sDesc_facie	String
sFaringe	String
lListar_faringe	List<String>
sDesc_faringe	String
sTorax	String
lListar_torax	List<String>
sDesc_torax	String
sAbdomen	String
Columna	String
Desc_columna	String
Desc_otros	String

Nombre: Modl_HistoriaClinica.	
Tipo de clase: controladora.	
Atributos	**Tipos**
sDesc_otros	String
sLabios	String
lListar_labios	List<String>
sDesc_labios	String
lListar_abdomen	List<String>
sDesc_abdomen	String
sColumna	String
lListar_columna	List<String>
sPaladar_oseo	String
lListar_paladar_oseo	List<String>
sDesc_paladar_oseo	String
sPaladar_blando	String
lListar_paladar_blando	List<String>
sDesc_paladar_blando	String
sOclusion_dentaria	String
lListar_oclusion_dentaria	List<String>
sDesc_oclusion_dentaria	String
sRelacion_maxilar	String
lListar_relacion_maxilar	List<String>

Nombre: Modl_HistoriaClinica.	
Tipo de clase: controladora.	
Atributos	**Tipos**
sDesc_relacion_maxilar	String
sMandibula	String
lListar_relacion_mandibula	List<String>
sDesc_mandibula	String
sMaxilar	String
lListar_maxilar	List<String>
sDesc_maxilar	String
sLaringe_sigue	String
lListar_laringe_sigue	List<String>
sLaringe_desplaza	String
lListar_laringe_desplaza	List<String>
sDesc_indirecta	String
sCuello_inspeccion	String
sExpontaneo	String
lListar_expontaneo	List<String>
sExpcuantita_impresiona	String
lLstar_expcuantita_impresiona	List<String>
sExpcuantita_referido	String
lListar_expcuantita_referido	List<String>

Nombre: Modl_HistoriaClinica.	
Tipo de clase: controladora.	
Atributos	**Tipos**
lListar_expcuantita_referido	List<String>
sExpresivo_cualit_numero_palabras	String
sH_vocales_a	String
bCheck_h_vocales_a	boolean
sH_vocales_e	String
bCheck_h_vocales_e	boolean
sH_vocales_i	String
bCheck_h_vocales_i	boolean
sH_vocales_o	String
bCheck_h_vocales_o	boolean
sH_vocales_u	String
bCheck_h_vocales_u	boolean
Para cada responsabilidad:	
Nombre:	CrearHistoriaClinica()
Descripción:	Este método es el encargado de crear la historia clínica al paciente, y de persistir los datos en la base de datos, teniendo en cuenta que el especialista haya insertado algún dato previamente.
Nombre:	CalcularCronologicaPaciente()
Descripción:	Este método calcula la edad cronológica del paciente, teniendo en cuenta la fecha actual y la fecha de nacimiento del mismo.

Nombre: Modl_HistoriaClinica.	
Nombre:	GetId_paciente()
Descripción:	Este método captura el identificador del paciente que fue seleccionado por el especialista para crearle la historia clínica.
Nombre:	SetId_paciente(intid_paciente)
Descripción:	Este método se encarga de tomar los datos del paciente de la hoja frontal.
Nombre:	CalcularEdadMesesPaciente()
Descripción:	Este método se encarga de calcular la edad en meses del paciente a partir de la fecha actual.
Nombre:	Getvariable()
Descripción:	Este método tiene la responsabilidad de devolver el valor del atributo al que le fue realizado la propiedad, y que ha sido declarado en la clase.
Nombre:	Setvariable()
Descripción:	Este método tiene la responsabilidad de modificar los valores de la variable.

Nota: Las restantes clases controladoras incorporadas durante el refinamiento del diseño se encuentran en el Anexo 3.

3.3. Modelo de datos.

El modelo de datos ayuda al ingeniero de software a representar lo siguiente:

- Los objetos (entidades) de datos que el sistema debe entender.
- Los atributos que describen a cada objeto.
- Las relaciones entre los objetos.

“El modelado de los datos emplea el Diagrama Entidad Relación (DER) como base. El DER, se centra solo en los datos, esto hace que proporcione el entendimiento del dominio de la información del

problema. Servirá a los diseñadores del software como base para producir el diseño de la base de datos". (32)

En la figura 2, se muestra el Modelo de Datos, se encuentra agrupado en paquetes con el objetivo de lograr un mayor entendimiento del mismo. En las figuras 3, 4 y 5 se encuentran especificados los paquetes: Historia Clínica, Test PPV y Test PPDL, con sus respectivas tablas y atributos.

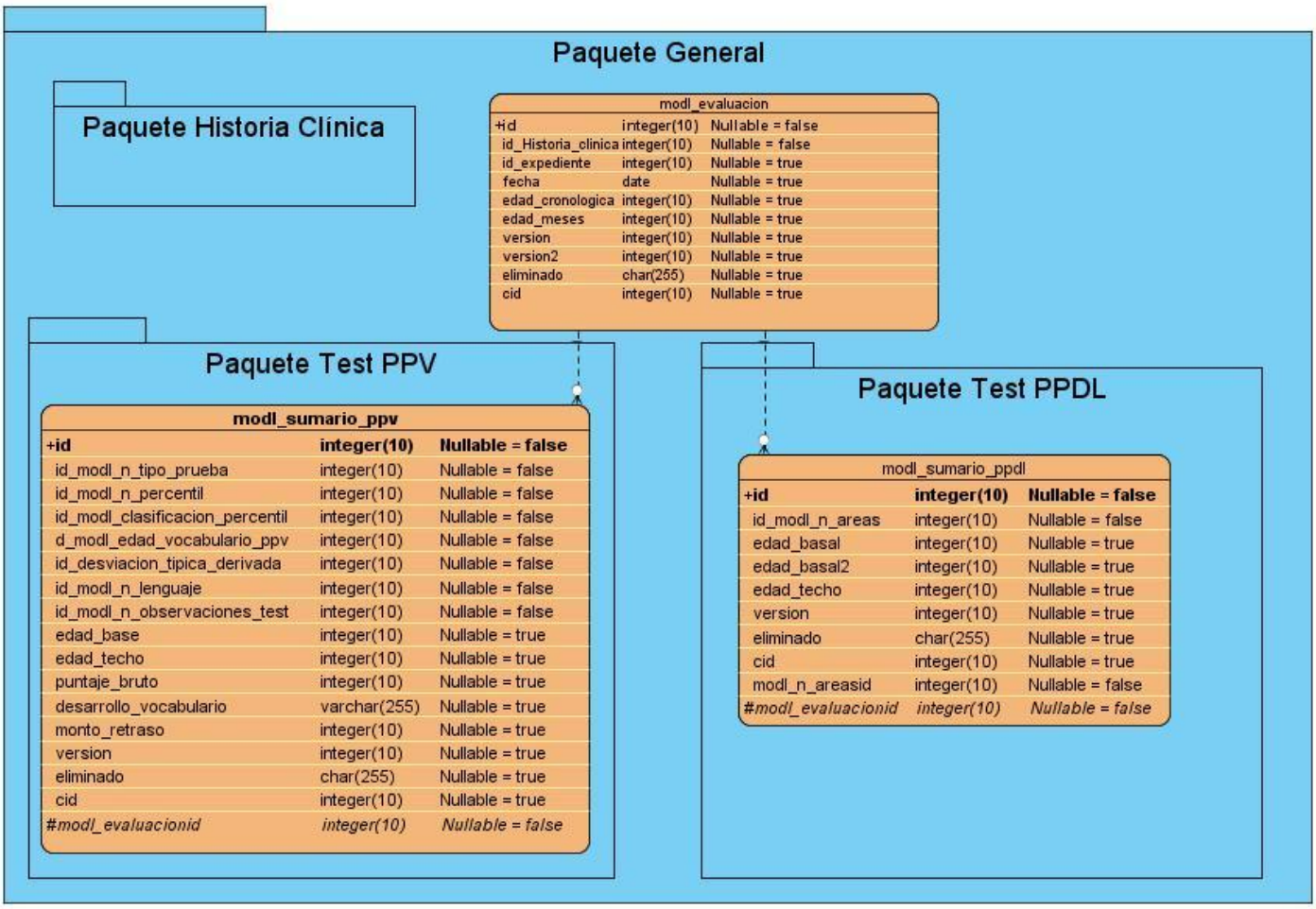


Figura2: Paquete General.

