# Universidad de las Ciencias Informáticas

**Facultad 7**


## Implementación del Test IPP y la configuración del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños

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

**Autora:** Arianna Martínez Luis

**Tutores:** MSc. Martha Denia Hernández Ramírez

Ing. Juan Miguel Peña Cabrera

*La Habana, junio 2011*

"*Año 53 de la Revolución*"

## DATOS DE CONTACTO.

***MSc. Martha Denia Hernández Ramírez.*** Graduada de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2008. Posee la categoría docente de Instructor. Durante su trabajo como profesora ha impartido las asignaturas Gestión de Software y Práctica Profesional 1. Actualmente imparte la asignatura Práctica Profesional 1.
En la vinculación con la producción, pertenece al Departamento de Sistemas Especializados en Medicina (SEM), del Centro de Informática Médica (CESIM) y específicamente, trabaja en el desarrollo del proyecto alasClínicas, donde se desempeña como analista principal.

***Ing. Juan Miguel Peña Cabrera.*** Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI), en el 2009. Instructor recién graduado en adiestramiento. Durante su trabajo como profesor ha impartido la asignatura Sistemas Operativos. Actualmente, se desempeña como tutor de estudiantes en la asignatura de Práctica Profesional.
En la vinculación con la producción, pertenece al Departamento de Sistemas Especializados en Medicina (SEM), del Centro de Informática Médica (CESIM) y específicamente, trabaja en el desarrollo del Proyecto alasTeleconsulta, donde se desempeña como desarrollador.

**RESUMEN.**

En la Universidad de las Ciencias Informáticas (UCI), se está informatizando un programa de atención y evaluación del neurodesarrollo. Este programa se denomina Renacer Contigo y en el mismo están presentes una serie de especialistas, que son los encargados de realizar las evaluaciones a los pacientes. Actualmente el trabajo realizado por este equipo se ve obstaculizado debido a que: se realiza de forma manual, aumentando la posibilidad de pérdidas de los archivos; estas pruebas son muy extensas y requiere la utilización de grandes tablas matemáticas, lo que impide que se puedan atender gran cantidad de pacientes en un mismo dia. Es por ello que se decidió realizar el Sistema de Evaluación del Neurodesarrollo en Niños (SENDN), el cual está dividido en módulos para cada especialidad que se evalúa en este programa.

En el Módulo Logopedia perteneciente al SENDN, se encuentran en fase de diseño las funcionalidades: realizar Test Inventario de las Primeras Palabras (IPP) y la configuración de este módulo; actualmente se hace necesario obtener un producto funcional que garantice el correcto funcionamiento de las mismas. El desarrollo de la solución propuesta está guiado por el Proceso Unificado de Desarrollo, PostgreSQL como gestor de base de datos y el lenguaje de programación Java. Se utiliza el Lenguaje Unificado de Modelado y Visual Paradigm para la creación de los artefactos. Como patrón de arquitectura de software, se emplea el Modelo-Vista-Controlador. La puesta en marcha de esta aplicación permitirá una mayor eficiencia en la realización del Test IPP, agilizando la evaluación de los niños en la consulta de Logopedia.

**Palabras Clave:** Neurodesarrollo, Logopedia, Test IPP.

**TABLA DE CONTENIDOS.**

## INTRODUCCIÓN.

En la actualidad, el conocimiento constituye la principal fuente de poder para cualquier organización; siendo la informática quien está estrechamente relacionada con su adquisición. Aparejado a esto, se ha evidenciado un vertiginoso avance en las Tecnologías de la Información y las Comunicaciones (TIC); lo que ha impulsado el desarrollo de aplicaciones informáticas que contribuyen con el funcionamiento de varios sectores de la sociedad.

Varios países han hecho uso de las TIC, por lo que se han visto beneficiados por este proceso de informatización. Como ejemplo de esto: "en Venezuela se desarrolló en el 2008 la Solución Informática para el Centro Nacional de Balance Alimentario; aplicación Web para la captura y envío de los datos requeridos por el Centro Nacional de Balance Alimentario."(1) Por otra parte, "en España se creó SALUS, un software para la gestión integral de hospitales, clínicas y centros médicos."(2) "En Uruguay, varias universidades hacen uso de las TIC para el estudio de los suelos, fomentando la educación e interés de los estudiantes."(3)

Cuba no se encuentra ajena al proceso de informatización de la sociedad y para el cumplimiento de este objetivo, se creó en el año 2002 la Universidad de las Ciencias Informáticas (UCI). El objetivo de esta universidad quedó explícito en las palabras del Comandante en Jefe Fidel Castro cuando expresó:

> *"(…)Tendremos una fábrica de software, podemos producir juegos, pero juegos educativos, ir estudiando cuál puede ser nuestro mercado, pensar qué servicios puede prestar esta universidad al resto de las universidades, qué servicios puede brindar a INFOMED, qué servicios puede brindar a la educación a distancia en materia de programación(…)"* (4)

Actualmente, en este importante Centro de Enseñanza Superior (CES), se desarrollan software para lograr la informatización del país; priorizando la salud como uno de los sectores más importantes. En este sentido, colaboran un grupo de instituciones cubanas y el propio Ministerio de Salud Pública (Minsap), en el desarrollo de sistemas informáticos para el sector. Se logra así, la informatización de los procesos

médicos asistenciales; garantizando de esta manera, el mejoramiento de la calidad y seguridad de la atención médica a la población.

Específicamente la Facultad 7 de la UCI, es la encargada de desarrollar dichas aplicaciones informáticas; por lo que se han creado varios proyectos productivos. Uno de ellos, está destinado a la informatización del programa Renacer Contigo; el cual consiste en un estudio que se realiza a niños de cero a cinco años de edad, egresados del Hospital Pediátrico William Soler de La Habana. Este programa es el encargado de realizar la atención y evaluación del neurodesarrollo de dichos pacientes. Cuenta con un equipo interdisciplinario en el cual están presentes una serie de especialistas en: fisiatría, nutrición, psicología, neurología, neurofisiología, logopedia y genética.

El trabajo realizado por este equipo, se ve obstaculizado por una serie de problemas que existen y que atentan, contra el buen funcionamiento de los servicios de atención y evaluación. La labor de estos especialistas se efectúa de forma manual, aumentando la posibilidad de pérdida de los archivos, o de errores con el procesamiento de la información. Para obtener los resultados, es necesario consultar grandes volúmenes de información, impidiendo que se puedan atender varios pacientes en un día. Es por ello, que en la UCI, se decide desarrollar el Sistema de Evaluación del Neurodesarrollo en Niños (SENDN), el cual estará dividido por módulos, para cada especialidad que se evalúa en el programa; contribuyendo así, con la gestión de la información que se genera en los procesos inmersos en el programa.

La presente investigación, está centrada en el Módulo Logopedia del SENDN, específicamente, en los procesos: realizar Test Inventario de las Primeras Palabras (IPP) y la configuración del módulo. Este trabajo, está precedido por un trabajo de diploma titulado: “Diseño del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños” (5); donde se obtuvo la propuesta de diseño del módulo. Actualmente, se hace necesario obtener un producto funcional a partir del diseño, que garantice la configuración del módulo y la aplicación del Test IPP.

Por lo antes planteado, se identifica como **problema a resolver**: ¿cómo hacer funcional el diseño de los procesos: realizar Test IPP y la configuración del Módulo Logopedia, del Sistema de Evaluación del Neurodesarrollo en Niños?

Este problema se enmarca en el **objeto de estudio**: los procesos de desarrollo del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños.

El **campo de acción**: los procesos de implementación y pruebas de los procesos: realizar Test IPP y la configuración del Módulo Logopedia, del Sistema de Evaluación del Neurodesarrollo en Niños.

Para la solución del problema, se plantea como **objetivo general**: implementar el Test IPP y la configuración del Módulo Logopedia, del Sistema de Evaluación del Neurodesarrollo en Niños, a partir del diseño realizado por el analista.

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

1. Análisis de los requisitos relacionados con los procesos: realizar Test IPP y la configuración del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños.

2. Revisión de la técnica de programación, plataforma, librerías, tecnologías, metodología y herramientas informáticas que se utilizarán, para el desarrollo del Test IPP y la configuración del Módulo Logopedia del Sistema de Evaluación del Neurodesarrollo en Niños; a partir de la arquitectura del Departamento de Gestión Hospitalaria de la Facultad 7.

3. Realización de una valoración crítica del diseño propuesto por los analistas, para el desarrollo del sistema.

4. Obtención de los artefactos de los flujos de trabajo: “Implementación” y “Pruebas”.

5. Implementación de las funcionalidades del Test IPP y la configuración del Módulo Logopedia, definido en el diseño del Sistema de Evaluación del Neurodesarrollo en Niños.

6. Realización de las pruebas a las funcionalidades implementadas, para garantizar el correcto funcionamiento.

Con esta investigación se obtendrá la implementación de los procesos relacionadas con el Test IPP y la configuración del Módulo Logopedia, del Sistema de Evaluación del Neurodesarrollo en Niños. Dentro de la solución se encontrarán todas las funcionalidades relacionadas con:

- ✓ Gestionar Áreas.
- ✓ Gestionar Clasificación.
- ✓ Gestionar la Clasificación de las Palabras.
- ✓ Gestionar Edad.
- ✓ Gestionar Ítem.
- ✓ Gestionar Lenguaje.
- ✓ Gestionar el Lenguaje Valor.
- ✓ Gestionar las Listas de Palabras.
- ✓ Gestionar las Observaciones del Test.
- ✓ Gestionar las Observaciones del Test Valor.
- ✓ Gestionar Palabras.
- ✓ Gestionar Percentil.
- ✓ Gestionar Tipo de Palabras.
- ✓ Crear Test IPP.
- ✓ Ver detalles del Test IPP.

El documento se estructura de la siguiente manera:

**Capítulo 1. Fundamentación teórica sobre el desarrollo del Módulo Logopedia del SENDN.**

En este capítulo se abordan los principales términos y definiciones utilizados para comprender el negocio, asociado al Módulo Logopedia del SENDN. Además, se describe la metodología de desarrollo: Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés), especificando, de los flujos de trabajo

implementación y prueba: los artefactos que se generan y los roles que se desempeñan. Por otra parte, se hace una revisión de la técnica de programación, plataforma, librerías, tecnologías y herramientas informáticas que se utilizarán, para el desarrollo de las funcionalidades.

**Capítulo 2. Descripción de la arquitectura de software a utilizar para desarrollar el Test IPP y la configuración del Módulo Logopedia.**

En el mismo, se plantean los requerimientos no funcionales que apoyarán la interfaz y el funcionamiento del software. Se describe la arquitectura a utilizar, definida previamente por el Departamento de Gestión Hospitalaria y se proponen las estrategias de integración para el sistema informático. Por otra parte, se especifica cómo se va a garantizar la seguridad de la aplicación, ya que es necesaria e importante la protección de la información con la que se va a trabajar. Se brindan las vistas que muestran: cómo el sistema se va a desplegar en una infraestructura, mostrando los requerimientos de hardware y software.

**Capítulo 3. Descripción y análisis de la solución propuesta para desarrollar el Test IPP y la configuración del Módulo Logopedia.**

Se realiza un análisis crítico del diseño elaborado por los analistas y se propone el refinamiento de los diagramas de clases del diseño y de interacción, que se adaptan a las nuevas funcionalidades que se desean implementar. Se modelan los diagramas de componentes correspondientes a cada uno de los casos de usos que se desarrollan; así como, se presentan implementaciones relevantes que constituyen aporte a la investigación.

**Capítulo 4. Realización de las pruebas a las funcionalidades: realizar Test IPP y la configuración del Módulo Logopedia.**

La realización de pruebas para verificar el correcto funcionamiento a un sistema informático, tiene gran importancia, ya que garantizan que el mismo, no presente fallos a la hora de desplegarlo en una determinada institución. En este capítulo se documentan las pruebas que se realizan a la aplicación, para garantizar el correcto funcionamiento de las funcionalidades. Además, se presentan ejemplos del diseño de los casos de prueba; así como el resultado de las no conformidades por iteraciones.

## CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA SOBRE EL DESARROLLO DEL MÓDULO LOGOPEDIA DEL SENDN.

En este capítulo, se abordarán los principales conceptos relacionados con el desarrollo del Módulo Logopedia del SENDN. Además, se abordan los objetivos y características del mismo, con el fin de comprender las principales tareas a realizar para dar cumplimiento a los requerimientos. Se realiza un estudio de la arquitectura que se utilizará para realizar el sistema, ya que la misma está previamente definida por el Departamento de Gestión Hospitalaria y por tanto, se hace necesario su análisis y comprensión. Por otra parte, se analiza la metodología de desarrollo a utilizar, enfocándose en las características, actividades y operaciones a realizar durante el proceso de implementación y pruebas, de las funcionalidades: realizar Test IPP y configuración del módulo.

### 1.1 Módulo Logopedia del SENDN.

"El neurodesarrollo es un campo basado en la neurociencia y la biología, que describe los mecanismos por los cuales los numerosos sistemas nerviosos se conectan entre sí. Este proceso, se presenta desde el inicio de la vida del desarrollo embriónico, hasta la muerte."(6) Se basa en la adquisición de funciones, dependientes del sistema nervioso, que implican un incremento de estructuras orgánicas y funcionales a través de un proceso de maduración.

El SENDN, surge para facilitar el trabajo de los especialistas que realizan el proceso de atención y evaluación del neurodesarrollo en el Programa Renacer Contigo. Las pruebas aplicadas por el personal especializado son extensas. La aplicación de los test, demanda la utilización de grandes y complejas tablas matemáticas para determinar los resultados, lo que aumenta la posibilidad de errores. El procesamiento de la información se realiza de forma manual, lo cual propicia la pérdida y deterioro de los archivos físicos. Esto provoca que se cometan errores en la entrada de datos y que se genere información duplicada; generando a su vez, retrasos en la obtención de información y en la generación de datos estadísticos. Por otra parte, para realizar estas pruebas se necesita de mucho tiempo, lo que impide que en un mismo día se pueda atender a gran cantidad de pacientes.

El SENDN se encuentra dividido en varios módulos: Expediente y Turnera, Fisiatría, Neurología, Nutrición, Neurofisiología, Psicología y Logopedia.

El Módulo Logopedia está basado en la informatización de los procesos principales que se desarrollan en esta área y en el cual, se definen los test que se aplican a los niños para la evaluación del neurodesarrollo; así como se configuran los nomencladores de estas pruebas.

Los exámenes que se realizan en este módulo, permiten valorar si el estado lingüístico del niño es acorde con su edad cronológica; lo cual constituye un parámetro muy importante para su valoración integral. Entre ellos se encuentran los exámenes logofoniátricos, que miden los niveles de comunicación oral como: el lenguaje, el habla y la voz; además de incluir, la realización de exámenes físicos y teniendo en cuenta la edad del niño, se prosigue a la aplicación de tres test.

Los test son:

- ✓ Inventario de las Primeras Palabras (IPP): examen que puede ser aplicado a niños cuya edad cronológica sea de 12 a 36 meses y que evalúa el desarrollo del vocabulario.
- ✓ "Prueba de Pesquisaje del Desarrollo del Lenguaje (PPDL): examen que permite el diagnóstico temprano en infantes con retardo en el desarrollo del lenguaje. Evalúa a todos los niños desde el nacimiento hasta los 36 meses de edad."(7)
- ✓ "Prueba de Vocabulario Imágenes Peabody (PPV, por sus siglas en inglés): evalúa el vocabulario receptivo de los pacientes, ya que se tiene que asociar la palabra que se diga, con la imagen que se muestra."(8)

### 1.1.2 Test IPP.

A partir del Test IPP se explora el vocabulario activo cuantitativo y cualitativo de un niño, en un rango de uno a tres años de edad. Consiste en una lista de 414 palabras fraccionadas en tres listas de 138 palabras cada una, que a su vez, están divididas en 21 categorías que permiten: realizar estudios con objetivos clínicos, educativos y de investigación.

"El test, presenta varias ventajas con relación a otras pruebas que evalúan el vocabulario, ya que permite evaluar a niños pequeños que quedan fuera de los rangos normativos de otras pruebas. Parte de un estudio de frecuencia del uso de palabras en niños de uno a tres años y ofrece una amplia visión, tanto cuantitativa, como cualitativamente. Por otra parte, permite evaluaciones sucesivas del niño sin recurrir al

mismo material de prueba, se aplica fácil y rápidamente por personal entrenado, pero no necesariamente calificado."(9)

En la Tabla 1 se muestra la lista de las palabras, las cuales están divididas en 21 categorías:

**Tabla 1. Categorías y cantidad de palabras del Test IPP.**

| Categoría | Total de palabras |
|---|---|
| Partes del cuerpo | 12 |
| Ropas y prendas de vestir | 9 |
| Comidas y bebidas | 21 |
| Muebles y habitaciones | 6 |
| Objetos del interior | 30 |
| Objetos del exterior | 15 |
| Personas | 19 |
| Lugares | 12 |
| Cualidades | 15 |
| Cuantificables | 10 |
| Preguntas | 4 |
| Respuestas | 3 |
| Tiempo | 4 |
| Vehículos | 12 |
| Juguetes | 15 |
| Partículas de relación y modificadores | 10 |
| Pronombres | 12 |
| Interjecciones | 6 |
| Verbos | 105 |
| Actividades | 15 |
| Animales | 33 |

El tiempo de aplicación del test, puede variar en función de las personas involucradas en la evaluación, pero generalmente, no excede de 30 minutos cuando se aplica la lista completa. Se obtiene un puntaje final que es comparado con la de un grupo de niños de su misma edad cronológica, para determinar su edad de vocabulario (EV). Esta edad, puede o no coincidir con su edad cronológica, por lo que un niño puede tener una EV acorde a su edad cronológica, por encima o por debajo, de su edad cronológica.

### 1.1.3 Configuración del Módulo Logopedia.

La configuración del Módulo Logopedia, es la encargada de permitir la gestión de los procesos en tiempo de ejecución. Está compuesta por varios componentes que ofrecen valor añadido al sistema. Aumentando de esta forma: la usabilidad, eficiencia, calidad y valor comercial del mismo; ya que el usuario que utilice el software, podrá decidir cuál es la información que realmente tiene importancia para su trabajo.

Estos componentes se denominan *nomencladores* y contienen información relacionada con la aplicación de los test. Permitiéndoles a los usuarios, gestionar la información que se genera en los procesos de atención y evaluación. Los nomencladores que componen la configuración del módulo son: Áreas, Edad, Lenguaje, Lenguaje Valor, Clasificación, Clasificación Palabras, Ítem, Palabras, Percentil, Observaciones Test, Observaciones Test Valor, Listas Palabras y Tipo de Prueba.

Para poder gestionar de manera más eficiente los datos de dichos nomencladores, el sistema permitirá realizar una serie de operaciones sobre los mismos. De esta forma, el administrador del sistema podrá: adicionar, eliminar, buscar, listar y ver datos de los diferentes nomencladores.

## 1.2 Herramientas y tecnologías para el desarrollo.

Para el desarrollo del Módulo Logopedia, se utilizan una serie de metodologías, tecnologías, plataformas y herramientas, que fueron previamente definidas por el Sistema de Gestión Hospitalaria (alasHIS). Esto se debe a que el SENDN, es un subsistema del alasHIS y por tanto, la arquitectura a utilizar está previamente definida por el mismo. Es por ello que, para la realización de este módulo no se realizará un proceso de selección, sino que se realizará un estudio que incluye la comprensión de dicha arquitectura. En los epígrafes subsiguientes, se mencionarán algunas de estas tecnologías utilizadas para desarrollar el

sistema y luego, en el capítulo II, se hará referencia a las tecnologías empleadas, que se encuentran divididas según las capas del patrón de arquitectura que se utiliza.

### 1.2.1 Metodología de Desarrollo de Software: Proceso Unificado de Desarrollo.

"Las metodologías son un conjunto de métodos o reglas, que por una parte, sirven de guía para realizar los trabajos que van dando forma al desarrollo del programa informático. Además, obligan a la dirección del proyecto y a los componentes de los equipos, a realizar ciertas comprobaciones sistemáticas. Se logra que el resultado final, al menos desde un punto de vista formal, no presente incoherencias y esté dirigido a un objetivo claro y prefijado."(10)

Como metodología de desarrollo para la creación del Módulo Logopedia, se utiliza el Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés). "Este es un proceso bien definido, estructurado y adaptable a las características y necesidades, de cada proyecto específico. Se definen un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software."(11) Su objetivo es producir software de alta calidad, que cumpla con los requerimientos de los usuarios dentro de la planificación y presupuesto establecidos.

El ciclo de vida de RUP se caracteriza por ser:

- ✓ Dirigido por casos de uso: el proceso de desarrollo sigue un hilo, avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso, son un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante.
- ✓ Centrado en la arquitectura: la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo. Por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- ✓ Iterativo e incremental: en donde el trabajo se divide en partes más pequeñas o mini proyectos. Cada mini proyecto, es una iteración que resulta en un incremento. Las iteraciones, hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto.

Define cuatro 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?

La Figura 1 muestra la relación que existe entre los roles, las actividades y los artefactos, de esta metodología de desarrollo de software:


**Figura 1. Relación entre roles, actividades y artefactos.**

Esta metodología puede ser vista en dos dimensiones: las fases y los flujos de trabajo, como se muestra en la Figura 2:

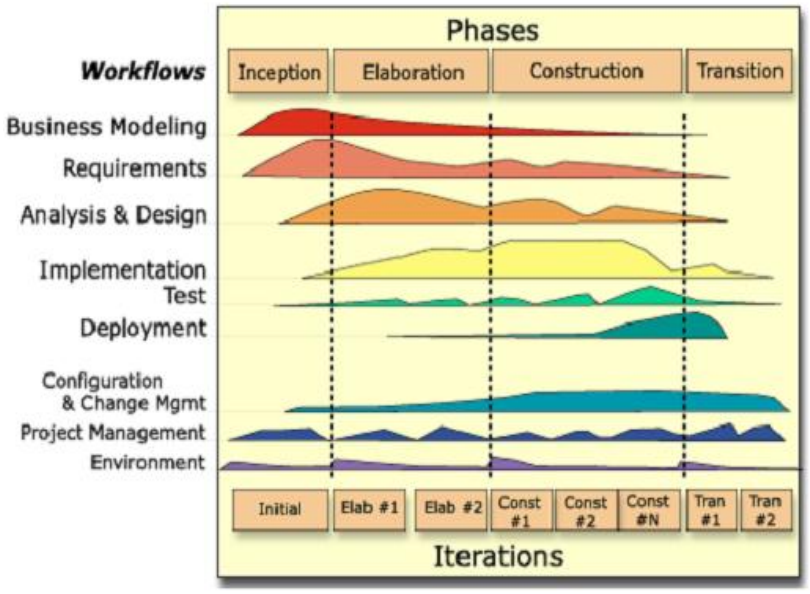


**Figura 2. Fases y flujos de trabajo de RUP.**

Las fases de RUP son:

- ✓ Inicio: tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos potenciales asociados al proyecto. Se propone una visión muy general de la arquitectura de software y se produce el plan de las fases y el de iteraciones.
- ✓ Elaboración: se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase. Se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.
- ✓ Construcción: el propósito de esta fase es completar la funcionalidad del sistema, para ello, se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.
- ✓ Transición: el objetivo de la fase de transición es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación. Se

capacita a los usuarios y se provee el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

"Los flujos de trabajo de RUP son:

- ✓ Modelo del Negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- ✓ Requerimiento: 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 y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- ✓ Prueba (Testeo): busca los defectos a los largo del ciclo de vida.
- ✓ Instalación o despliegue: produce release 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 producir un producto que satisfaga las necesidades de los clientes.
- ✓ Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización y actualización concurrente de elementos, control de versiones, entre otras.
- ✓ 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."(12)

Teniendo en cuenta la precedencia de un análisis y diseño del sistema, la presente investigación está basada en los flujos de trabajo: "Implementación" y "Pruebas", por lo que los roles que se desempeñan, son los de implementador y probador. Es por ello que, se mencionarán las principales características y los

artefactos que se generan en estos flujos de trabajo.

**Flujo de trabajo "Implementación":** se implementan las clases y objetos en ficheros fuente, binarios y ejecutables. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- ✓ Planificar qué subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.
- ✓ Cada implementador decide en qué orden implementa los elementos del subsistema.
- ✓ Si encuentra errores de diseño, los notifica.
- ✓ Se prueban los subsistemas individualmente.
- ✓ Se integra el sistema siguiendo el plan.

Se genera como artefacto el Modelo de Implementación: este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y otros tipos de ficheros, necesarios para la implantación y despliegue del sistema.

**Flujo de trabajo "Pruebas":** este flujo de trabajo, es el encargado de evaluar la calidad del producto que se está desarrollando. El desarrollo del flujo de trabajo, consistirá en: planificar qué es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados; de forma que la información obtenida, sirva para ir refinando el producto a desarrollar.

"Los objetivos de este flujo de trabajo son:

- ✓ Encontrar y documentar defectos en la calidad del software.
- ✓ Asesorar sobre la calidad del software percibida.
- ✓ Verificar las funciones del producto de software según lo diseñado.
- ✓ Verificar que los requisitos tengan su apropiada implementación."(13)

Se genera como artefacto el Modelo de Pruebas: Para cada caso de uso, se establecen pruebas de caja negra que validarán la correcta implementación del mismo. Cada prueba, es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados.