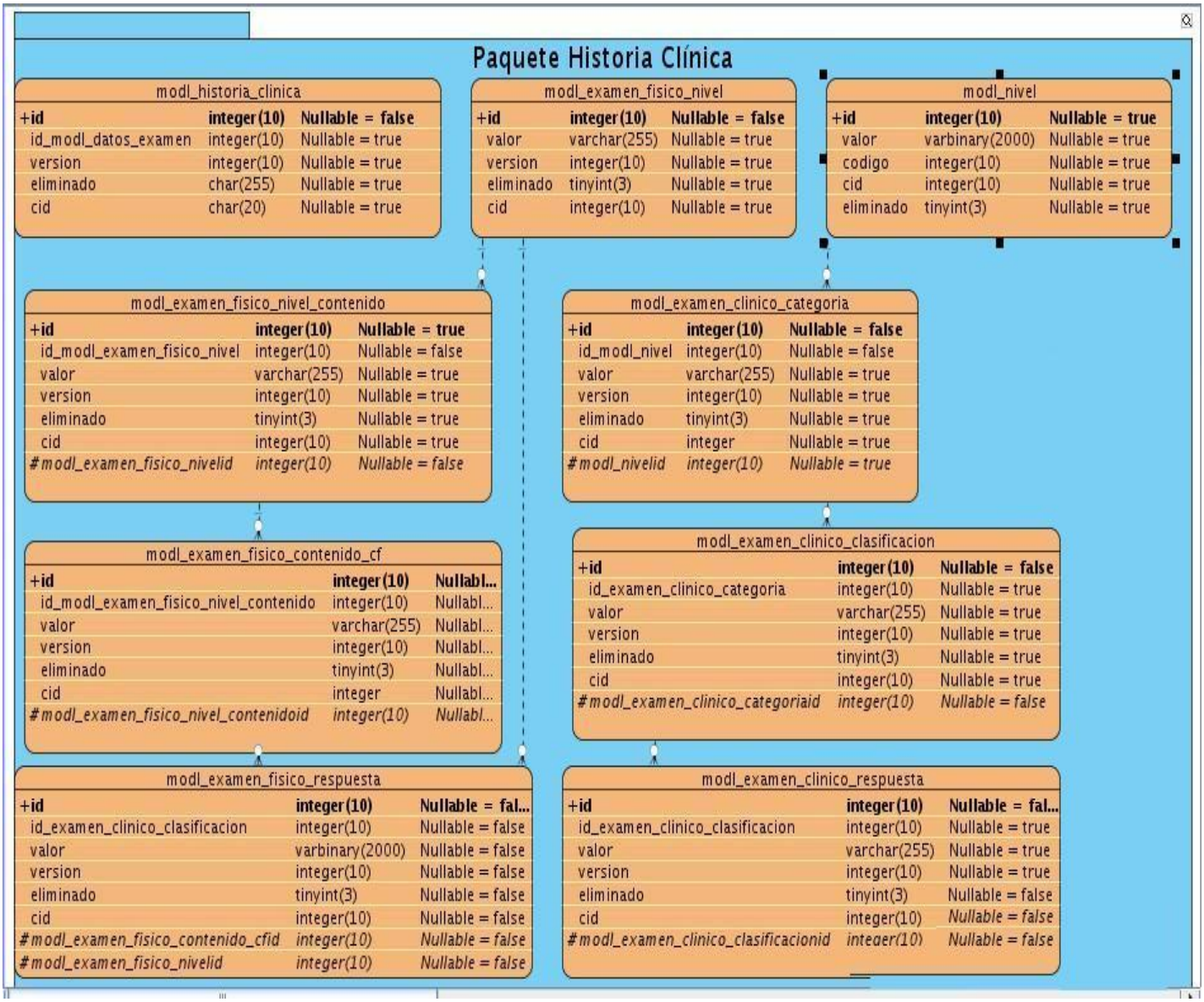


Figura3: Paquete Historia Clínica.

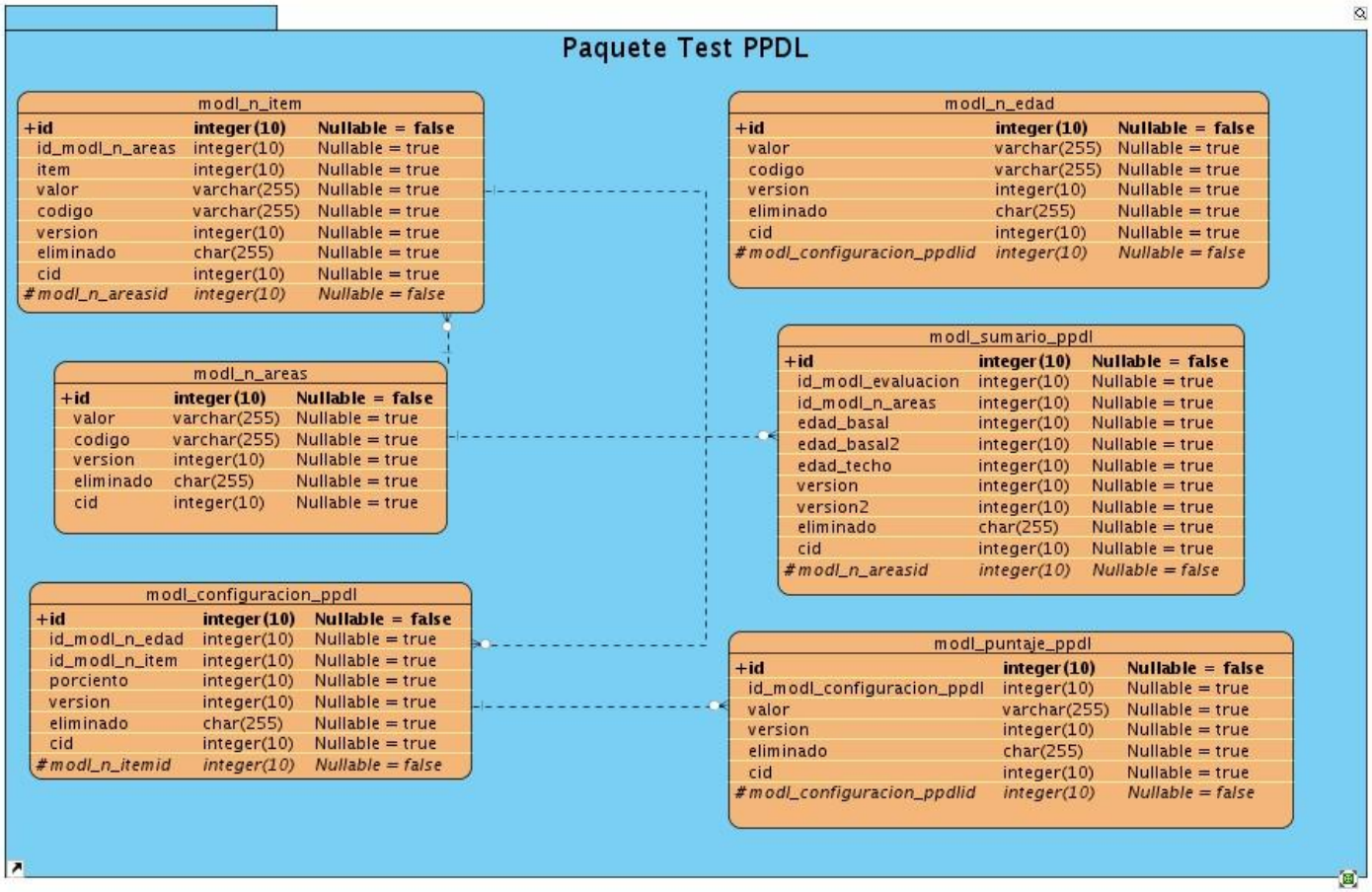


Figura4: Paquete Test PPDL.

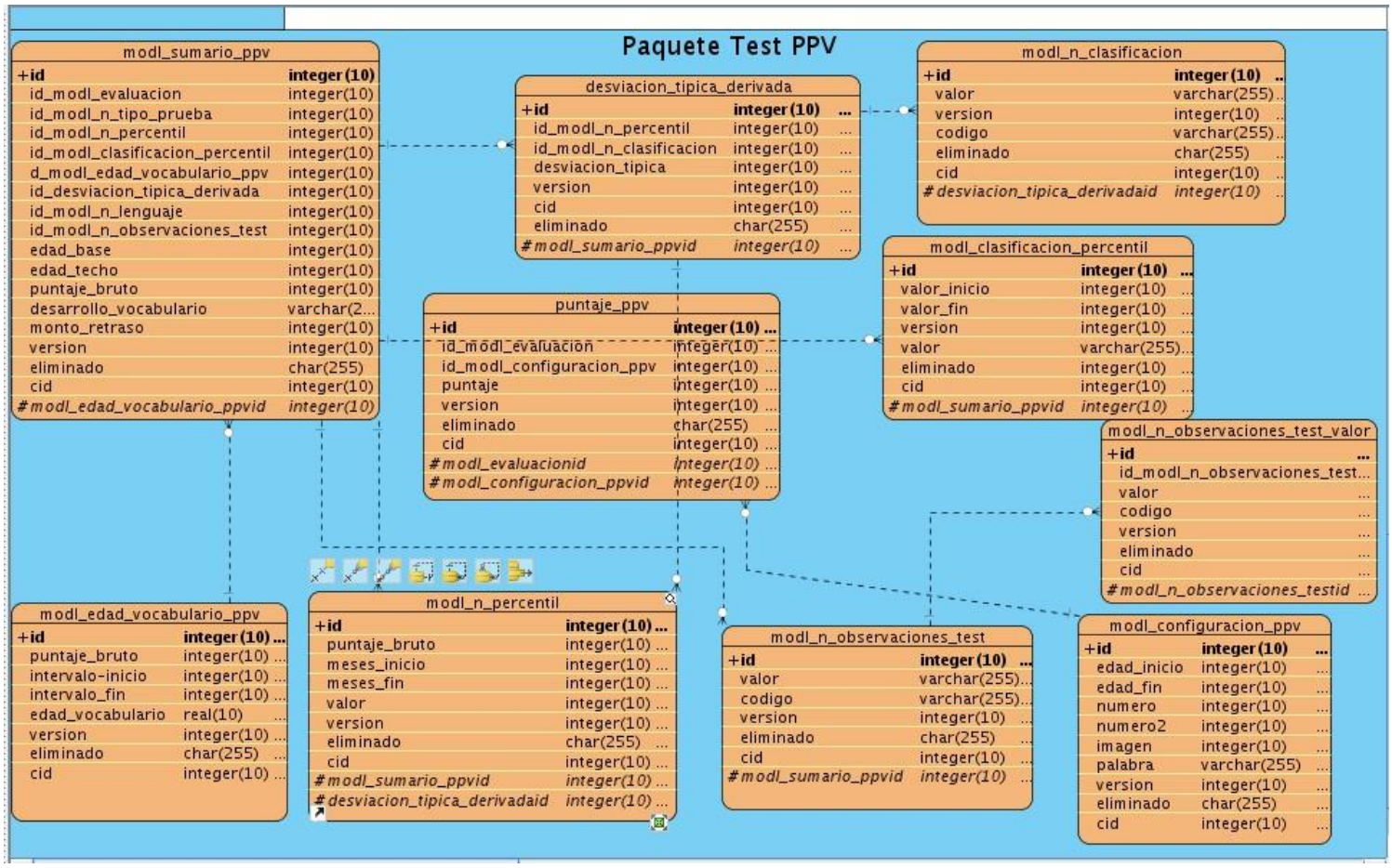


Figura 5: Paquete Test PPV.

En las tablas 5-8, se describen algunas entidades de la base de datos, en las que serán almacenados todos los datos registrados por el sistema. Las restantes tablas pertenecientes al Modelo de Datos se encuentran en el Anexo 4.

Tabla 5: Descripción de la tabla modl_clasificacion_percentil.

Nombre: modl_clasificacion_percentil.		
Descripción: contiene la clasificación según el percentil y los rangos de edades.		
Atributo	**Tipo**	**Descripción**
id_ percentil	Integer	Identificador de la tabla.
valor_inicio	Integer	Valor de inicio del puntaje bruto.
valor_fin	Integer	Valor final del puntaje bruto.
valor	Varchar	Valor del puntaje bruto.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 6: Descripción de la tabla modl_configuracion_ppdl.