### 1.2.2 Lenguaje de modelado: Lenguaje Unificado de Modelado.

"El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), es un lenguaje de modelado visual para especificar, visualizar, construir y documentar artefactos de un sistema de software."(14)

UML, proporciona una forma estándar de representar los planos de un sistema. Comprende tanto elementos conceptuales, como los procesos de negocio y las funciones del sistema. Incluye además, elementos concretos como: las clases escritas de un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables.

RUP emplea UML como base para el desarrollo de software en cada una de sus fases y disciplinas. La especificación de UML no define un proceso estándar, pero está pensado para ser útil, en un proceso de desarrollo iterativo.

*"Las ventajas de este lenguaje de modelado son:*

- ✓ 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.
- ✓ Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- ✓ 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."(15)

### 1.2.3 Herramienta para la elaboración de diagramas: Visual Paradigm.

La herramienta de ingeniería de software asistida por computadora (CASE, por sus siglas en inglés) denominada como Visual Paradigm, utiliza UML como lenguaje de modelado. Es una herramienta que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. “El software de modelado UML, ayuda a una más rápida construcción de aplicaciones de calidad, más óptimas y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, obtener código inverso, generar código desde diagramas y generar documentación.”(16)

Esta importante herramienta posee varios beneficios, dentro de los más importantes se pueden mencionar:

- ✓ Persistencia de forma fácil.

Los desarrolladores emplean mucho esfuerzo en salvar y cargar objetos entre la memoria y la base de datos, lo que hace que el programa sea complicado y difícil de mantener. Simplifican estas tareas mediante la generación de una capa de persistencia entre objeto y modelos de datos.

- ✓ Generador de mapeo objeto-relacional sofisticado.

La capa de mapeo objeto-relacional que se genera, incorpora características como soporte de transacciones, capaz de conectar en caché, agrupación de conexiones y personalización de sentencias SQL.

- ✓ Amplia cobertura para bases de datos.

Soporta una amplia gama de base de datos, incluidos Oracle, DB2, Cloudscape / Derby, Sybase Adaptive Server Enterprise, Sybase SQL Anywhere, Microsoft SQL Server, PostgreSQL, MySQL y otros.

- ✓ Base de datos de ingeniería inversa.

Permite la ingeniería inversa existente en una base de datos a través de la Conexión Java a la Base de Datos (JDBC, por sus siglas en inglés), en el modelo entidad-relación. Los desarrolladores pueden transformar el modelo entidad-relación al modelo de objetos y rediseñar la base de datos, para un mayor desarrollo.

- ✓ Integración con IDE.

Visual Paradigm no sólo es una aplicación independiente, se puede integrar a los principales IDEs: Eclipse/WebSphere ®, Borland.

### 1.2.4 Lenguaje de programación: Java.

"Un lenguaje de programación, es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Permite a uno o más programadores especificar de manera precisa:

- ✓ Sobre qué datos una computadora debe operar.
- ✓ Cómo deben ser estos almacenados y transmitidos.
- ✓ Qué acciones debe tomar bajo una variada gama de circunstancias."(17)

Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. "En la actualidad, es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general."(18)

"Es un lenguaje de programación orientado a objetos y por lo tanto, soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo."(19)

Java toma mucha de su sintaxis de C y C++; pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores. Los programas escritos en este lenguaje pueden ejecutarse en cualquier tipo de hardware.

### 1.2.5 Servidor de aplicaciones: JBoss AS.

Un servidor de aplicaciones, es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el Protocolo de Transferencia de Hipertexto (HTTP, por sus siglas en inglés). 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 AS es el servidor de aplicaciones de código abierto más utilizado actualmente en todo el mundo. Se encuentra certificado por J2EE y soporta sistemas de gran complejidad y alta concurrencia. Está basado en Java, por lo que puede ser utilizado en cualquier sistema operativo que lo soporte. Implementa todo el paquete de servicios de J2EE." (20)

"Las características destacadas de JBoss AS incluyen:

- ✓ Producto de licencia de código abierto sin coste adicional.
- ✓ Confiable a nivel de empresa.
- ✓ Orientado a arquitectura de servicios.
- ✓ Flexibilidad consistente.
- ✓ Servicios del middleware para cualquier objeto de Java."(21)

Es el primer servidor de aplicaciones de código abierto, preparado para la producción disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de negocio electrónico. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, puede ser descargado, utilizado, instalado, y distribuido, sin restricciones por la licencia. Por este motivo, es la plataforma más popular de middleware para desarrolladores, vendedores de software independientes, así como, para grandes empresas.

### 1.2.6 Sistema Gestor de Bases de Datos: PostgreSQL.

Un Sistema Gestor de Base de Datos (SGBD), es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

"PostgreSQL es una derivación libre del proyecto PORTGRES y utiliza el lenguaje SQL92/SQL99. Es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional."(22)

Es multiplataforma y funciona en los principales sistemas operativos, como Linux, UNIX y Windows. También apoya el almacenamiento de grandes objetos binarios, imágenes, sonidos o vídeo.

Características de PostgreSQL:

- ✓ Múltiples lenguajes de procedimientos, ya que los disparadores y otros procedimientos pueden ser escritos en varios lenguajes de procedimientos. El código del lado del servidor es comúnmente escrito en PL / PostgreSQL, un lenguaje de procedimiento similar al de Oracle PL / SQL.
- ✓ Múltiples-cliente de Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés), debido a que soporta el desarrollo de aplicaciones cliente en varios lenguajes, interfaz para PostgreSQL desde C, C + +, ODBC, Perl, PHP, Tcl / Tk, y Python.
- ✓ Diferentes tipos de datos como: integer, string, numeric, boolean, char, varchar, date, interval, y timestamp, tipos geométrica, tipo de datos booleanos y tipos de datos diseñados específicamente para hacer frente a las direcciones de red.
- ✓ La extensibilidad es una de las características más importantes de PostgreSQL ya que puede ser ampliado, se pueden añadir nuevos tipos de datos, nuevas funciones y operadores, e incluso nuevos lenguajes de procedimiento y de cliente.
- ✓ Posee integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

### 1.2.7 Entorno de Desarrollo Integrado: Eclipse.

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios.

“Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs, pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.”(sic)(23)

Eclipse, es una plataforma de software de código abierto independiente de una plataforma para desarrollar lo que se nombra "Aplicaciones de Cliente Enriquecido". Esta plataforma, ha sido usada para desarrollar un entorno integrado de desarrollo. Este IDE es más general para el desarrollo de aplicaciones en Java, en la que se ha especializado. Es una herramienta que necesita de mucha ayuda del hardware para realizar la compilación del código fuente escrito, además de ser multiplataforma.

"En cuanto a la utilización de eclipse para la creación de aplicaciones clientes se puede decir que:

- ✓ Provee al programador una serie de frameworks, para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, entre otras.
- ✓ El Kit de Desarrollo de Software de Eclipse (SDK, por sus siglas en inglés), incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java.
- ✓ El IDE también hace uso de un espacio de trabajo, en este caso, un grupo de metadata en un espacio para archivos planos; permitiendo modificaciones externas a los archivos, en tanto se refresque el espacio de trabajo correspondiente."(24)

Como resultado de la revisión de las herramientas, técnicas y tecnologías, que son utilizadas para el desarrollo del módulo, se concluyó que:

- ✓ Se utilizará como metodología de desarrollo de software RUP, que se caracteriza por ser dirigido por casos de uso, iterativo e incremental y centrado en la arquitectura.
- ✓ Para la creación de diagramas y esquemas que guíen el desarrollo del sistema, se empleará el lenguaje de modelado UML.
- ✓ La herramienta Visual Paradigm es la que se utilizará para la creación de los diagramas del sistema, con su ventaja de poder modelar todo el ciclo de vida del software.
- ✓ Para desarrollar las funcionalidades requeridas, se usará el lenguaje de programación orientado a objetos Java.

- ✓ Se dispondrá como servidor para brindar las aplicaciones a las máquinas clientes y servir de intermediario entre los procesos, de JBoss AS.
- ✓ Como herramienta de código abierto para permitir la gestión de la información almacenada en la base de datos se utilizará PostgreSQL.
- ✓ Se empleará Eclipse como la herramienta que brinda el ambiente necesario, para poder realizar el desarrollo de las funcionalidades.

En este capítulo se abordaron los procesos llevados a cabo en el Módulo Logopedia del SENDN. Por otra parte, se analizaron las funcionalidades que se van a implementar dentro del módulo y los aspectos relacionados con la configuración del mismo, ya que se hace necesario tener un dominio del funcionamiento de la aplicación, para poder desarrollar de manera eficiente, los procesos que necesitan automatización.

Este análisis se ha realizado teniendo en cuenta las pautas para la implementación definidas por el Departamento de Gestión Hospitalaria, ya que el SENDN, al cual pertenece el Módulo Logopedia, constituye un subsistema del alasHIS y por tanto, su arquitectura está basada en la del alasHIS.

## CAPÍTULO 2. DESCRIPCIÓN DE LA ARQUITECTURA DE SOFTWARE A UTILIZAR PARA DESARROLLAR EL TEST IPP Y LA CONFIGURACIÓN DEL MÓDULO LOGOPEDIA.

A medida que aumenta la complejidad de los sistemas de software, surgen nuevos aspectos del desarrollo de aplicaciones que hasta ese momento, no se habían tenido en cuenta. Dentro de esta tendencia, se encuentra el creciente interés por los aspectos arquitectónicos del software del que se está siendo testigo en los últimos tiempos. Un aspecto crítico a la hora de desarrollar sistemas de software complejos es el diseño de su arquitectura, representada como un conjunto de elementos computacionales y de datos interrelacionados de un modo determinado.

"La importancia de representar de forma explícita la arquitectura de los sistemas de software es evidente:

- ✓ Estas representaciones elevan el nivel de abstracción, facilitando la comprensión de los sistemas de software complejos.
- ✓ Hacen que aumenten las posibilidades de reutilizar tanto la arquitectura como los componentes que aparecen en ella."(25)

Debido a la importancia que tiene para realizar un software de calidad, el uso de la adecuada arquitectura y la correcta descripción de la misma; en este capítulo se detallarán los elementos que conformarán la arquitectura del sistema. Para ello, se describe el patrón de arquitectura se software Modelo Vista Controlador (MVC), en el cual, residen una serie de tecnologías que contribuyen con el uso eficiente de este patrón. Se describen los Requerimientos No Funcionales (RNF) del software, que son necesarios para desarrollar un sistema: eficiente, usable y fiable.

Por otra parte, se detallan las estrategias de integración que Módulo Logopedia utiliza, para consultar los datos necesarios de otras tablas que conforman los esquemas del alasHIS. Se plantean además, varias especificaciones para garantizar las seguridad de la aplicación informática, debido a que la información con la que se va a trabajar contiene datos personales de los pacientes y de las consultas médicas realizadas. Se mostrará el diagrama de despliegue que se propone utilizar para desplegar el sistema en la institución hospitalaria y por último, se exponen las estrategias de codificación y los estándares a utilizar, para desarrollar las funcionalidades: realizar Test IPP y la configuración del Módulo Logopedia perteneciente al SENDN.

### 2.1 Descripción de la arquitectura de software.

“La arquitectura de software, se define como la estructura de los componentes de un programa o sistema, sus interrelaciones y los principios y reglas que gobiernan su diseño y evolución en el tiempo. 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.”(26)

### 2.2 Patrón de arquitectura.

Un patrón, es un modelo que se puede seguir para realizar algo. Los patrones surgen de la experiencia de los seres humanos de tratar de lograr ciertos objetivos. Capturan la experiencia existente y probada para promover buenas prácticas. Los patrones de arquitectura de software describen un problema particular y recurrente del diseño, que surge en un contexto específico y presenta un esquema genérico y probado de su solución. El patrón de arquitectura que se utiliza para el desarrollo de las funcionalidades: realizar Test IPP y la configuración del Módulo Logopedia es el Modelo-Vista-Controlador.

#### 2.2.1 Modelo-Vista-Controlador.

El MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, como se puede apreciar en la Figura 3.

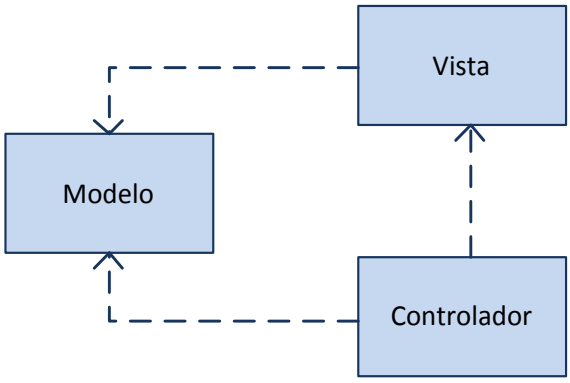


**Figura 3. Patrón de arquitectura Modelo Vista Controlador.**

Los componentes del MVC se nombran:

- ✓ Presentación: presenta el sistema al usuario, es el encargado de comunicar y capturar la información del mismo hacia las otras capas.
- ✓ Lógica de negocio: en esta capa residen los programas que se ejecutan, recibe las peticiones del usuario y envía las respuestas. Es aquí donde se establecen las reglas del negocio.
- ✓ Acceso a datos: contiene las entidades, los objetos de acceso a datos y las clases que interactúan con la base de datos.

El uso de este patrón es favorable; establece una separación entre los componentes de un programa permitiendo implementarlos por separado; esto posibilita que si uno de estos componentes tiene un mal funcionamiento pueda reemplazarse sin que se vean afectados los demás.

### 2.3 Tecnologías presentes en las capas del MVC.

En este epígrafe se tratarán una serie de conceptos relacionados con las tecnologías a utilizar en el proceso de desarrollo de las funcionalidades para el SENDN. Estas tecnologías, aparecerán según su ubicación en las capas del MVC: Presentación, Lógica de negocio y Acceso a datos.

#### 2.3.1 Tecnologías en la capa Presentación.

**Java Server Faces (JSF): "**es un framework Java que permite crear interfaces de usuario (UI) para aplicaciones web, mediante componentes reutilizables. Permite el manejo de estados y eventos, así como la asociación entre los datos de la interfaz y los datos de la aplicación web. Permite desarrollar rápidamente aplicaciones de negocio dinámicas, en las que toda la lógica de negocio se implementa en Java, o es llamada desde Java, creando páginas para las vistas muy sencillas. Además, resuelve validaciones, conversiones, mensajes de error e internacionalización."(27)

**Richfaces:** es una biblioteca de componentes para JSF y un avanzado framework para la integración de AJAX (técnica de desarrollo web para crear aplicaciones interactivas); con facilidad en la capacidad de desarrollo de aplicaciones de negocio. Sus componentes vienen listos para su uso, por lo que los desarrolladores, pueden ahorrar tiempo de inmediato para aprovechar las características de los

componentes para crear aplicaciones Web. Richfaces también aprovecha al máximo los beneficios del framework JSF incluyendo: la gestión de estática y dinámica los recursos.

“Son características de Richfaces las siguientes:

- ✓ Se integra perfectamente en el ciclo de vida de JSF.
- ✓ Incluye funcionalidades Ajax.
- ✓ Contiene un set de componentes visuales, los más comunes para el desarrollo de una Aplicación Web Rica.”(28)

**Ajax4JSF:** es una librería de código abierto que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas, dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript (lenguaje de programación usado en páginas web).

“La librería Ajax4Jsf presenta varios beneficios como:

- ✓ Permite intensificar el conjunto de beneficios de JSF mientras se trabaja con AJAX.
- ✓ Dispone de un paquete de recursos con las clases de la aplicación Java. Además de su núcleo, la funcionalidad de Ajax que se incluye en Ajax4jsf, proporciona un avanzado apoyo a la gestión de diferentes recursos: imágenes, código JavaScript y Hojas de Estilo en Cascada (CSS, por sus siglas en inglés).
- ✓ La capacidad de habilitar Ajax a componentes JSF sin cambios en el propio componente.
- ✓ Dispone de una arquitectura abierta y soporte para los estándares en la industria; permitiendo mezclar componentes que son de distintas librerías.
- ✓ Trabaja en el lado del servidor.”(29)

**Facelets:** es un framework simplificado de presentación, en donde es posible diseñar de forma libre una página web y luego asociarle los componentes JSF específicos. Aporta mayor libertad al diseñador y mejora los informes de errores que tiene JSF entre otras cosas.

Entre las principales características de Facelets se encuentran:

- ✓ Facilidad en la creación de plantillas para los componentes y páginas.
- ✓ Un buen sistema de reporte de errores.
- ✓ No es necesaria la configuración del Lenguaje de Marcado Extensible (XML, por sus siglas en inglés).

“Las principales ventajas de Facelets son:

- ✓ Construcción de interfaces basadas en plantillas.
- ✓ Rápida creación de componentes por composición.
- ✓ Fácil creación de funciones y librerías de componentes.”(30)

**Lenguaje Extensible de Marcado de Hipertexto** (XHTML, por sus siglas en inglés)**:** es un lenguaje de marcado pensado para sustituir al Lenguaje de Marcas de Hipertexto (HTML, por sus siglas en inglés), como estándar para las páginas web. Su objetivo es lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas.

“Al estar orientado al uso de un etiquetado correcto, exige una serie de requisitos básicos a cumplir en lo que a código se refiere. Entre estos requisitos básicos se puede mencionar la estructuración coherente dentro del documento, donde se incluirían: elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados, entre otros.”(31)

### 2.3.2 Tecnologías en la capa Lógica de negocio.

**JBoss Seam:** es un framework que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar 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) con anotaciones. “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 EJB3 y JSF.”(32)

### 2.3.3 Tecnologías en la capa Acceso a datos.

**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 que permiten establecer estas relaciones.

Es una herramienta ORM completa, que ha conseguido una excelente reputación en la comunidad de desarrollo de software libre, principalmente a la de desarrolladores en Java; posicionándose claramente como el producto de código abierto, líder en este campo gracias a sus prestaciones, buena documentación y estabilidad. “Es valorado por muchos, incluso, como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte, la reciente integración dentro del grupo JBOSS que seguramente, generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones.”(33)

Como todas las herramientas de su tipo, busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto, permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos, a partir de la información disponible.

**Enterprise JavaBeans (EJB3):** es un componente que trabaja del lado del servidor y que es utilizado para encapsular la lógica de negocio de las aplicaciones informáticas. Se conoce como la plataforma para construir aplicaciones portables y eficientes, empleando Java, como lenguaje de programación.

Entre las principales características se encuentran:

- ✓ Contienen lógica de negocio, que opera sobre los datos de la empresa.
- ✓ El acceso del cliente es mediado por el contenedor en el cual el enterprise bean es desplegado. Siendo este acceso transparente para el cliente.
- ✓ El estándar EJB 3 es desarrollado por Java Community Process (JCP).

"Los beneficios de emplearlo son:

- ✓ Simplifica el desarrollo, el contenedor EJB es responsable de la administración de transacciones y autorizaciones de seguridad.
- ✓ La lógica del negocio reside en los enterprise beans y no en el lado del cliente, permitiendo que el desarrollo del lado del cliente, esté desacoplado de la lógica del negocio.
- ✓ Puede residir en diferentes servidores y puede ser invocado por un cliente remoto."(34)

**Java Persistence API (JPA):** es la API para la persistencia de objetos Java a cualquier base de datos relacional. Esta API fue desarrollada para la plataforma Java EE e incluida en el estándar de EJB 3.0. Busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí ocurría con EJB2, y permitir usar objetos regulares (conocidos como POJOs).

"Java Persistence API consta de tres áreas:

- ✓ El Java Persistence API
- ✓ El lenguaje de consulta.
- ✓ El mapeo de los metadatos objeto/relacional."(35)

### 2.3.4 Tecnologías horizontales.

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

**Java Platform Enterprise Edition (JEE):** es un estándar del sector para desarrollar aplicaciones Java portátiles, robustas, escalables y seguras para el servidor. Java EE proporciona API de servicios Web, modelos de componente, gestión y comunicación que lo convierten en el estándar del sector para implementar aplicaciones de Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) y Web 2.0 de clase empresarial.

"Java EE 5 es la última versión de Java EE. Acelera y simplifica el desarrollo Java empresarial, especialmente para servicios web, aplicaciones web y componentes de transacción. Así mismo, introduce el nuevo modelo de persistencia de bases de datos."(36)

**Java Runtime Environment (JRE):** se corresponde con un conjunto de utilidades que permite la ejecución de programas java sobre todas las plataformas soportadas.

La Máquina Virtual Java (JVM, por sus siglas en inglés), es una instancia de JRE en tiempo de ejecución, este es el programa que interpreta el código Java y las librerías de clases estándar que implementan las API de Java. Ambas JVM y API, deben ser consistentes entre sí, de ahí que sean distribuidas de modo conjunto. El usuario sólo necesita el JRE, para ejecutar las aplicaciones desarrolladas en lenguaje Java.

## 2.4 Requerimientos No Funcionales.

Los requerimientos no funcionales de un software, son propiedades o cualidades que el producto debe poseer. Estos muestran las características que hacen al producto atractivo, usable, rápido, o confiable. A diferencia de los requerimientos funcionales, estos no modifican o alteran la funcionalidad del producto. Los requerimientos deben ser especificados por escrito y posibles de verificar o probar, además, sus especificaciones deben ser lo más abstracto y conciso posible y descritos como una característica del sistema. Los requerimientos no funcionales se dividen en varias categorías, a continuación se listan algunas de los más usados en el sistema a desarrollar:

### 2.4.1 RNF Usabilidad.

- ✓ El sistema estará diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo, en un tiempo reducido.
- ✓ Los usuarios normales tendrán un plazo de 20 días para tener un dominio básico del sistema, además de conocer el funcionamiento eficiente de la aplicación. Los usuarios avanzados, contarán con un plazo de 30 días para tener un dominio superior del sistema en cuestión.

### 2.4.2 RNF Fiabilidad.

- ✓ En los servidores de los hospitales y en el Centro de Datos Nacional del Minsap, se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación, como de base de datos. Se garantizarán además, políticas de respaldo a toda la información, evitando pérdidas en caso de desastres ajenos al sistema.
- ✓ Las informaciones médicas relacionadas con los pacientes y que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas, para evitar accesos o modificaciones, no autorizadas.
- ✓ Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario, o por el administrador del sistema.
- ✓ Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.
- ✓ Se establecerán mecanismos de control y verificación para los procesos susceptibles a que se cometa fraude. Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.
- ✓ El sistema implementará un 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.

- ✓ Se implementará un control de cambios a determinados campos de información, de forma tal que sea posible determinar cuáles han sido las actualizaciones que se le han realizado.
- ✓ Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos, independientemente de que para este, el elemento ya no exista.
- ✓ Se permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

### 2.4.3 RNF Eficiencia.

- ✓ El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos.
- ✓ 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.
- ✓ Se respetarán las 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), para destruir referencias que ya no estén siendo usadas, optimizar el trabajo con cadenas, entre otras buenas prácticas, que ayudan a mejorar el rendimiento.

### 2.4.4 RNF Soporte.

- ✓ Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP *(Requisito de Seguridad de acceso y administración de usuarios).*
- ✓ Se permitirá administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación *(Monitoreo de funcionamiento).*

- ✓ Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo, además de recuperar la base de datos, a partir de los respaldos realizados *(Respaldo y recuperación de base de datos).*
- ✓ Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema, para esto debe existir un registro de trazas que almacene todas las transacciones realizadas en el sistema, indicando para cada caso como mínimo: usuario que realizó la transacción, tipo de operación que se realizó, fecha y hora en que se realizó la operación e información contenida en el registro modificado *(Auditoría).*
- ✓ Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores *(Configuración de parámetros).*

**2.4.5 RNF Réplica.**

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

**2.4.6. RNF Restricciones de diseño.**

*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 7 o superior o Firefox 3.x).
  - ✓ *Servidor de Aplicaciones:* servidor con sistema operativo Linux, que soporte el JRE 1.5 o superior, al JBoss AS 4.2 y tenga una JVM 6.4 o superior. Estas mismas condiciones se aplican para los servidores de aplicación del Centro de Datos.
  - ✓ *Servidor de Base de Datos:* servidor sistema operativo Linux, que soporte a PostgreSQL Server 8.3 o superior en los servidores de base de datos de cada hospital, y 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.

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

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

### 2.4.8 RNF Interfaz.

El sistema constará de las siguientes interfaces:

- **Interfaz de usuario.**
  - ✓ La interfaz contará con teclas de función y menús desplegables, que faciliten y aceleren su utilización.
  - ✓ Todos los textos y mensajes en pantalla aparecerán en idioma español.
  - ✓ Las ventanas del sistema contendrán claro y 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.

- ✓ 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.
- ✓ Se diseñarán salidas de información en forma de gráficos, estadísticas, pronósticos y análisis comparativos que puedan ser usados como soporte para la toma de decisiones.
- ✓ El sistema incluirá reportes estándares y parametrizables, que permitirán al usuario configurar la información de salida y el orden en que aparecen los datos.

- **Interfaz de comunicación.**
  - ✓ Para el intercambio electrónico de datos entre aplicaciones, se usará el estándar Health Level Seven (HL7).
  - ✓ El software usará el formato estándar Web Services Description Language (WSDL, por sus siglas en inglés), para la descripción de los servicios web.
  - ✓ Implementará además, mecanismos de encriptación de datos para el intercambio de información con sistemas externos.
  - ✓ El sistema utilizará mecanismos de compactación de los datos que se intercambiarán con sistemas externos con el objetivo de minimizar el tráfico en la red y economizar el ancho de banda.

### 2.4.9 RNF Seguridad.

Este es uno de los más difíciles, ya que provocará los mayores riesgos si no se maneja correctamente. La seguridad de un sistema no solo tiene en cuenta la seguridad del sistema propiamente dicho sino, además, el ambiente en el que se usará el mismo; por lo que se tiene que contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso a este y las regulaciones legales que afecta o determina el uso del sistema y que serán tenidas en cuenta si se incumple.