Nombre: modl_configuracion_ppdl.		
Descripción: contiene la configuración del test ppdl.		
Atributo	**Tipo**	**Descripción**
id_configuracion	Integer	Identificador de la tabla.
id_modl_n_edad	Integer	Identificador del nomenclador para la edad.
id_modl_n_item	Integer	Identificador del nomenclador para el ítem del test.
porciento	Integer	Porciento que deben cumplir los pacientes a partir de una edad determinada.

Nombre: modl_configuracion_ppdl.		
Descripción: contiene la configuración del test ppdl.		
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 7: Descripción de la tabla modl_configuracion_ppv.

Nombre:modl_configuracion_ppv.		
Descripción: contiene la configuración del test ppv.		
Atributo	**Tipo**	**Descripción**
id_configuracion_ppv	Integer	Identificador de la tabla.
edad_inicio	Integer	Inicio del intervalo de la edad del paciente.
edad_fin	Integer	Fin del intervalo de la edad del paciente.
numero	Integer	Número de la imagen.
imagen	Integer	Imagen que será mostrada al paciente.
palabra	varchar	Palabra asociada a la imagen.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 8: Descripción de la tabla modl_desviacion_tipica_derivada.

Nombre: modl_desviacion_tipica_derivada.		
Descripción: se almacenan los valores de las desviaciones típicas derivadas por puntajes brutos.		
Atributo	**Tipo**	**Descripción**
id_tipico	Integer	Identificador de la tabla.
id_modl_n_percentil	Integer	Identificador del nomenclador percentil.
id_modl_n_clasificacion	Integer	Identificador del nomenclador clasificación.
desviacion_tipica	Integer	Atributo que posee el valor de la desviación típica.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Nota: En las tablas anteriormente presentadas, no existen acentos, ni caracteres extraños, con el objetivo de evitar errores e incompatibilidades. No todos los compiladores los reconocen y además la codificación del fichero de código fuente, al cambiar de una estación a otra, puede provocar afectaciones en su visibilidad.

3.4. Vista de Implementación.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos de código fuente, binarios, o ejecutables. Se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado serán componentes y paquetes, pueden agruparse según un criterio lógico y con vistas a simplificar la implementación. A continuación se muestran los diagramas de componentes por paquetes: paquete vistas, paquete controladoras y paquete modelos.

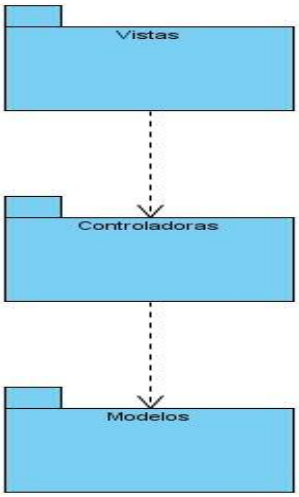


Figura 6: Diagrama de componentes logopedia (paquetes).

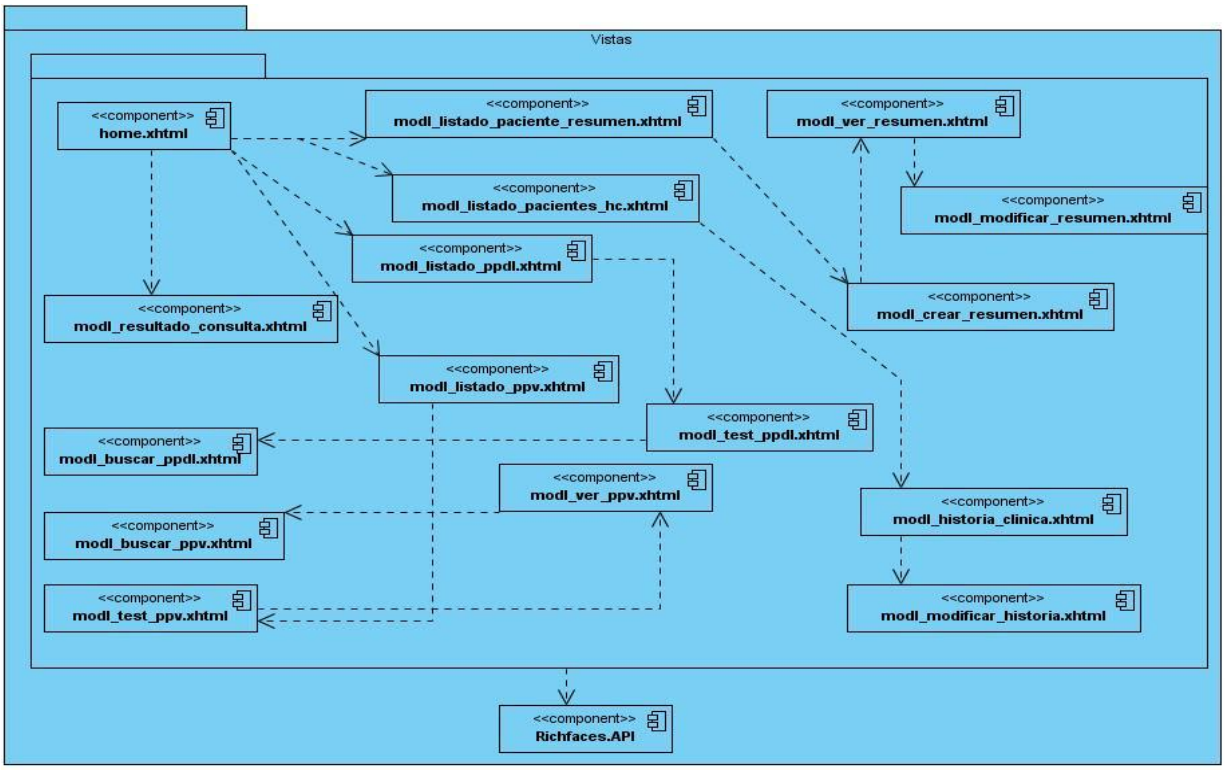


Figura 7: Diagrama de Componentes Paquete Vistas.

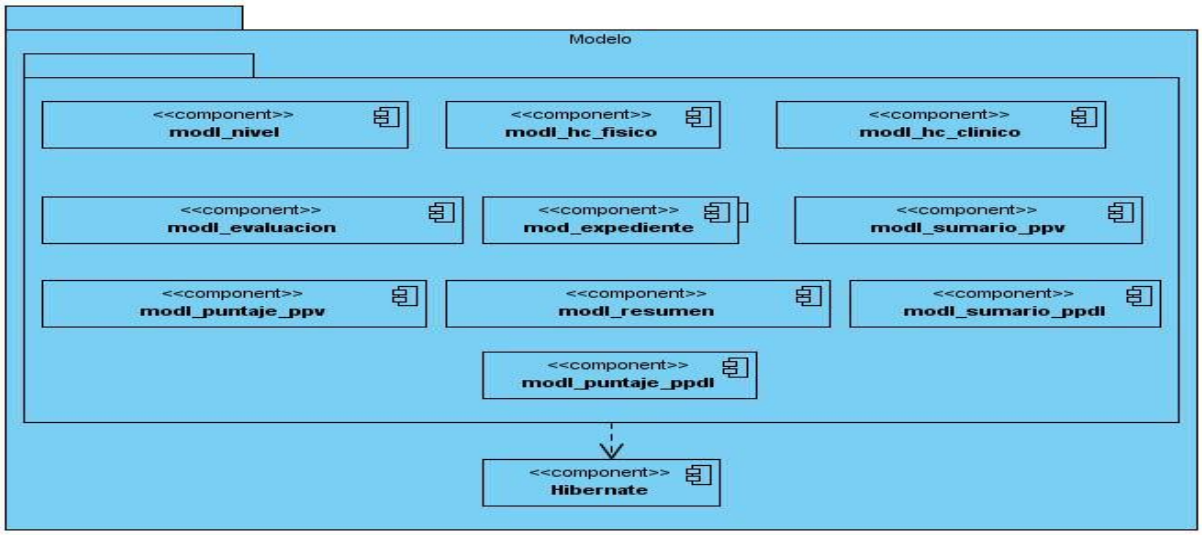


Figura 8: Diagrama de componentes paquete modelos.

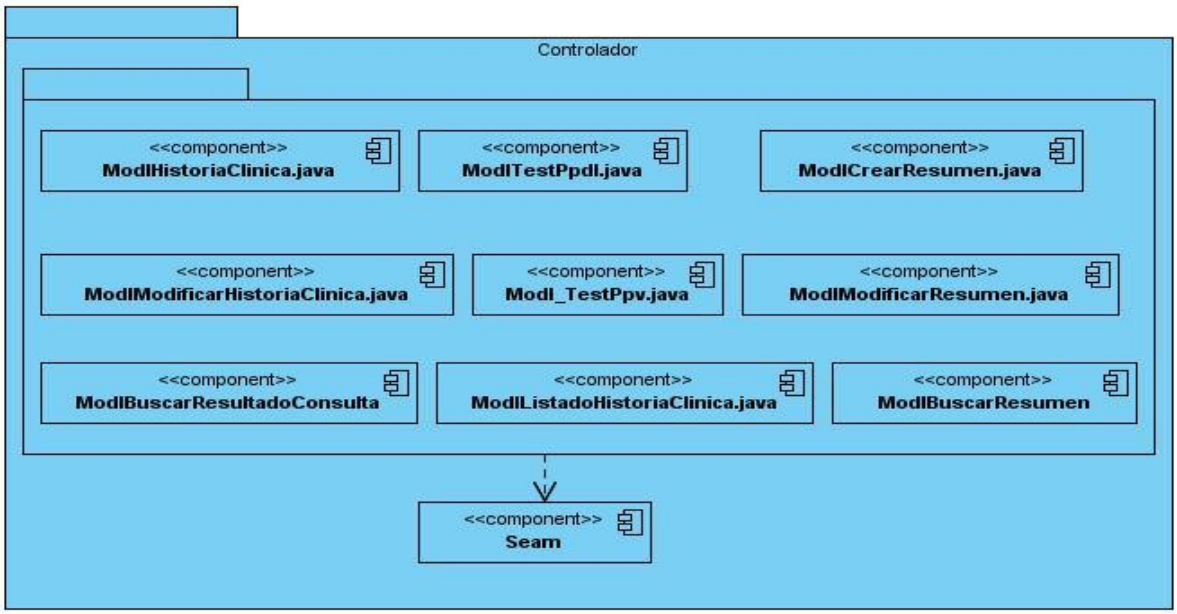


Figura 9: Diagrama de componentes paquete controlador.

En el desarrollo de este capítulo se corroboró que el diseño propuesto por el analista no era óptimo, por lo que se hizo necesaria la inclusión de nuevas clases, métodos y relaciones. El análisis de la función y responsabilidad de las clases controladoras utilizadas en la implementación permitió confirmar el

cumplimiento de los Requerimientos No Funcionales definidos, con los que el sistema debe cumplir, los cuales repercuten directamente en el óptimo funcionamiento y mantenimiento del sistema. La Vista de Implementación es necesaria para una mejor comprensión de las organizaciones y dependencias lógicas entre componentes software, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. El refinamiento de la base de datos es fundamental antes del inicio de la producción de código fuente, pues se evita la pérdida de tiempo y recursos.

CAPÍTULO 4. Validación de la solución propuesta.

El objetivo de este capítulo es describir el proceso de las pruebas de caja negra realizadas al módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños, con el propósito de validar la solución propuesta y encontrar fallas no detectadas hasta entonces.

4.1. Tipos de prueba.

Para desarrollar productos software de calidad, la prueba es una de las tareas más importantes y, cuando se aplica linealmente en el ciclo de vida del producto, desempeña un papel crucial en la gestión de proyectos.

Las pruebas evalúan el producto y permiten determinar si cumple con el objetivo previsto, por lo que es necesario diseñar un plan de pruebas que se adapte y sea coherente con la metodología de desarrollo, que proporcione un enfoque de fácil acceso a la estructura para verificar los requisitos y cuantificar su rendimiento, y que identifique las diferencias entre los resultados previstos y los reales. Es el proceso por medio del cual, se evalúa la correcta interpretación y aplicación de los requisitos especificados.

Cualquier producto de ingeniería se puede probar de dos formas:

Pruebas de caja negra**:** encargadas de comprobar que cada función del software es operativa. Las pruebas de caja negra se llevan a cabo sobre la interfaz del software. Tratan de demostrar que las entradas se manejan de forma adecuada y que se produce el resultado esperado. “Buscan encontrar errores en cinco categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de inicialización y terminación
- Errores de rendimiento”. (33)

Pruebas de caja blanca: “se centran en la estructura lógica interna del software. Se basan en un examen detallado de los procedimientos y caminos lógicos del sistema. Son pruebas que aseguran que la operación

interna se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada". (34)

En la presente investigación el tipo de prueba a utilizar es el de caja negra, debido que el probador se limita a suministrar datos como entrada y estudiar la salida de la aplicación; están especialmente enfocadas a los módulos que van a ser interfaz con el usuario y se apoyan en la especificación de requisitos del software. Además son las definidas por el Departamento de Sistemas Especializados en Salud.

4.2. Casos de Prueba.

Para desarrollar software de calidad y libre de errores, el plan de pruebas y los casos de prueba son de vital importancia. Un buen caso de prueba, es aquel que tiene una alta probabilidad de demostrar un error no descubierto hasta entonces. Un caso de prueba bien diseñado tiene gran posibilidad de llegar a resultados más fiables y eficientes, mejorar el rendimiento del sistema, y reducir los costos.

Un caso de prueba es un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema; sus componentes son:

1. Propósito: descripción del requisito que se está probando.

2. Método: vía que permitirá realizar las pruebas.

3. Versión: versión de la aplicación en prueba, el hardware, el software, el sistema operativo, los archivos de datos, entre otros.

4. Resultados: acciones y resultados esperados o entradas y salidas.

5. Documentación: documentos de la prueba y sus anexos.

El Centro de Calidad para Soluciones Informáticas (Calisoft) definió dos fases de pruebas para el Proceso de Desarrollo de Software de la Universidad, dentro de los cuales se encuentran las Pruebas Internas (desarrolladas dentro del mismo Proyecto, Departamento o Centro) y las Pruebas de Liberación (realizadas por un tercero confiable).

Pruebas Internas: Durante esta fase, el proyecto verifica el resultado de la implementación probando según sea necesaria cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Durante esta fase se deben desarrollar artefactos de prueba como:

Diseños de casos de prueba, Listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.

Pruebas de Liberación: Pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

De igual forma la metodología RUP propone el Flujo de Trabajo Prueba, con el fin de evaluar y valorar la calidad del producto.

Para facilitar el cumplimiento a lo definido por la UCI y lo planteado en la metodología utilizada, se planificaron en el cronograma tareas dirigidas a comprobar la operatividad de las funcionalidades del sistema desarrollado, mediante la realización de pruebas internas en el departamento. Los artefactos definidos para la validación del software fueron los casos de prueba, los cuales se diseñaron y aplicaron con la colaboración del grupo interno de calidad del Departamento SES.

4.2.1 Métodos de prueba.

Para la realización de las pruebas de caja negra existen varios métodos. Entre ellos se encuentran:

Partición equivalente: es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos. El diseño de casos de prueba para la partición equivalente, se basa en la evaluación de las clases de equivalencia.

Análisis de valores límite: consiste en elegir las pruebas que ejecuten los valores límite, con esta técnica se complementa la partición equivalente.

Prueba de comparación: esta técnica se basa en la comparación de salidas de un mismo software pero de sus diferentes versiones.

El método a utilizar en la aplicación de las pruebas de caja negra, es el de partición equivalente, pues reduce el número total de casos de prueba a desarrollar, obteniéndose así en un conjunto manejable que evalúe el software.

Se muestra a continuación el caso de prueba Crear Resumen, en el que se describen las variables válidas e inválidas de entrada y salida en el caso de uso. En la tabla 27, se presentan las distintas secciones en el caso de uso, además de los escenarios por sección y la descripción de la funcionalidad. La tabla 28, muestran las variables usadas en la realización de la funcionalidad. Se cuenta además con la tabla 29:

Descripción de los casos de prueba por sección, que detalla las distintas secciones del CU, y contiene los valores de entrada reales para probar. Además, la contiene la respuesta que debe proporcionar el sistema en cada escenario y el flujo central de navegación para completar al mismo.

Tabla 27: Secciones a probar en el Caso de Uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Adicionar Resumen.	EC 1.1: Adicionar el resumen exitosamente	Se agrega un nuevo resumen con los datos correspondientes.

Tabla 28: Descripción de variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Observaciones.	Campo de texto	SI	Se admiten números, letras y demás caracteres.
2	Tipos de exámenes	Checkbok	SI	El usuario puede seleccionar si desea indicar: estudios auditivos, de neuroimagen, psicométricos, de laboratorio, genéticos, microbiológicos o neurofisiológicos.
8	Edad Base PPV.	Mostrar	SI	Campo para mostrar la edad base calculada en la
9	Edad Techo PPV.	Mostrar	SI	Campo para mostrar la edad techo calculada en la
10	Edad Base PPDL.	Mostrar	SI	Campo para mostrar la edad base calculada en la
11	Edad Techo PPDL.	Mostrar	SI	Campo para mostrar la edad techo calculada en la

Tabla 29: Matriz de Datos.
SC 1. Crear Resumen.

Escenario	Variable 1 *(Observaciones)*	Variable 2 *(Estudios Auditivos)*	Variable 3 *(Estudios de Neuroimagen)*	Variable 4 *(Estudios Psicométricos)*	Variable 5 *(Estudios de Laboratorio)*	Variable 6 *(Estudios Genéticos)*	Variable 7 *(Estudios Microbiológicos)*	Variable 8 *(Estudios Neurofisiológicos)*	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC1.1: Adicionar Resumen exitosamente.	*El estado del paciente es normal para la edad.*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*El sistema adiciona un nuevo Resumen.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*
EC1.2: Existen campos incompletos	*NA*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*El sistema adiciona un nuevo Resumen sin observaciones.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.*

Escenario	Variable 1 *(Observaciones)*	Variable 2 *(Estudios Auditivos)*	Variable 3 *(Estudios de Neuroimagen)*	Variable 4 *(Estudios Psicométricos)*	Variable 5 *(Estudios de Laboratorio)*	Variable 6 *(Estudios Genéticos)*	Variable 7 *(Estudios Microbiológicos)*	Variable 8 *(Estudios Neurofisiológicos)*	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	NA	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*El sistema no inserta datos.*		• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*
	El estado del paciente es normal para la edad.	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen sin examenes complementarios.*		
EC1.2: Existen campos incompletos	*El estado del paciente es normal para la edad.*	*Seleccionado*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen con observaciones estudios auditivos.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*

Escenario	Variable 1 *(Observaciones)*	Variable 2 *(Estudios Auditivos)*	Variable 3 *(Estudios de Neuroimagen)*	Variable 4 *(Estudios Psicométricos)*	Variable 5 *(Estudios de Laboratorio)*	Variable 6 *(Estudios Genéticos)*	Variable 7 *(Estudios Microbiológicos)*	Variable 8 *(Estudios Neurofisiologicos)*	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	El estado del paciente es normal para la edad.	*Seleccionado*	*Seleccionado*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen con observaciones, estudios auditivos y estudios de neuroimagen.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*
	El estado del paciente es normal para la edad.	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen con observaciones, estudios auditivo, estudios de neuroimagen y estudios psicométricos.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*
EC1.2: Existen campos incompletos	*El estado del paciente es normal para la edad.*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Sin seleccionar*	*Sin seleccionar*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen con observaciones, estudios auditivo, estudios de neuroimagen, estudios psicométricos, y estudios de laboratorio.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*

Escenario	Variable 1 *(Observaciones)*	Variable 2 *(Estudios Auditivos)*	Variable 3 *(Estudios de Neuroimagen)*	Variable 4 *(Estudios Psicométricos)*	Variable 5 *(Estudios de Laboratorio)*	Variable 6 *(Estudios Genéticos)*	Variable 7 *(Estudios Microbiológicos)*	Variable 8 *(Estudios Neurofisiológicos)*	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	El estado del paciente es normal para la edad.	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Sin seleccionar*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen con observaciones, estudios auditivo, estudios de neuroimagen, estudios psicométricos, estudios de laboratorio, y estudios genéticos.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen*
	El estado del paciente es normal para la edad.	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Seleccionado*	*Sin seleccionar*	*El sistema adiciona un nuevo Resumen con observaciones, estudios auditivo, estudios de neuroimagen, estudios psicométricos, estudios de laboratorio, estudios genéticos y estudios neurofisiológicos.*	*Satisfactorio*	• *Menú Logopedia.* • *Submenú Crear Resumen.* • *Vínculo Crear Resumen.*

4.3. Resultados de las pruebas realizadas.

Como resultado de la aplicación de las pruebas de caja negra al sistema, se obtuvo un total de 27 no conformidades, de ellas, 10 son significativas y 17 no significativas; a las cuales se les dio solución de forma satisfactoria. En la tabla 30 se muestran las no conformidades detectadas en el caso de uso Crear Resumen.