- ✓ La seguridad puede ser tratada en tres aspectos diferentes: confidencialidad, integridad y disponibilidad, por lo que se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos, solo a los niveles establecidos de acuerdo con la función que realizan.
- ✓ Las contraseñas podrán cambiarse sólo por el propio usuario, o por el administrador del sistema.
- ✓ Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.
- ✓ El sistema proporcionará un registro de actividades (log) de cada usuario.
- ✓ Ninguna información que se haya ingresado en el mismo, será eliminada físicamente de la BD.
- ✓ Se permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

### 2.4.10 RNF Rendimiento.

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

## 2.5 Estrategias de integración.

El SENDN al cual pertenece el Módulo Logopedia, se integra al alasHIS, ya que es un subsistema del mismo. Existe una relación entre todos los componentes del sistema alasHIS, el cual cuenta con otros subsistemas que brindan información relevante, que permite el funcionamiento de los demás.

Para la aplicación del Test IPP y la configuración del Módulo Logopedia, se requiere la obtención de los listados de pacientes a los cuales se va a atender y los datos personales de los mismos. Esta información la contiene el Módulo Expediente y Turnera, el acceso a la misma se realiza a través de la conexión a la tabla mode_expediente. Dicha conexión, viene dada con el esquema HC_Local del alasHIS, que es la responsable de gestionar entre otros aspectos, los datos personales del paciente en la tabla hoja_frontal con su identificador, quien es el encargado de efectuar la relación.

**2.6 Especificación de los términos de seguridad.**

- ✓ Para fomentar la seguridad en el sistema, se propone un control de acceso a nivel de usuarios y protegidos por contraseñas, así como su diferenciación, atendiendo al rol de cada uno, para asegurar que cada usuario puede acceder solamente a los lugares que su rol se lo permite.
- ✓ Se propone que las contraseñas de los usuarios solo podrán ser cambiadas por los usuarios cada cierto tiempo, o por los administradores del sistema, ante cualquier situación anormal.
- ✓ El sistema permitirá a través de una bitácora de sucesos, llevar una traza de todas las operaciones realizadas por cada uno de los usuarios, mediante un registro de actividades en todo momento.
- ✓ La fidelidad de los datos es otro de los aspectos a tener en cuenta. Esta se logra mediante el cifrado de la información proveniente de la comunicación entre los componentes internos del sistema y otros sistemas que soliciten información desde otra Institución Hospitalaria. Esto impide la lectura o modificación de los datos confidenciales que se manejan y de esta forma, aumenta la seguridad**.**

**2.7 Vista de Despliegue.**

El proceso de desarrollo de software, ofrece un conjunto de modelos que permiten expresar el producto desde cada una de las perspectivas de interés. Entre estos modelos se encuentra el diagrama de despliegue, el cual muestra la forma en la cual, la aplicación se va a desplegar a través de una infraestructura.

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

Para la implantación y la utilización de la aplicación en una institución hospitalaria, el usuario debe conectarse a esta mediante una PC cliente utilizando un navegador web. Las peticiones por el protocolo HTTP serán procesadas por el servidor de aplicaciones que enviará la respuesta al cliente y en caso que

sea necesario también hará peticiones mediante el protocolo TCP/IP al servidor de base de datos. El usuario puede imprimir reportes o gráficos desde la PC cliente, utilizando una impresora.

En la Figura 4 se representa el diagrama que muestra una propuesta de cómo quedará desplegado el Módulo Logopedia perteneciente al SENDN:

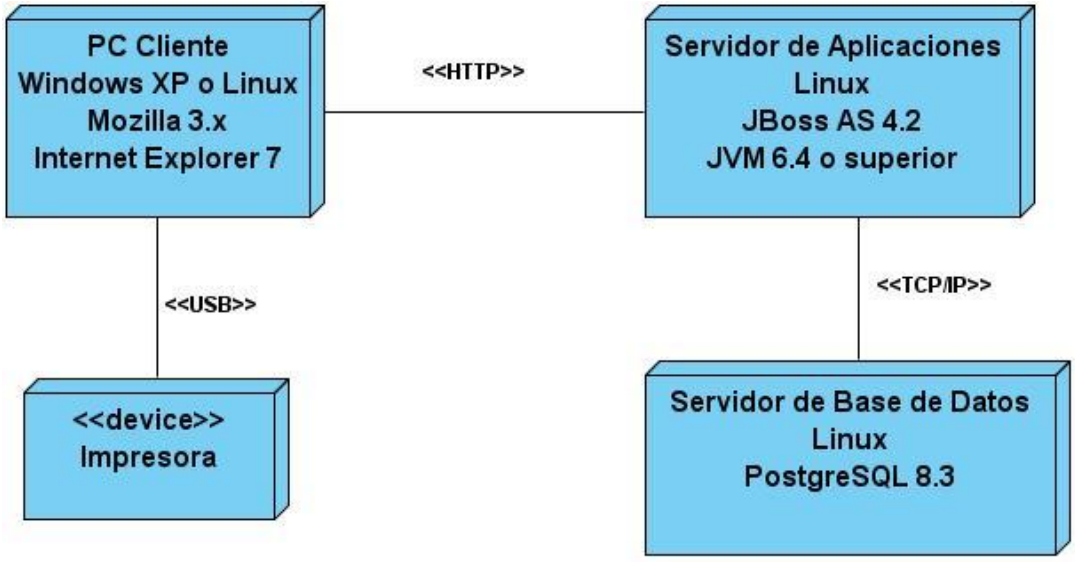


**Figura 4. Diagrama de Despliegue del Módulo Logopedia del SENDN.**

### 2.8 Estrategias de codificación. Estándares y estilos a utilizar.

El mejor método para lograr la legibilidad del código implementado por más de una persona, es la aplicación de un estándar de codificación. Un estándar de codificación completo, debe comprender todos los aspectos de la generación de código. Estos estándares se utilizan con el fin de lograr organización, uniformidad y homogeneidad como si un único programador hubiese escrito todo el código. La aplicación del mismo, también ayuda a evitar las confusiones y el no entendimiento del código entre todos los desarrolladores de un proyecto y el resto del personal que debe revisarlo.

Al aplicar correctamente y de forma sistemática un estándar de codificación bien definido, desde el principio hasta el final del proyecto de software, se asegura que el producto se convierta en un sistema de software fácil de comprender y de mantener, generando un código con calidad y organización,

minimizando los errores a la hora de entenderlo; ya sea para su revisión, para realizarle modificaciones, o agregarle más funcionalidades al sistema.

Se incluyen en el estándar el uso de las notaciones:

- ✓ CamellCasing, para las variables y métodos con nombres compuestos por múltiples palabras juntas, el cual sugiere iniciar cada palabra con letra mayúscula, excepto la primera palabra que debe iniciar con minúscula.
- ✓ PascalCasing, para las clases con nombres compuestos por múltiples palabras juntas, el cual sugiere iniciar cada palabra con letra mayúscula.

A continuación, se muestran algunas restricciones de la nomenclatura del código haciendo uso de los estándares de codificación mencionados anteriormente:

**2.8.1 Idioma.**

Se debe utilizar como idioma el español, aunque las palabras no se acentuarán.

**2.8.2 Indentación.**

Su objetivo es 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 o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay solo una instrucción. Nunca colocar ({) en la línea de un código cualquiera, esto requiere una línea propia.

### 2.8.3 Clases y Objetos.

- ✓ *Apariencia de clases y objetos.*

Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto, se empleará notación PascalCasing. Para estos nombres, no se empleará la acentuación de las palabras.

Ejemplo: MiClase ()

Para el caso de las instancias se comenzará con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.

- ✓ *Apariencia de atributos.*

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. Para los nombres de estos atributos no se empleará la acentuación.

- ✓ *Declaración de parámetro en funciones.*

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.*

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

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

- ✓ *Ubicación de comentarios.*

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

- ✓ *Líneas en blanco.*

Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura, así como, de la implementación de una función.

- ✓ *Espacios en blanco.*

Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código.

Ejemplo: item = nomitem.

- ✓ *Aspectos generales.*

*Sobre 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 de cerrar un arreglo.
- Después del paréntesis abierto y antes del paréntesis cerrado.
- Antes de un punto y coma.

**2.8.5 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.

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

Ejemplo: ConfugurarNomencladorEdad.

*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: configurarNomencladorEdad.

**2.8.6 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. Además de que, los mismos no se acentuarán.

Ejemplo: Nueva_Linea_Base.

- ✓ *Apariencia de las vistas.*

Las dos primeras letras representan el módulo. Todas las letras en minúscula. El nombre a emplear para las vistas deben 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. La totalidad de las letras deben estar denotadas en minúsculas. 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 escritas 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 escritas 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: “modl_n_percentil”.

- ✓ *Apariencia de los campos.*

Todas las letras en minúsculas. 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_lenguaje”.

- ✓ *Nombre de los campos.*

En caso de identificadores; todos los campos de los mismos van a comenzar con “id”, seguido de underscoard y posteriormente el nombre del campo.

Ejemplo: id_ lenguaje.

- ✓ *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. Además, no se acentuarán los nombres de los mismos, cuando las palabras empleadas lo requieran.

En este capítulo se realizó la descripción de la arquitectura de software, detallando los patrones de arquitectura, los modelos utilizados, y las tecnologías presentes en cada capa de los modelos, para garantizar el desarrollo eficiente de las funcionalidades. Se describieron además, los Requerimientos No

Funcionales, ya que estos son de gran importancia por ser los encargados de garantizar la estructura, presencia y eficiencia del sistema. Por otra parte, se explicó lo referente al tema de la seguridad del software, basándose principalmente en el nivel de acceso de los usuarios. Se tuvo en cuenta que la información con la cual se va a trabajar es muy sensible y puede verse comprometida con el acceso de usuarios no autorizados, que puedan manipular o dañar los datos; provocando los mayores riesgos si no se maneja de manera correcta.

Además, se proporcionó la vista de despliegue, que son las encargadas de visualizar cómo el sistema quedará implementado en la infraestructura donde se va a usar y los componentes que serán utilizados durante el despliegue del mismo. Por último, se describieron las estrategias y de codificación, que serán usadas durante todo el desarrollo del sistema propuesto. Estas estrategias, permitirán que el código fuente quede implementado de forma tal, que garantice la legibilidad, organización y uniformidad del mismo.

## CAPÍTULO 3. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA PARA DESARROLLAR EL TEST IPP Y CONFIGURACIÓN DEL MÓDULO LOGOPEDIA.

La elaboración del presente capítulo persigue como objetivos: estudiar la propuesta de diseño obtenida durante el flujo de trabajo “Análisis y Diseño”; ya que para poder darle continuidad al proceso de desarrollo de software, es necesario realizar un análisis valorativo de todo lo realizado en flujos de trabajos anteriores. Todo esto se logra mediante un proceso de refinamiento, para adaptar los nuevos requerimientos y necesidades al sistema. Para ello, se realizará una descripción de las nuevas clases y operaciones que son necesarias agregar para implementar los casos de uso. Además, se obtendrá la vista de implementación para mostrar los elementos físicos del sistema, mediante componentes. Por último, se muestra el diagrama de componentes realizado para desglosar los diferentes tipos de datos dentro del sistema e identificar las relaciones existentes entre ellos.

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

Antes de desarrollar las funcionalidades del Módulo Logopedia del SENDN, correspondiente al flujo de trabajo “Implementación”; se hace necesario realizar un análisis y valoración de las actividades llevadas a cabo en el flujo de trabajo anterior. Al realizar el análisis de los artefactos generados por el analista durante el flujo de trabajo “Análisis y Diseño”, se llegó a la conclusión de que existían varios de ellos, que no cumplían con las nuevas especificaciones del sistema. Por lo antes planteado, se hizo necesario realizar un refinamiento del diseño, lo cual incluía la modificación de los Diagramas de Clases del Diseño y los Diagramas de Secuencia; con el objetivo de añadir los nuevos métodos y atributos que cumplieran con las nuevas necesidades.

Se hizo necesario además, la modificación del diagrama de clases persistentes de la base de datos, añadiendo atributos, clases y relaciones generando nuevamente el modelo de datos y el script para la base de datos. Por otra parte, la propuesta de diseño carecía de componentes necesarios para poder desarrollar los casos de uso del sistema, además de no cumplir con las pautas de diseño establecidas por el Departamento de Gestión Hospitalaria de la facultad 7. Debido a esto, se realizó un proceso de reestructuración de la propuesta de diseño, para luego desarrollar las funcionalidades: realizar Test IPP y la configuración del Módulo Logopedia.