Tabla 30: Registro de defectos y dificultades detectadas.

Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Crear resumen	1	En el campo nombre admite números y caracteres especiales.	• Módulo Logopedia. • Submenú Crear Resumen.	Implementación.	x	Significativa.		Resuelta	Validar que en el campo nombre no admita números ni caracteres especiales.
	2	En el campo primer apellido admite números y caracteres especiales.	• Módulo Logopedia. • Submenú Crear Resumen.	Implementación	x	Significativa.		Resuelta	Validar que en el campo primer apellido no admita números ni caracteres especiales.

Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Crear resumen	3	Cambiar en nombre de encabezado crear resumen por Realizar resumen.	• Módulo Logopedia. • Submenú Crear Resumen.	Implementación.		No Significativa.		Resuelta	Cambiar el valor del archivo que contiene el texto a mostrar por ‘’Realizar resumen’’.
	4	Eliminar los 2 puntos en los tabs que se encuentran en los nombres de las pruebas.	• Módulo Logopedia. • Submenú Crear Resumen.	Implementación		No Significativa.		Resuelta	Cambiar el valor del archivo, y eliminar los dos puntos.

En el desarrollo de este capítulo, se confirmó que la ejecución de pruebas en el ciclo de vida del software es de vital importancia, pues validan que los requisitos fueron implementados correctamente. Debe haber al menos un caso de prueba por requisito, a menos que un requisito a su vez cuente con requisitos secundarios, permitiendo así que sean examinados por separado. Se diseñaron 15 casos de prueba para el Módulo Logopedia, que permitieron descubrir errores, mejorar el rendimiento del sistema, y reducir los costos. La obtención de un producto que cumpla con las aspiraciones del cliente, solo es posible con el desarrollo eficiente del flujo de trabajo Prueba, pues se confirmó su validez, al comprobar la calidad del software.

CONCLUSIONES

Una vez finalizada la investigación referida a la implementación de los procesos: realizar evaluaciones y crear resumen del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños se arribaron a las siguientes conclusiones:

- El estudio y aplicación del Proceso Unificado de Desarrollo de Software al ciclo de vida del sistema, constituyó una práctica eficiente que permitió la obtención exitosa de los artefactos generados durante el desarrollo del software.
- El estudio de las ventajas y desventajas de las herramientas a utilizar, confirmó reiteradamente que eran las más indicadas para el desarrollo del trabajo, pues son tecnologías libres que ofrecen los beneficios necesarios para la implementación de las soluciones de informática médica.
- El análisis crítico del diseño propuesto por el analista, permitió realizar el refinamiento del diseño y de la base de datos, lo que posibilitó iniciar el flujo de implementación con la mínima cantidad de errores, evitando de esta manera la pérdida de tiempo y recursos.
- La descripción de la arquitectura propuesta por el Departamento de Gestión Hospitalaria del Centro de Informática Médica, permitió el aprovechamiento óptimo de las ventajas de patrón Modelo-Vista-Controlador y viabilizó el desarrollo de un sistema flexible y robusto.
- La realización de los casos de prueba, permitió verificar que los requisitos funcionales fueron implementados correctamente y facilitó la obtención de un producto que cumple con las aspiraciones del cliente.
- El desarrollo de los procesos realizar evaluaciones y crear resumen del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños, facilitará la gestión de la información en el hospital William Soler; así como la aplicación del programa Renacer Contigo, en todos los hospitales del país.

De esta forma, se ha cumplido con el objetivo y las tareas trazadas en el trabajo de diploma, por lo que se obtuvo un sistema informático, que permite automatizar los procesos de gestión de la información de la especialidad de Logopedia del hospital William Soler.

RECOMENDACIONES

Para lograr la continuidad de este trabajo, debido a la importancia que tiene la especialidad de Logopedia para las instituciones hospitalarias, se recomienda:

- Generar reportes estadísticos que permitan mostrar la cantidad de pacientes donde la edad de vocabulario no coincide con la edad cronológica.
- Proponer al responsable del Programa Renacer Contigo, desplegar el SENDN en todos los hospitales pediátricos de Cuba.
- Aplicar al sistema pruebas de caja blanca, con el objetivo de encontrar funciones o procedimientos que nunca son ejecutados por el programa, comprobando que se utilizan todas las estructuras de datos internas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Castro Ruz, Fidel.***Discurso por el 1ro. de mayo. 2001.*

2. **Castro Ruz, Fidel**. La idea esencial es acercar los servicios primarios a los ciudadanos. *Granma.* 2003.

3. **Castro Ruz, Fidel**. *Discurso pronunciado el 24 de Febrero 1960.*

4.Infomed Especialidades. [En línea] [Citado el: 15 de Octubre de 2010.] http://www.sld.cu/sitios/rehabilitacion-logo/temas.php?idv=18920.

5. Iqual ingenieros. [En línea] [Citado el: 5 de Febrero de 2011.] http://iqingenieros.com/inicio/101-p/321-proceso-unificado-de-desarrollo-de-software.html..

6.Ivanex.Sistema de selección de rutas óptimas. [En línea] [Citado el: 6 de Febrero de 2011.] http://ivanex.wikidot.com/metodologia..

7.EcuRed.[En línea] [Citado el: 6 de Febrero de 2011.] http://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software#Pruebas_de_Software..

8. **Pressman, Roger.***Ingeniería de Software. Un enfoque práctico.* 2002.

9.INFORMATION SYSTEM GROUP TIC-194. [En línea] [Citado el: 9 de Febrero de 2011.] http://indalog.ual.es/mtorres/LP/Prueba.pdf..

10. WebShpere Application Server Community Edition. . *International Business Machines Corporation IBM.* [En línea] [Citado el: 16 de Octubre de 2010.] http://publib.boulder.ibm.com/wasce/V2.1.0/es/java-ee-5-certified.html..

11.Jatun. [En línea] [Citado el: 15 de Octubre de 2010.] http://www.jatun.com/web/company/training/javaee5..

12.Junta de Andalucia. [En línea] [Citado el: 2010 de Octubre de 2010.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces..

13.RedIRIS. [En línea] [Citado el: 25 de Octubre de 2010.] https://forja.rediris.es/docman/view.php/585/1067/HIBERNATE.doc..

14.**Herrera, Cristhian.** Adictos al Trabajo. [En línea] [Citado el: 28 de Octubre de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring..

15. **Fajardo Araujo, Solainy y Guerra Ferrer, Maikel .***Módulo de Fisiatría del Sistema de Evaluación del Neurodesarrollo en Niños.* 2009.

16. Coplec. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.coplec.org/?q=book/export/html/240..

17.**Carvajal, Carlos.** Departamento de Electrónica de la Universidad Técnica Federico Santa María. [En línea][Citado el: 5 de Noviembre de 2010.] http://www2.elo.utfsm.cl/~elo326/Presentaciones/Ronda2/JPA.pdf..

18.Sun Microsystems. [En línea] [Citado el: 12 de Octubre de 2010.] http://java.sun.com/javaee/technologies/javaee5.jsp..

19. **JACOBSON, Ivar , BOOCH, Grady y RUMBAUGH, James .***El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley, 1999.

20.Manuales de Ayuda.com. [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.manualesdeayuda.com/manuales/bases-de-atos/postgresql/caracteristicas-de-postgresql-01844.html..

21.Mitecnologico. [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.mitecnologico.com/Main/HerramientasCase.

22.Descargas de Software. [En línea] [Citado el: 5 de Diciembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p..

23. Eclipse. [En línea] [Citado el: 12 de Noviembre de 2010.] http://plataformaeclipse.com..

24.WorldLingo. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.worldlingo.com/ma/enwiki/es/Eclipse_%28software%29..

25.Entérate, Internet, Cómputo y Telecomunicaciones . [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.enterate.unam.mx/Articulos/2006/febrero/arquitec.htm..

26.Mitecnologico. [En línea] [Citado el: 10 de Enero de 2011.] http://www.mitecnologico.com/Main/LaNormaIsoIec9126..

27.Glosario.net. [En línea] [Citado el: 10 de Febrero de 2011.] http://tecnologia.glosario.net/terminos-tecnicos-internet/fiabilidad-681.html..

28. Buenas Tareas, una exclusiva Base de Datos para los estudiantes. [En línea] [Citado el: 6 de Febrero de 2011.] http://www.buenastareas.com/ensayos/Arquitectura-Del-Software/922875.html..

29.Carloscar7. [En línea] [Citado el: 15 de Enero de 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=9&ved=0CEsQFjAI&url=http%3A%2F%2Fcarloscar7.files.wordpress.com%2F2008%2F02%2Fcarlos-david-carballo-espana.doc&rct=j&q=Consiste%20en%20un%20conjunto%20de%20patrones%20y%20abstracciones%20coherentes%20que%20p.

30. Buen Master.com. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.buenmaster.com/?a=536.

31.Ciberhabit.Cuidad de la Informática.Museo. [En línea] [Citado el: 12 de Enero de 2011.] http://www.inegi.gob.mx/inegi/contenidos/espanol/ciberhabitat/museo/cerquita/redes/seguridad/intro.htm..

32.Scribd.com. [En línea] [Citado el: 10 de Enero de 2011.] http://www.scribd.com/doc/19808824/diagramas-de-despliegue-2222.

33. Visual Basic. [En línea] [Citado el: 10 de Febrero de 2011.] http://vbasic.gdsweb.com.ar/intro.htm...

34.Modelado de Sistemas con UML. [En línea] [Citado el: 8 de Marzo de 2011.] http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html..

BIBLIOGRAFÍA

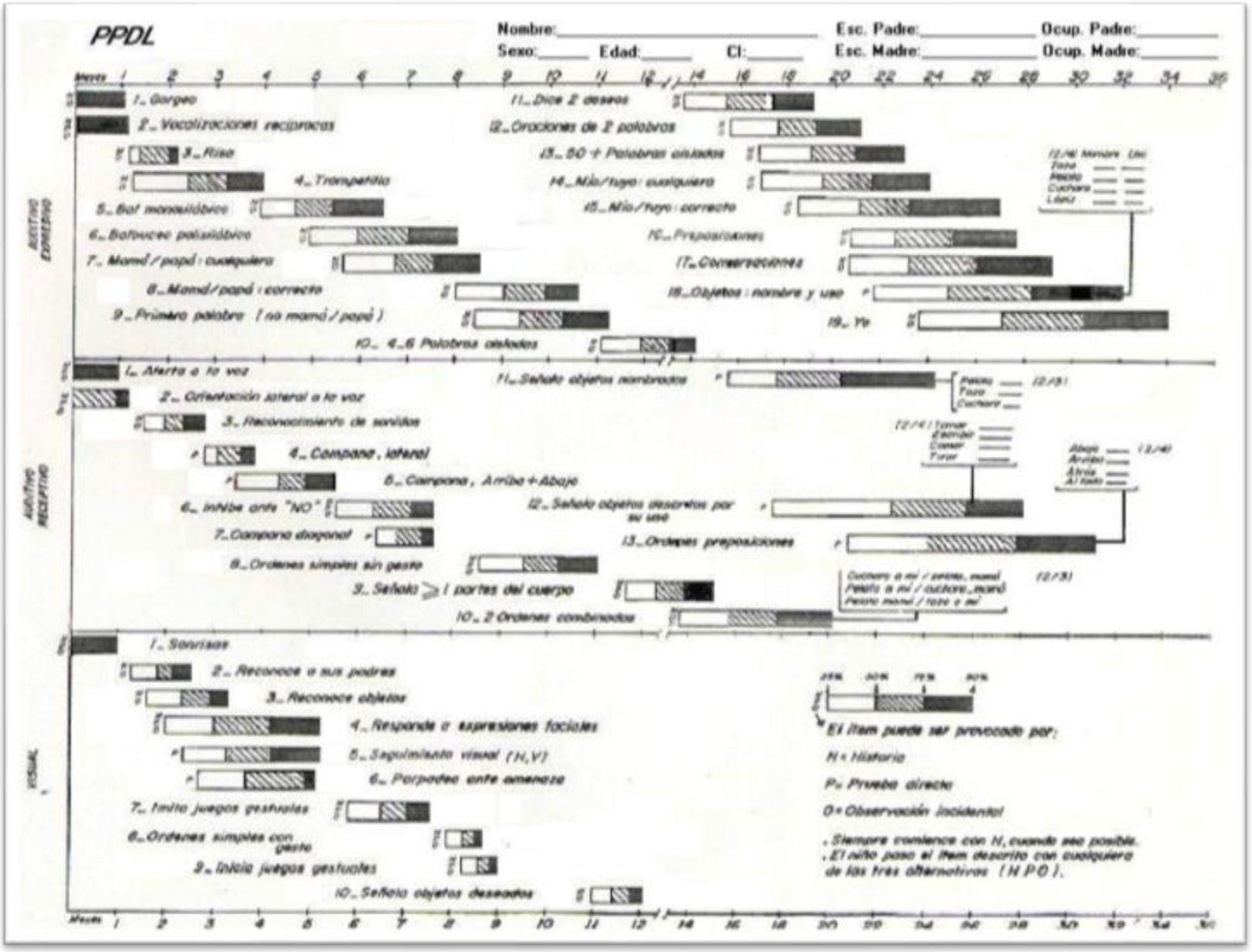
1. Buen Master.com. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.buenmaster.com/?a=536.
2. Buenas Tareas, una exclusiva Base de Datos para los estudiantes. [En línea] [Citado el: 6 de Febrero de 2011.] http://www.buenastareas.com/ensayos/Arquitectura-Del-Software/922875.html.
3. Carloscar7. [En línea] [Citado el: 15 de Enero de 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=9&ved=0CEsQFjAI&url=http%3A%2F%2Fcarloscar7.files.wordpress.com%2F2008%2F02%2Fcarlos-david-carballo-espana.doc&rct=j&q=Consiste%20en%20un%20conjunto%20de%20patrones%20y%20abstracciones%20coherentes%20que%20p.
4. Carvajal, Carlos. Departamento de Electrónica de la Universidad Técnica Federico Santa María. [En línea][Citado el: 5 de Noviembre de 2010.] http://www2.elo.utfsm.cl/~elo326/Presentaciones/Ronda2/JPA.pdf.
5. Castro Ruz, Fidel Discurso pronunciado el 24 de Febrero 1960.
6. Castro Ruz, Fidel. Discurso por el 1ro. de mayo. 2001.
7. Castro Ruz, Fidel. La idea esencial es acercar los servicios primarios a los ciudadanos. Granma. 2003.
8. Ciberhabit. Cuidad de la Informática. Museo. [En línea] [Citado el: 12 de Enero de 2011.] http://www.inegi.gob.mx/inegi/contenidos/espanol/ciberhabitat/museo/cerquita/redes/seguridad/intro.htm.
9. Coplec.[En línea] [Citado el: 30 de Octubre de 2010.] http://www.coplec.org/?q=book/export/html/240.
10. Descargas de Software. [En línea] [Citado el: 5 de Diciembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.
11. Eclipse BIRT. Eclipse Foundation. [En línea] [Citado el: 26 de Enero de 2011.] http://www.eclipse.org/birt/phoenix
12. Eclipse. [En línea] [Citado el: 12 de Noviembre de 2010.] http://plataformaeclipse.com.
13. EcuRed. [En línea] [Citado el: 6 de Febrero de 2011.] http://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software#Pruebas_de_Software.

14. Entérate, Internet, Cómputo y Telecomunicaciones. [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.enterate.unam.mx/Articulos/2006/febrero/arquitec.htm.
15. Fajardo Araujo, Solainy y Guerra Ferrer, Maikel .Módulo de Fisiatría del Sistema de Evaluación del Neurodesarrollo en Niños. 2009.
16. Glosario.net. [En línea] [Citado el: 10 de Febrero de 2011.] http://tecnologia.glosario.net/terminos-tecnicos-internet/fiabilidad-681.html.
17. Gobierno Federal SEP. [En línea] [Citado el: 13 de Febrero de 2011.] http://www.itlalaguna.edu.mx/academico/carreras/sistemas/ingsofware1/UNIDAD5.pdf.
18. Heffelfinger, David. Jasper Reports 3.5 for Java Developers. Birmingham: Packt Publishing. 2009.
19. Herrera, Cristhian. [En línea] [Citado el: 28 de Octubre de 2010.] http://www.tutoriales.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring.
20. Infomed Especialidades. [En línea] [Citado el: 15 de Octubre de 2010.] http://www.sld.cu/sitios/rehabilitacion-logo/temas.php?idv=18920.
21. INFORMATION SYSTEM GROUP TIC-194. [En línea] [Citado el: 9 de Febrero de 2011.] http://indalog.ual.es/mtorres/LP/Prueba.pdf.
22. Iqual ingenieros. [En línea] [Citado el: 5 de Febrero de 2011.] http://iqingenieros.com/inicio/101-p/321-proceso-unificado-de-desarrollo-de-software.html.
23. Ivanex. Sistema de selección de rutas óptimas. [En línea] [Citado el: 6 de Febrero de 2011.] http://ivanex.wikidot.com/metodologia.
24. JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH, James .El Lenguaje Unificado de Modelado. s.l.: Addison Wesley, 1999.
25. Jatun. [En línea] [Citado el: 15 de Octubre de 2010.] http://www.jatun.com/web/company/training/javaee5.
26. Junta de Andalucia. [En línea] [Citado el: 2010 de Octubre de 2010.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces..
27. Manuales de Ayuda.com. [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.manualesdeayuda.com/manuales/bases-de-atos/postgresql/caracteristicas-de-postgresql-01844.html.
28. Microsoft. SQL Server 2008: Reporting. Microsoft.[En línea] [Citado el: 28 de Enero de 2011.] http://www.microsoft.com/sqlserver/2008/en/us/reporting.aspx.

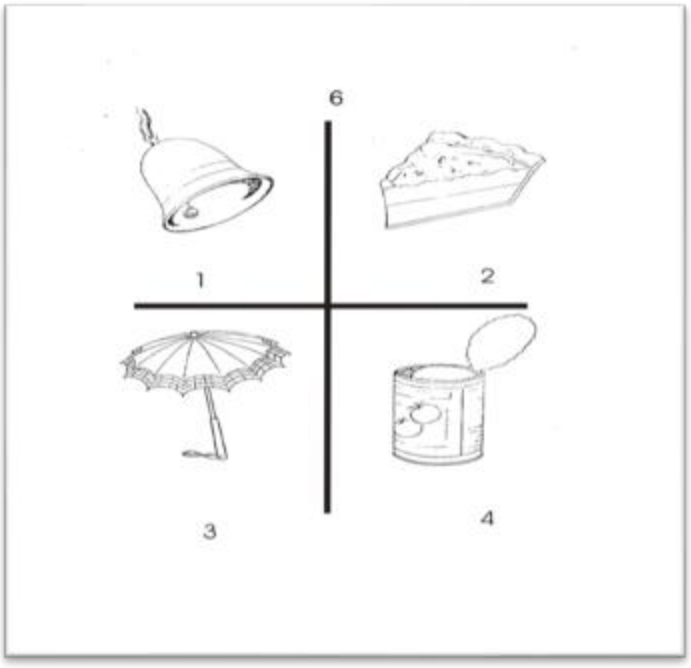
29. Mitecnologico. [En línea] [Citado el: 10 de Enero de 2011.] http://www.mitecnologico.com/Main/LaNormaIsoIec9126.
30. Mitecnologico. [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.mitecnologico.com/Main/HerramientasCase.
31. Modelado de Sistemas con UML. [En línea] [Citado el: 8 de Marzo de 2011.] http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.
32. Pressman, Roger. Ingeniería de Software. Un enfoque práctico. 2002.
33. RedIRIS. [En línea] [Citado el: 25 de Octubre de 2010.] https://forja.rediris.es/docman/view.php/585/1067/HIBERNATE.doc.
34. Scribd.com. [En línea] [Citado el: 10 de Enero de 2011.] http://www.scribd.com/doc/19808824/diagramas-de-despliegue-2222.
35. Sobre PostreSQL. PostgreSQL-es.org. [En línea] [Citado el: 7 de Febrero de 2011.] http://www.postgresql-es.org/sobre_postgresql.
36. Sun Microsystems. [En línea] [Citado el: 12 de Octubre de 2010.] http://java.sun.com/javaee/technologies/javaee5.jsp.
37. The Apache Software Foundation. . The Apache License, Version 2.0. Apache Foundation. [En línea] [Citado el: 10 de Febrero de 2011.] http://httpd.apache.org/docs/2.0/es/license.html.
38. Visual Basic. [En línea] [Citado el: 10 de Febrero de 2011.] http://vbasic.gdsweb.com.ar/intro.htm.
39. WebShpere Application Server Community Edition. International Business Machines Corporation IBM. [En línea] [Citado el: 16 de Octubre de 2010.] http://publib.boulder.ibm.com/wasce/V2.1.0/es/java-ee-5-certified.html.
40. WorldLingo. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.worldlingo.com/ma/enwiki/es/Eclipse_%28software%29.