### 3.1.1 Diagramas de clases del diseño.

"El Diagrama de Clase del Diseño (DCD), es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones." (37)

"El DCD describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información:

- ✓ Clases, asociaciones y atributos.
- ✓ Interfaces, con sus operaciones y constantes.
- ✓ Métodos.
- ✓ Información sobre los tipos de los atributos.
- ✓ Navegabilidad.
- ✓ Dependencias."(38)

"A continuación se describen los estereotipos web utilizados para modelar los diagramas de clases:

- ✓ *Página servidora:* representa la página web que tiene código que se ejecuta en el servidor y que interactúa con recursos en el mismo. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.
- ✓ *Página cliente:* una instancia de página cliente es una página web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador y sus atributos son las variables declaradas dentro del script, que son accesibles para páginas con cualquier función dentro de esta.
- ✓ *Formulario:* colección de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de

entrada del formulario (input boxes, text areas, radio buttons, check boxes y hidden fields). No tienen operaciones.

- ✓ *Build:* representa una asociación especial que relaciona las páginas clientes con las páginas servidoras, de forma general se expresa como que las páginas que se encuentran en el servidor construyen las páginas en el cliente. Debe ser una relación direccional, donde una página servidor puede construir una o más páginas cliente.
- ✓ *Submit:* es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.
- ✓ *Redirect:* la página de servidor además de construir una página cliente puede redireccionar el procesamiento a otra página. Esto se representa con una asociación con el estereotipo <<redirect>>."(39)

En la Figura 5 se presenta el DCD del caso de uso "Crear Área" perteneciente a la configuración del Módulo Logopedia.

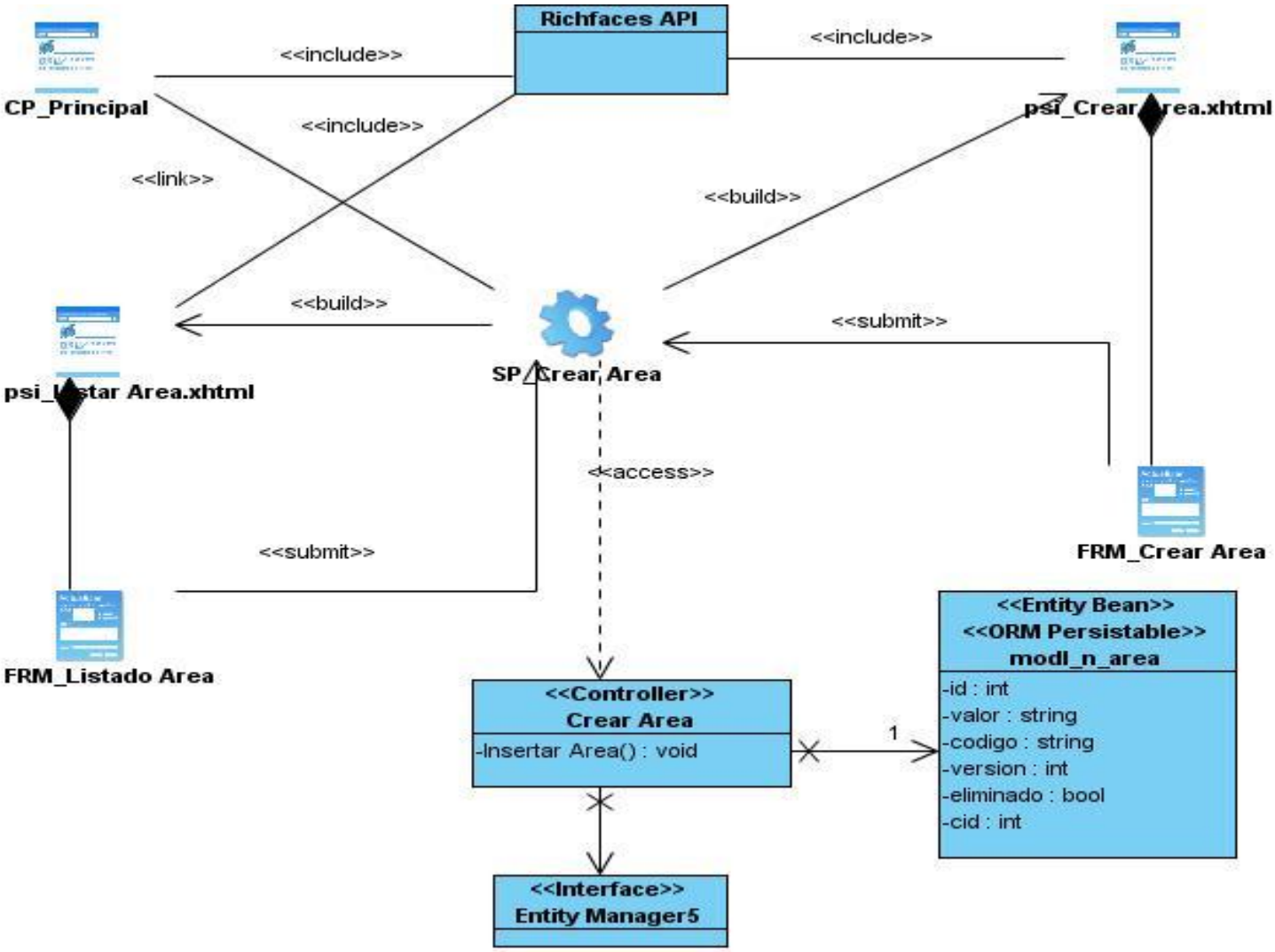


**Figura 5. DCD del caso de uso Crear Área.**

En la Figura 6, se puede apreciar el DCD correspondiente al caso de uso “Crear Test IPP”.

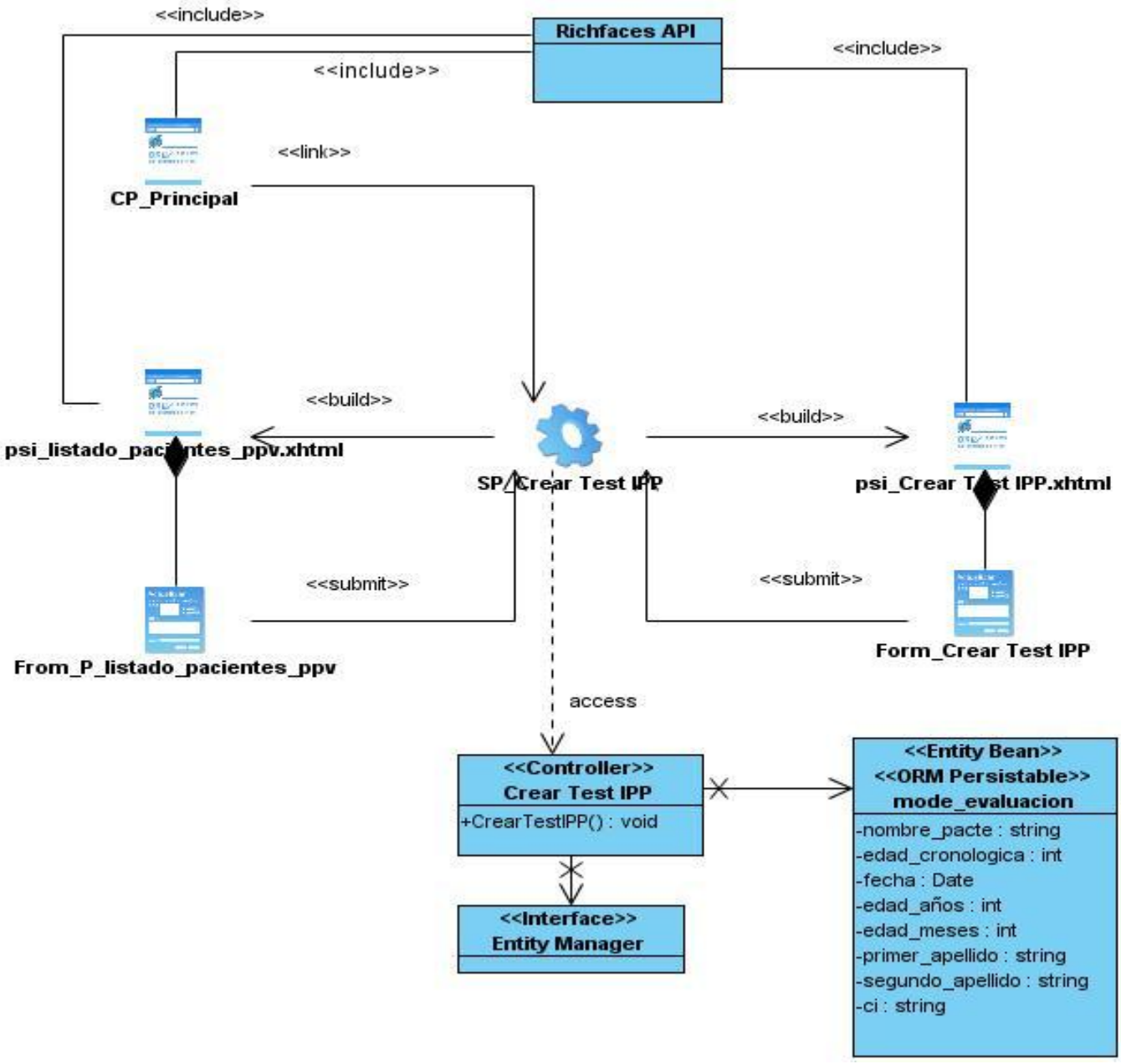


**Figura 6. DCD del caso de uso Crear Test IPP.**

### 3.1.2 Diagramas de interacción.

"Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento. Un diagrama de interacción capta el comportamiento de un solo caso de uso. El diagrama muestra cierto número de ejemplos de objetos y los mensajes que se pasan entre estos objetos dentro del caso de uso."(40) Son importantes para modelar los aspectos dinámicos de un sistema y para construir sistemas ejecutables a través de ingeniería hacia adelante e ingeniería inversa. Comúnmente contienen:

- ✓ Objetos.
- ✓ Enlaces.
- ✓ Mensajes.

"Pueden servir para visualizar, especificar, construir y documentar los aspectos dinámicos de una sociedad particular de objetos, o pueden ser usados para modelar un flujo particular de control de un caso de uso."(41)

Los diagramas de interacción están conformados por los diagramas de secuencia y los diagramas de colaboración:

- ✓ Diagramas de colaboración: muestran las relaciones entre los objetos y los mensajes que intercambian.
- ✓ Diagramas de secuencia (DS): muestran las interacciones expresadas en función de secuencias temporales.

En la Figura 7, se puede apreciar el DS del caso de uso "Crear Área", perteneciente a la configuración del Módulo Logopedia.

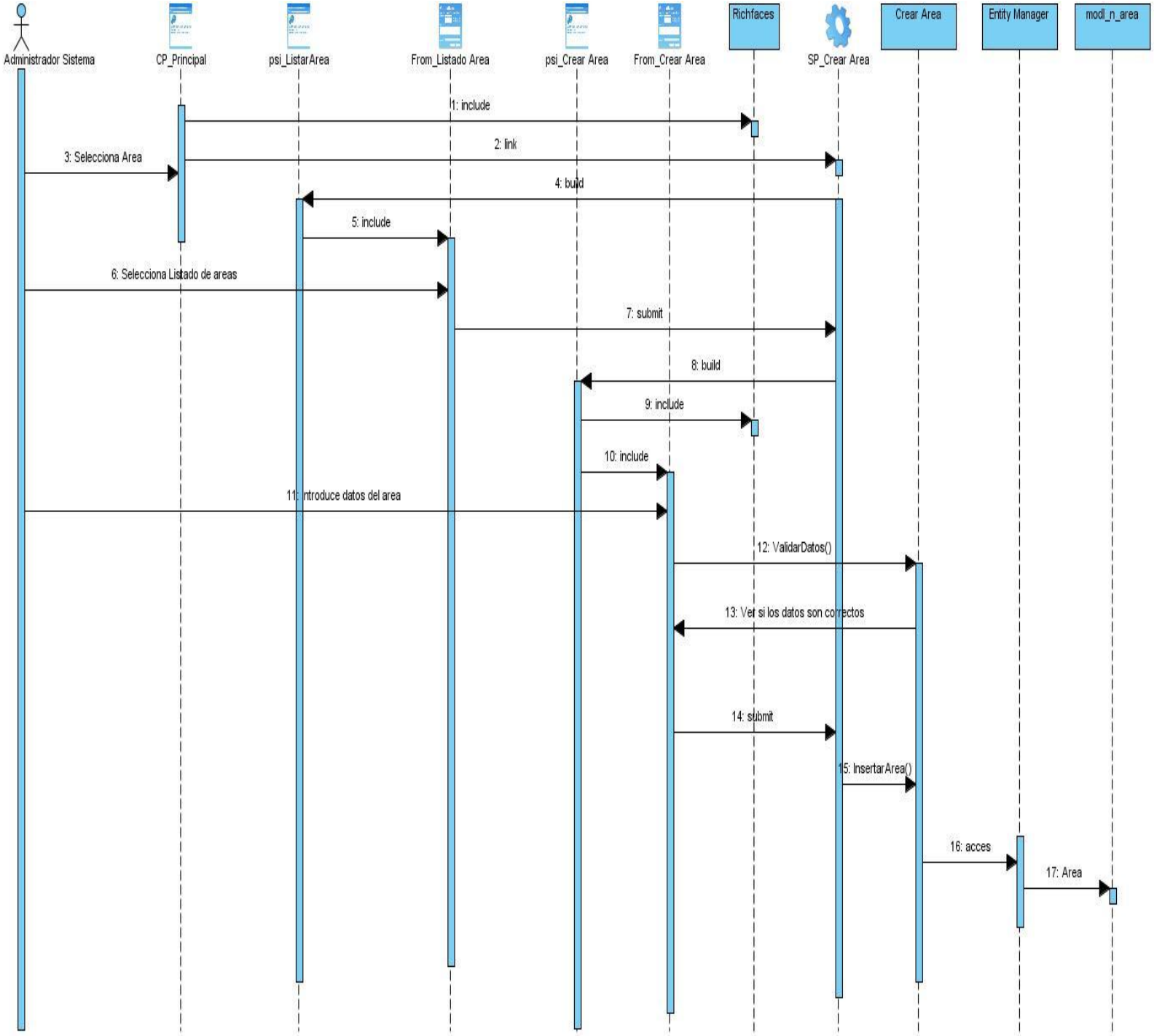
Administrador Sistema
CP_Principal
psi_ListarArea
From_Listado Area
psi_Crear Area
From_Crear Area
Richfaces
SP_Crear Area
Crear Area
Entity Manager
modl_n_area
1: include
2: link
3: Selecciona Area
4: build
5: include
6: Selecciona Listado de areas
7: submit
8: build
9: include
10: include
11: Introduce datos del area
12: ValidarDatos()
13: Ver si los datos son correctos
14: submit
15: InsertarArea()
16: acces
17: Area

**Figura 7. DS del caso de uso Crear Área.**

En la Figura 8, se puede apreciar el DS correspondiente al caso de uso "Crear Test IPP", que se inicia cuando se selecciona un paciente del listado para aplicarle el test.

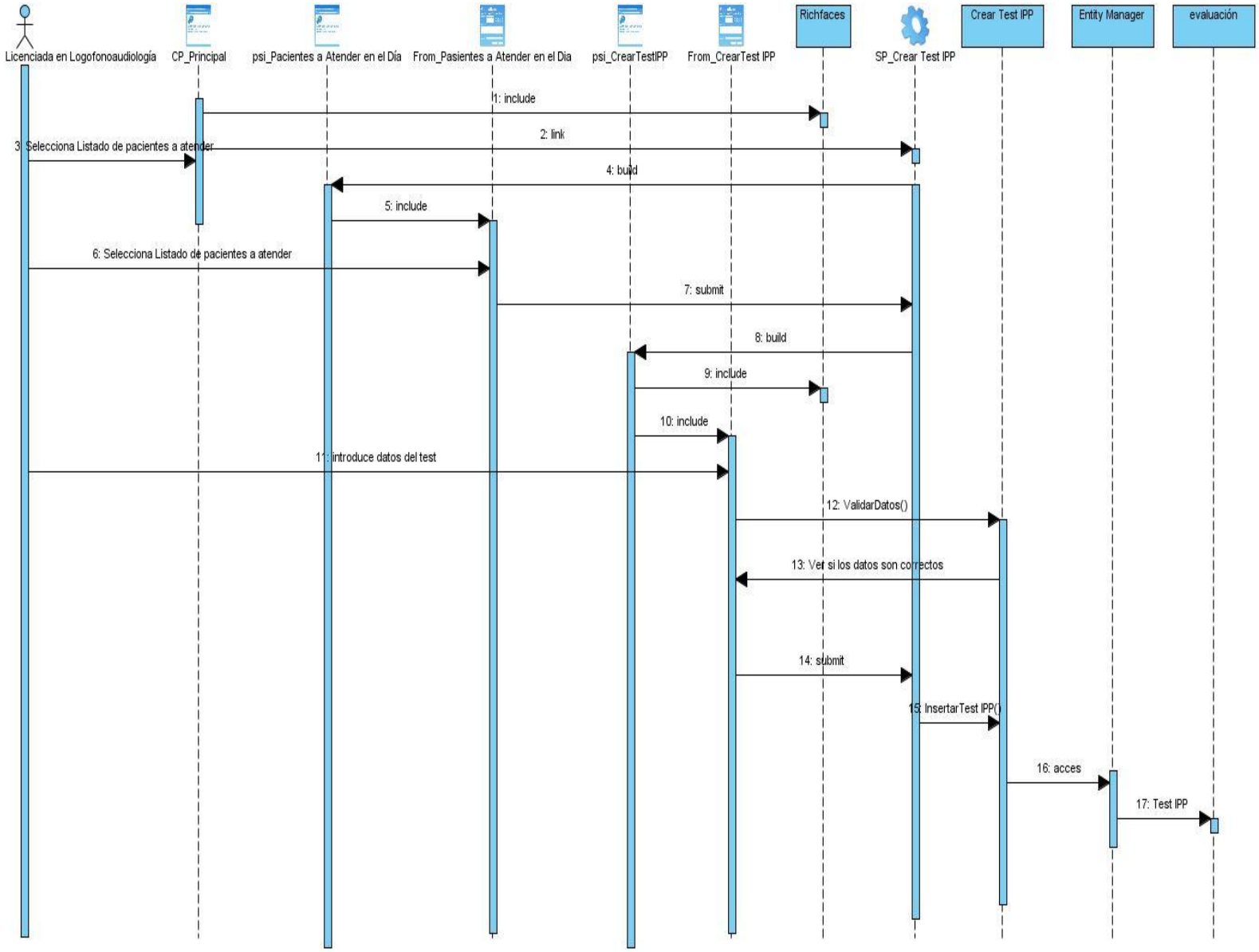

**Figura 8. DS del caso de uso Crear Test IPP.**

### 3.2 Descripción de las nuevas clases u operaciones necesarias.

A continuación se describirán solo algunas de las nuevas clases que son necesarias para poder desarrollar las funcionalidades. De acuerdo a la utilización de los estándares y estilos de codificación descritos en el Capítulo II, los atributos, métodos y nombres de las clases, no se acentúan. *(Ver Tablas 2, 3, 4, 5, 6)*

**Tabla 2. Descripción de la clase Configurar nomenclador Área.**

| **Nombre: ConfigurarNomencladorArea** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| id | Integer |
| eliminado | Boolean |
| nomenclador | ModlNAreas |
| **Para cada responsabilidad**: | |
| Nombre: | insertarArea() |
| Descripción: | Inserta los datos de área. |
| Nombre: | eliminarArea() |
| Descripción: | Elimina los datos de área. |
| Nombre: | modificarArea() |
| Descripción: | Modifica los datos de área. |

**Tabla 3. Descripción de la clase Configurar nomenclador Edad.**

| **Nombre: ConfigurarNomencladorEdad** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| id | Integer |
| eliminado | Boolean |
| nomenclador | ModlNEdad |
| **Para cada responsabilidad**: | |
| Nombre: | insertarEdad() |
| Descripción: | Inserta los datos de edad. |
| Nombre: | eliminarEdad() |
| Descripción: | Elimina los datos de edad. |
| Nombre: | modificarEdad() |

| | |
|---|---|
| Descripción: | Modifica los datos de edad. |

**Tabla 4. Descripción de la clase Configurar nomenclador Clasificación.**

| **Nombre: ConfigurarNomencladorClasificacion** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| id | Integer |
| eliminado | Boolean |
| nomenclador | ModlNClasificacion |
| **Para cada responsabilidad**: | |
| Nombre: | insertarClasificacion () |
| Descripción: | Inserta los datos de clasificación. |
| Nombre: | eliminarClasificacion () |
| Descripción: | Elimina los datos de clasificación. |
| Nombre: | modificarClasificacion () |
| Descripción: | Modifica los datos de clasificación. |

**Tabla 5. Descripción de la clase Configurar nomenclador Listas Palabras.**

| **Nombre: ConfigurarNomencladorListasPalabras** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |
| id | Integer |
| eliminado | Boolean |
| nomenclador | ModlNListasPalabras |
| **Para cada responsabilidad**: | |
| Nombre: | insertarListasPalabras() |
| Descripción: | Inserta los datos de listas palabras. |
| Nombre: | eliminarListasPalabras () |
| Descripción: | Elimina los datos de listas palabras. |
| Nombre: | modificarListasPalabras () |
| Descripción: | Modifica los datos de listas palabras. |

**Tabla 6. Descripción de la clase Configurar nomenclador Lenguaje Valor.**

| **Nombre: ConfigurarNomencladorLenguajeValor** | |
|---|---|
| **Tipo de clase: Controladora** | |
| **Atributo** | **Tipo** |

| **Nombre: ConfigurarNomencladorLenguajeValor** | |
|---|---|
| id | Integer |
| eliminado | Boolean |
| valor | String |
| nomenclador | ModlNLenguajeValor |
| **Para cada responsabilidad**: | |
| Nombre: | insertarLenguajeValor() |
| Descripción: | Inserta los datos de lenguaje valor. |
| Nombre: | eliminarLenguajeValor() |
| Descripción: | Elimina los datos de lenguaje valor. |
| Nombre: | modificarLenguajeValor() |
| Descripción: | Modifica los datos de lenguaje valor. |

### 3.3 Modelo de datos.

Un modelo de datos, es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. El cual puede ser usado como una herramienta para especificar los tipos de datos y la organización de los mismos.

"El modelo de datos es una combinación de tres componentes:

- ✓ una colección de estructuras de datos (los bloques constructores de cualquier base de datos que conforman el modelo).
- ✓ una colección de operadores o reglas de inferencia, los cuales pueden ser aplicados a cualquier instancia de los tipos de datos listados, para consultar o derivar datos de cualquier parte de estas estructuras en cualquier combinación deseada.
- ✓ una colección de reglas generales de integridad, las cuales explícita o implícitamente definen un conjunto de estados."(42)

### 3.4 Breve descripción de las tablas presentes en la base de datos.

A continuación, se describen algunas de las nuevas técnicas de validación que son necesarias para poder desarrollar las funcionalidades. De acuerdo a la utilización de los estándares y estilos de codificación

descritos en el Capítulo II, los nombres de las tablas de la base de datos no se acentúan. *(Ver tablas 7, 8, 9, 10, 11)*

**Tabla 7. Descripción de la tabla de la base de datos que almacena la información de las áreas.**

| **Nombre:** modl_n_areas | | |
|---|---|---|
| **Descripción:** Esta tabla permite conocer la edad del paciente. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer | Identificador de la tabla. |
| valor | String | Atributo que permite buscar por el valor en el listado. |
| codigo | String | Atributo que permite buscar por el código en el listado. |
| version | Integer | Atributo para persistir o actualizar la entidad. |
| eliminado | Boolean | Permite verificar si el área ha sido eliminada. |
| cid | Integer | Identificador de modificaciones registradas en la bitácora. |

**Tabla 8. Descripción de la tabla de la base de datos que almacena la información de las edades.**

| **Nombre:** modl_n_edad | | |
|---|---|---|
| **Descripción:** Esta tabla contiene información sobre las edades. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer | Identificador de la tabla. |
| valor | String | Atributo que permite buscar por el valor en el listado. |
| codigo | String | Atributo que permite buscar por el código en el listado. |
| version | Integer | Atributo para persistir o actualizar la entidad. |
| eliminado | Boolean | Permite verificar si la edad ha sido eliminada. |
| cid | Integer | Identificador de modificaciones registradas en la bitácora. |

**Tabla 9. Descripción de la tabla de la base de datos que almacena la información del lenguaje.**

| **Nombre:** modl_n_lenguaje | | |
|---|---|---|
| **Descripción:** tabla que contiene información sobre el lenguaje. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer | Identificador de la tabla. |
| valor | String | Atributo que permite buscar por el valor en el |

| | | |
|---|---|---|
| | | listado. |
| codigo | String | Atributo que permite buscar por el código en el listado. |
| version | Integer | Atributo para persistir o actualizar la entidad. |
| eliminado | Boolean | Permite verificar si el lenguaje ha sido eliminado. |
| cid | Integer | Identificador de modificaciones registradas en la bitácora. |

**Tabla 10. Descripción de la tabla de la base de datos que almacena la información de las clasificaciones.**

| **Nombre:** modl_n_clasificacion | | |
|---|---|---|
| **Descripción:** tabla que contiene información sobre las clasificaciones. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer | Identificador de la tabla. |
| valor | String | Atributo que permite buscar por el valor en el listado. |
| codigo | String | Atributo que permite buscar por el código en el listado. |
| version | Integer | Atributo para persistir o actualizar la entidad. |
| eliminado | Boolean | Permite verificar si la clasificación ha sido eliminada. |
| cid | Integer | Identificador de modificaciones registradas en la bitácora. |

**Tabla 11. Descripción de la tabla de la base de datos que almacena la información de las listas de palabras del Test IPP.**

| **Nombre:** modl_n_listas_palabras | | |
|---|---|---|
| **Descripción:** Tabla que contiene un listado de palabras. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer | Identificador de la tabla. |
| valor | String | Atributo que permite buscar por el valor en el listado. |
| codigo | String | Atributo que permite buscar por el código en el listado. |
| version | Integer | Atributo para persistir o actualizar la entidad. |
| eliminado | Boolean | Permite verificar si las listas de palabras han |

| | | |
|---|---|---|
| | | sido eliminadas. |
| cid | Integer | Identificador de modificaciones registradas en la bitácora. |

**3.5 Vista de Implementación.**

La vista de implementación muestra el empaquetado físico de las partes reutilizables del sistema en unidades sustituibles, llamadas componentes. Una vista de implementación muestra los elementos físicos del sistema mediante componentes, así como sus interfaces y dependencias entre ellos. Los componentes son piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas.