ANEXOS

Anexo1: Áreas del Test PPDL.



Anexo 2: Lámina Test PPV



Anexo 3:

Tabla 2: Descripción de la clase ModlModificarHistoriaClinica.

Nombre: ModlModificarHistoriaClinica.	
Tipo de clase:controladora.	
Atributos	**Tipos**
Craneo	String
Listar_craneo	List<String>
Desc_craneo	String
Facie	String
Listar_facie	List<String>
Atributos	**Tipos**

Nombre: ModlModificarHistoriaClinica.	
Atributos	**Tipos**
Desc_facie	String
Faringe	String
Listar_faringe	List<String>
Desc_faringe	String
Torax	String
Listar_torax	List<String>
Desc_torax	String
Abdomen	String
Listar_abdomen	List<String>
Desc_abdomen	String
Columna	String
Listar_columna	List<String>
Desc_columna	String
Desc_estadio_puberal	String
Desc_otros	String
Labios	String
Listar_labios	List<String>
Desc_labios	String
Listar_abdomen	List<String>

Nombre: ModlModificarHistoriaClinica.	
Atributos	**Tipos**
Desc_abdomen	String
Columna	String
Listar_columna	List<String>
Paladar_oseo	String
Listar_paladar_oseo	List<String>
Desc_paladar_oseo	String
Paladar_blando	String
Listar_paladar_blando	List<String>
Desc_paladar_blando	String
sOclusion_dentaria	String
Listar_oclusion_dentaria	List<String>
Desc_oclusion_dentaria	String
Relacion_maxilar	String
Listar_relacion_maxilar	List<String>
Desc_relacion_maxilar	String
Mandibula	String
Listar_relacion_mandibula	List<String>
Desc_mandibula	String
Maxilar	String
Listar_maxilar	List<String>

Nombre: ModlModificarHistoriaClinica.	
Atributos	**Tipos**
Desc_maxilar	String
Laringe_sigue	String
Listar_laringe_sigue	List<String>
Laringe_desplaza	String
Listar_laringe_desplaza	List<String>
Desc_indirecta	String
Cuello_inspeccion	String
Expontaneo	String
Listar_expontaneo	List<String>
Expcuantita_impresiona	String
Lstar_expcuantita_impresiona	List<String>
Expcuantita_referido	String
Listar_expcuantita_referido	List<String>
Expresivo_cualit_numero_palabras	String
H_vocales_a	String
Check_h_vocales_a	boolean
H_vocales_e	String
Check_h_vocales_e	boolean
H_vocales_i	String
Check_h_vocales_i	boolean

Nombre: ModlModificarHistoriaClinica.	
Atributos	**Tipos**
H_vocales_o	String
Check_h_vocales_o	boolean
H_vocales_u	String
Check_h_vocales_u	boolean
Para cada responsabilidad:	
Nombre:	CargarDatos()
Descripción:	Este método es el encargado de cargar los datos que seleccionó el especialista, previamente cuando creó la historia clínica logofoniátrica
Nombre:	ModificarDatos()
Descripción:	Este método modifica los datos que el especialista necesite cambiar después de creada la historia clínica logofoniátrica.
Nombre:	getIdPaciente()
Descripción:	Este método captura el identificador del paciente que fue seleccionado por el especialista para crearle la historia clínica.
Nombre:	setIdPaciente(intid_paciente)
Descripción:	Este método se encarga de tomar los datos del paciente de la hoja frontal.

Tabla 3: Descripción de la clase ModlControladoraTestPpv.

Nombre: ModlControladoraTestPpv.	
Tipo de clase: controladora.	
Atributos	**Tipos**
id_paciente_ppv	Integer
aux	ModlConfiguracionPpv
aux1	ModlConfiguracionPpv
hoja	HojaFrontal
imagen_a_mostrar	List<Integer>
edadmeses	int
techo	boolean
base	boolean
contadorAcertadas	int
contadorFallos	int
edadBase	Integer
edadTecho	Integer
contadorIntentos	int
cbx_respuesta	String
listado_imagenes	List<ModlConfiguracionPpv>
ipg_mostrar	Integer
eval	boolean

Nombre: ModlControladoraTestPpv	
Atributos	**Tipos**
dir_imagen	String
palabraAsociada	String
num_mostrar	Integer
figura_mostrar	Integer
temp	Integer
cbx_calificacion_evaluacion	String
resul_desarrollo_vocabulario	String
graficar	boolean
norma_edad	boolean
monto_retraso	Integer
resul_edad_vocabulario	Integer
norma_percentil	boolean
resul_percentil	Integer
puntucion_tipica_derivada	boolean
resul_ci	Integer
iInt_ci_inicio	Integer
int_ci_fin	Integer
iuntaje_bruto	Integer
edad_cronologica	Integer
num_figura_incorrecta	Integer

Nombre: ModlControladoraTestPpv	
Atributos	**Tipos**
fallos_totales	Integer
obs_calidad	String
listar_obs_calidad	List<String>
obs_cantidad	String
listar_obs_cantidad	List<String>
obs_inteligibilidad	String
listar_obs_inteligibilidad	List<String>
obs_num_ejemplos	String
listar_obs_num_ejemplos	List<String>
obs_rapport	String
listar_obs_rapport	List<String>
obs_adivinanzas	String
listar_obs_adivinanzas	List<String>
obs_rapidez	String
listar_obs_rapidez	List<String>
obs_atencion	String
listar_obs_atencion	List<String>
obs_timidez	String
listar_obs_timidez	List<String>
obs_esfuerzo	String

Nombre: ModlControladoraTestPpv	
Para cada responsabilidad:	
Nombre:	ActualizarPuntajePpv(ModlConfiguracionPpvconfiguracion)
Descripción:	Este método es el encargado de llenar la lista que contendrá los datos de una parte del examen y que posteriormente se insertará en la base de datos.
Nombre:	RegistrarEvaluacion()
Descripción:	Este método calcula la edad base y la edad techo del paciente.
Nombre:	GetId_paciente()
Descripción:	Este método captura el identificador del paciente que fue seleccionado por el especialista para realizarle el examen.
Nombre:	SetId_paciente(intid_paciente)
Descripción:	Este método se encarga de tomar los datos del paciente de la hoja frontal.
Nombre:	CargarListadoImagenes()
Descripción:	Este método se encarga de cargar el listado de imágenes que será mostrado al paciente a partir de su edad cronológica.
Nombre:	CargarResultadosFinales()
Descripción:	Este método tiene la responsabilidad de ubicar al paciente en una clasificación determinada, en dependencia de los resultados obtenidos durante la identificación de las imágenes.
Nombre:	InsertarPuntajeBd()
Descripción:	Este método tiene la responsabilidad de insertar los datos en la tabla modl_puntaje_ppv

Anexo 4:

Tabla 9: Descripción de la tabla modl_edad_vocabulario**.**

Nombre: modl_edad_vocabulario.		
Descripción:recoge la edad de vocabulario del paciente.		
Atributo	**Tipo**	**Descripción**
id_edad_vocabulario	Integer	Identificador de la tabla.
edad	Integer	Edad del paciente.
intervalo_inicio	Integer	Inicio del intervalo de la edad del paciente.
intervalo_fin	Integer	Fin del intervalo de la edad del paciente.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
Atributo	**Tipo**	**Descripción**
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 10: Descripción de la tabla modl_edad_vocabulario_ppv.

Nombre: modl_edad_vocabulario_ppv.		
Descripción:almacena la edad de vocabulario del paciente que se va a realizar el test ppv.		
Atributo	**Tipo**	**Descripción**
id_paciente_ppv	Integer	Identificador de la tabla.
puntaje_bruto	Integer	Atributo que recoge la edad techo menos la cantidad de errores.

Nombre: modl_edad_vocabulario_ppv.		
Atributo	**Tipo**	**Descripción**
edad_vocabulario	real	Edad de vocabulario que tiene el paciente.
intervalo_inicio	Integer	Inicio del intervalo de edad del paciente.
intervalo_fin	Integer	Fin del intervalo de edad del paciente.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 11: Descripción de la tabla modl_evaluacion.

Nombre: modl_evaluacion		
Descripción:almacena todos los datos cuando se realiza una evaluación.		
Atributo	**Tipo**	**Descripción**
id_evaluacion	Integer	Identificador de la tabla.
id_expediente	Integer	Identificador de la tabla expediente.
fecha	date	Atributo que recoge la fecha del día que se realizó la consulta.
edad_cronologica	Integer	Atributo que recoge la edad cronológica del niño.
edad_meses	Integer	Atributo que recoge la edad en meses del niño.
version	Integer	Atributo para persistir o actualizar la entidad.

Nombre: modl_evaluacion		
Atributo	**Tipo**	**Descripción**
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 12: Descripción de la tabla modl_examen_clinico_categoria.

Nombre: modl_examen_clinico_categoria.		
Descripción:registra las categorías del examen clínico.		
Atributo	**Tipo**	**Descripción**
id_examen_clinico	Integer	Identificador de la tabla.
id_modl_nivel	Integer	Identificador de la tabla nivel.
Atributo	**Tipo**	**Descripción**
valor	varchar	Atributo que tiene el nombre de la categoría.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 13: Descripción de la tabla modl_examen_clinico_clasificacion.

Nombre: modl_examen_clinico_clasificacion.		
Descripción:registra la clasificación del examen clínico.		
Atributo	**Tipo**	**Descripción**
id_clasificacion_examen_clinico	Integer	Identificador de la tabla.
id_modl_examen_clinico_categoria	Integer	Identificador de la tabla examen clínico categoría.
valor	varchar	Atributo que tiene el nombre de la clasificación.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 14: Descripción de la tabla modl_examen_comp.

Nombre: modl_examen_comp.		
Descripción: registra los datos del examen complementario.		
Atributo	**Tipo**	**Descripción**
id_examen_complementario	Integer	Identificador de la tabla.
valor	varchar	Atributo que tiene el nombre de los exámenes complementarios que se realizan.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido

		eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 15: Descripción de la tabla modl_examen_fisico_contenido_cf.

Nombre: modl_examen_fisico_contenido_cf.		
Descripción: contiene la clasificación del contenido del examen físico.		
Atributo	**Tipo**	**Descripción**
id_contenido_fisico	Integer	Identificador de la tabla.
id_modl_examen_fisico_nivel_contenido	Integer	Identificador de la tabla que tiene el contenido del examen.
valor	varchar	Atributo que tiene nombre de clasificación del contenido del examen físico.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 16: Descripción de la tabla modl_examen_fisico_nivel.

Nombre: modl_examen_fisico_nivel.		
Descripción: contiene el nivel del examen físico.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_nivel	Integer	Identificador de la tabla.
valor	varchar	Atributo que contiene el nivel del examen físico.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario fue eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 17: Descripción de la tabla modl_examen_fisico_nivel_contenido.

Nombre: modl_examen_fisico_nivel_contenido.		
Descripción: contiene los elementos del examen físico.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_nivel_contenido	Integer	Identificador de la tabla.
id_modl_examen_fisico_nivel	Integer	Identificador de la tabla que contiene el nivel del examen.
valor	varchar	Atributo que tiene el contenido del examen físico.
version	Integer	Atributo para persistir o actualizar la entidad.

Nombre: modl_examen_fisico_nivel_contenido.		
Atributo	**Tipo**	**Descripción**
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 18: Descripción de la tabla modl_examen_fisico_respuesta.

Nombre: modl_examen_fisico_respuesta.		
Descripción: contiene las respuestas asociadas al examen físico.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_examen_fisico_nivel	Integer	Identificador de la tabla que contiene el nivel del examen.
id_modl_examen_fisico_contenido_cf	Integer	Identificador de la tabla que contiene la clasificación del contenido del examen físico.
valor	varchar	Atributo que tiene las posibles respuestas del examen físico.
version	Integer	Atributo para persistir o actualizar la entidad
eliminado	Boolean	Verifica si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 19: Descripción de la tabla modl_hc_examen _comp.

Nombre: modl_hc_examen _comp.		
Descripción: contiene la historia clínica del examen complementario.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
id_modl_examen_comp	Integer	Identificador de la tabla que contiene el examen complementario.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 20: Descripción de la tabla modl_historia_clinica_log_clinico.

Nombre: modl_historia_clinica_log_clinico.		
Descripción: contiene la historia clínica del examen clínico.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
id_modl_examen_clinico_respuesta	Integer	Identificador de la tabla que contiene las respuestas asociadas al examen clínico.
valor	varchar	Atributo que recoge si fue seleccionado algún valor en la historia clínica.

Nombre: modl_historia_clinica_log_clinico.		
Atributo	**Tipo**	**Descripción**
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 21: Descripción de la tabla modl_historia_clinica_log_fisico.

Nombre: modl_historia_clinica_log_fisico.		
Descripción: contiene la historia clínica del examen físico.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
id_modl_examen_fisico_respuesta	Integer	Identificador de la tabla que contiene las respuestas asociadas al examen físico.
valor	varchar	Atributo que recoge las descripciones de la historia clínica.
version	Integer	Atributo para persistir o actualizar la entidad
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 22: Descripción de la tabla modl_nivel.

Nombre: modl_nivel.		
Descripción: contiene los niveles del examen clínico.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
valor	varchar	Atributo que contiene los niveles del examen clínico.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 23: Descripción de la tabla modl_puntaje_ppdl.

Nombre: modl_puntaje_ppdl.		
Descripción: contiene el puntaje del test ppdl.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_configuracion_ppdl	Integer	Identificador de la tabla que contiene la configuración del test ppdl.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
valor	varchar	Atributo que contiene el puntaje del test.
version	Integer	Atributo para persistir o actualizar la entidad.

Nombre: modl_puntaje_ppdl.		
Atributo	**Tipo**	**Descripción**
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 24: Descripción de la tabla modl_puntaje_ppv.

Nombre: modl_puntaje_ppv.		
Descripción: contiene el puntaje del test ppv.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
id_modl_configuracion_ppv	Integer	Identificador de la tabla que contiene la configuración del test ppv.
puntaje	Integer	Atributo que contiene la puntuación que se le asigna la paciente.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 25: Descripción de la tabla modl_sumario_ppdl.

Nombre: modl_sumario_ppdl		
Descripción: contiene el resultado final del test.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
id_modl_n_areas	Integer	Identificador de la tabla que contiene las áreas del test ppdl.
edad_basal	Integer	Edad máxima donde el paciente vence el test.
edad_techo	Integer	Edad mínima donde el paciente no vence el test.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.
cid	Integer	Identificador de modificaciones registradas en la bitácora.

Tabla 26: Descripción de la tabla modl_sumario_ppv.

Nombre: modl_sumario_ppv.		
Descripción: almacena todos los resultados de los test y la historia clínica.		
Atributo	**Tipo**	**Descripción**
id_examen_fisico_respuesta	Integer	Identificador de la tabla.
id_modl_evaluacion	Integer	Identificador de la tabla evaluación.
id_modl_n_tipo_prueba	Integer	Identificador del nomenclador tipo prueba.

Nombre: modl_sumario_ppv.		
Atributo	**Tipo**	**Descripción**
id_modl_n_percentil	Integer	Identificador del nomenclador percentil.
id_modl_clasificacion_percentil	Integer	Identificador de la tabla clasificación percentil.
id_modl_edad_vocabulario_ppv	Integer	Identificador de la tabla edad de vocabulario del test.
id_modl_desviacion_tipica_derivada	Integer	Identificador de la tabla desviación típica derivada.
id_modl_n_lenguaje	Integer	Identificador del nomenclador lenguaje.
id_modl_n_observaciones_test	Integer	Identificador del nomenclador observación test.
edad_base	Integer	Láminas que estén por debajo de la edad base serán respondidas correctamente.
edad_techo	Integer	Láminas que estén por encima de la edad techo serán respondidas incorrectamente.
puntaje_bruto	Integer	Atributo que recoge el puntaje bruto.
desarrollo_vocabulario	Varchar	Atributo que recoge el desarrollo del vocabulario.
monto_retraso	Integer	Atributo que recoge el retraso del paciente.
version	Integer	Atributo para persistir o actualizar la entidad.
eliminado	Boolean	Permite verificar si el usuario ha sido eliminado.

cid	Integer	Identificador de modificaciones registradas en la bitácora.

GLOSARIO DE TÉRMINOS

Actor del negocio: cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio, para beneficiarse de sus resultados.

Atención Temprana: permite conjuntamente con la familia y la comunidad, ofrecer a los niños con déficit en su Neurodesarrollo, un conjunto de acciones optimizadoras y compensadoras que faciliten su adecuada maduración en todos los ámbitos y que les permita alcanzar el máximo nivel de desarrollo personal y de integración social.

Confidencialidad: propiedad de la información mediante la cual, se garantizará el acceso a la misma solo por parte de las personas que estén autorizadas.

Discapacidad: dentro de la experiencia de Salud, es toda restricción o ausencia, (debida a una deficiencia), de la capacidad de realizar una actividad en la forma o dentro del nivel normal, para su edad, sexo y condiciones socioculturales. Representa la objetivación de una deficiencia y refleja alteraciones en el ámbito de la persona.

Encriptación de los datos: proceso mediante el cual cierta información o texto sin formato, es cifrado de forma que el resultado sea ilegible, a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada en el momento de almacenar o transmitir información sensible para que ésta no pueda ser obtenida con facilidad por terceros.

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.

Implementación: proceso llevado a cabo por los informáticos para hacer funcional el diseño de un software, un pseudocódigo o un conjunto de reglas previamente definidas.

Integridad: validez y consistencia de los elementos de información almacenados y procesados en un sistema informático. Son las medidas de salvaguarda que se incluyen en un sistema de información, para evitar la pérdida accidental de los datos.

Log: registro oficial de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática, un log es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué, un evento ocurre para un dispositivo en particular o aplicación.

Neurodesarrollo: adquisición de funciones, dependientes del sistema nervioso, que implican un incremento de estructuras orgánicas y funcionales a través de un proceso de maduración.

Notación Camel: Camel-Casing es común en Java. Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador debe estar en minúscula.

Notación Pascal: en este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras unidas, iniciando cada palabra con letra mayúscula.

ORM (Object-Relational-Mapping):el mapeo de objetos-relacional es una técnica de programación utilizada para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación, orientado a objetos y utilizado en una base de datos, creando una base de datos orientada a objetos virtual, por encima de la base de datos relacional.

Proceso de Desarrollo de Software: conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto. A modo de plantilla, explica los pasos necesarios para terminar el proyecto.

Proceso: 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.

Requerimientos Funcionales: son capacidades o condiciones que el sistema debe cumplir, indican que es lo que el software debe hacer, especifican como debe comportarse el sistema en situaciones particulares y como debe ser el comportamiento de entrada y salida del sistema.

Trazas: la traza de un algoritmo (o programa) indica la secuencia de acciones (instrucciones) de su ejecución, así como, el valor de las variables del algoritmo (o programa) después de cada acción (instrucción).

UCIN: Unidades de Cuidados Intensivos Neonatales.

UCIP: Unidades de Cuidados Intensivos Pediátricos. 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.

Vista Lógica: esta vista recoge las estructuras de información del software desarrollado. Muestra como la funcionalidad es diseñada dentro del sistema, define la estructura y el comportamiento del mismo.