"El diagrama de componentes describe la descomposición física del sistema en componentes, a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos. Los componentes identifican objetos físicos que hay en tiempos de ejecución, de compilación o de desarrollo y tienen identidad propia con una interfaz bien definida. Cada componente incorpora la implementación de ciertas clases del diseño del sistema." (43) La Figura 9 muestra el diagrama de componentes en paquetes, perteneciente al Módulo Logopedia.

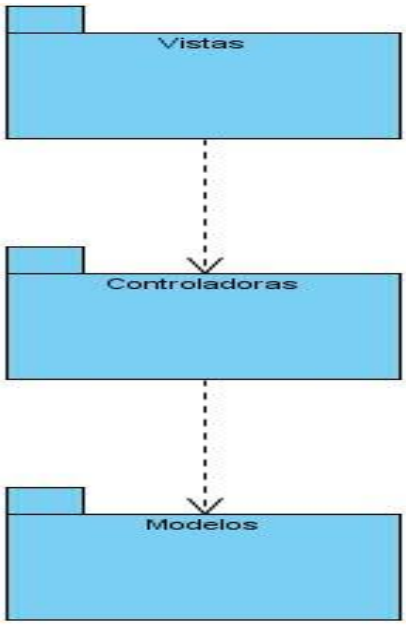

**Figura 9. Diagrama de componentes Logopedia (Paquetes).**

En el paquete Vistas residen todas las páginas XHTML, las cuales son las encargadas de brindarles la información a los usuarios en forma de interfaces, para viabilizar la interacción de los mismos con el sistema. Todas estas páginas se integran con el Richfaces, el cual es una biblioteca de componentes que se utiliza para facilitar la creación de aplicaciones web, como se puede apreciar en la Figura 10.

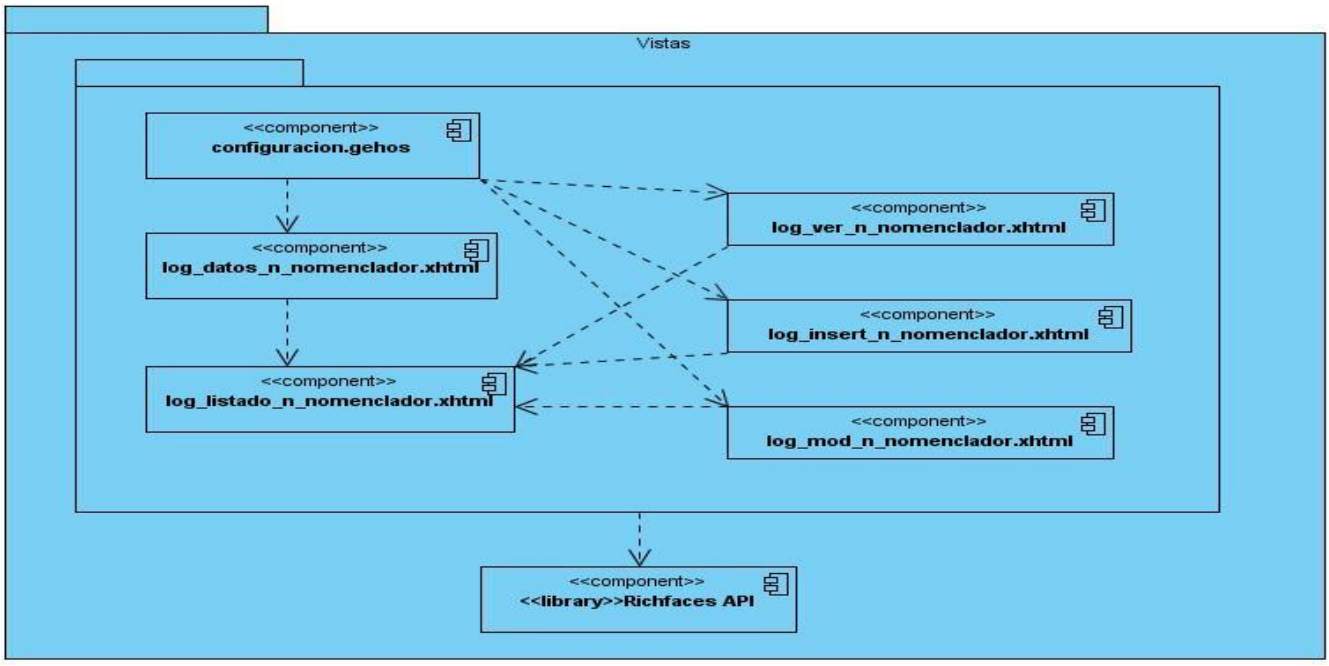


**Figura 10. Diagrama de Componentes Logopedia (Paquete Vistas).**

En el paquete Modelos, residen todas las tablas de la base de datos que son generadas mediante el mapeo de los datos, utilizando la herramienta ORM Hibernate. Las mismas, se pueden observar en la Figura 11.

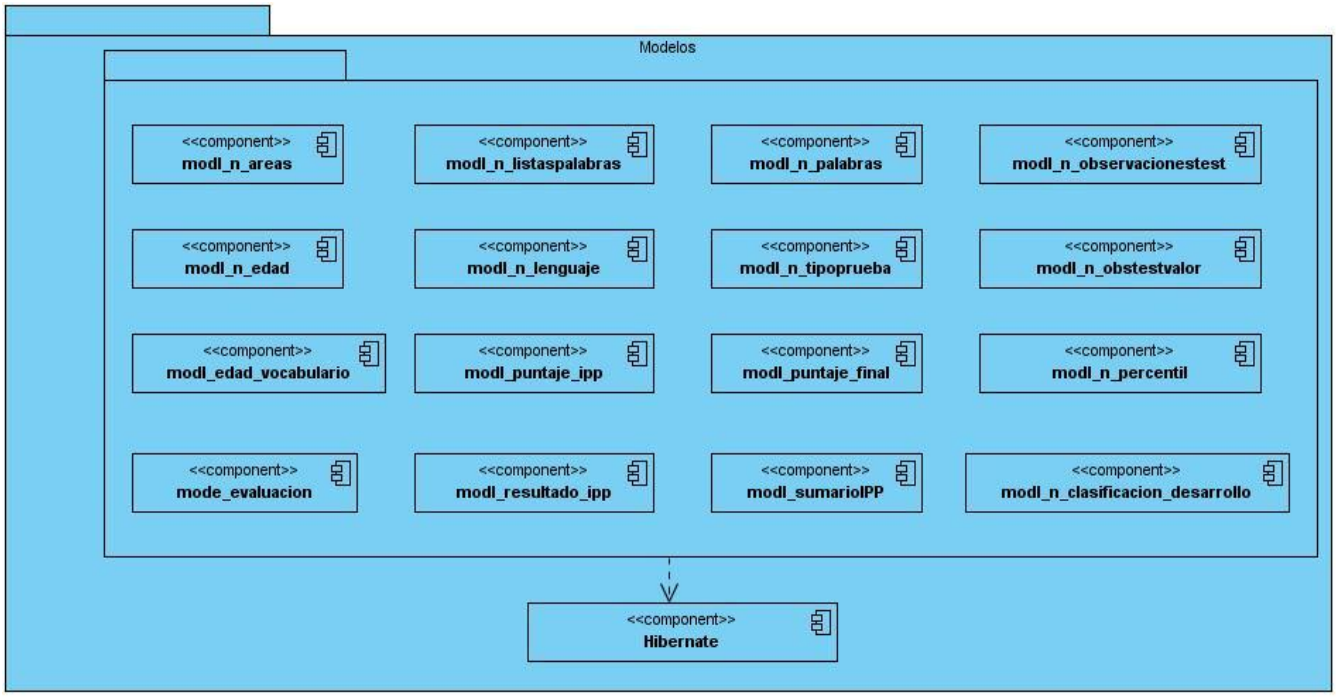


**Figura 11. Diagrama de Componentes Logopedia (Paquete Modelos).**

Por su parte, el paquete Controladoras, consta con las clases creadas utilizando el lenguaje de programación Java, que son empleadas para desarrollar todas las funcionalidades y en las cuales se implementan todos los métodos para lograr el funcionamiento del sistema. Todas ellas utilizando el framework Seam, para propiciar la integración con los demás componentes. Este paquete se puede apreciar en la Figura 12.

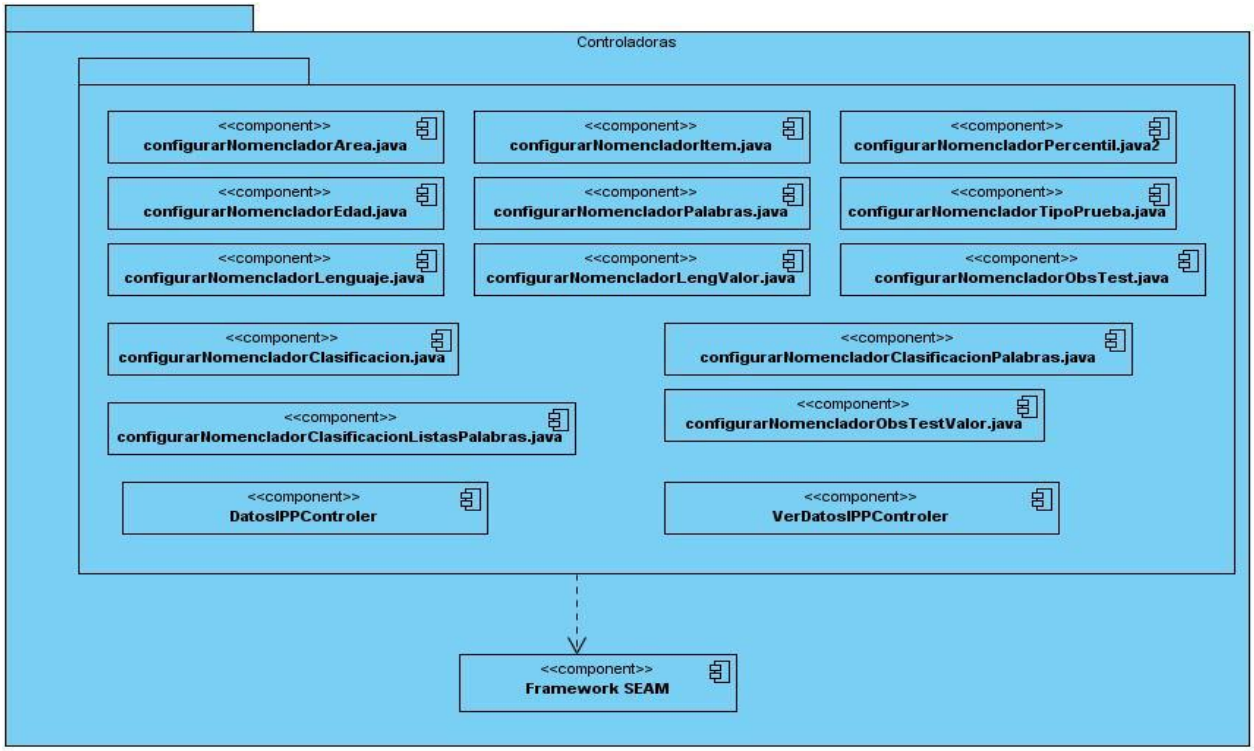


**Figura 12. Diagrama de Componentes Logopedia (Paquete Controladoras).**

En este capítulo se ha descrito y analizado la propuesta de solución del problema a resolver, planteado para este trabajo. Se ha realizado una valoración crítica del diseño propuesto por el analista, arribándose a la conclusión de que era necesario incluir nuevas clases, operaciones y métodos; para poder desarrollar de manera eficiente, los requerimientos funcionales y no funcionales. Han sido presentadas las clases del diseño que participan en la realización de los casos de uso de la aplicación propuesta, los diagramas de secuencia del diseño y la descripción textual de cada una de las clases controladoras donde se implementan las principales funcionalidades. Así, como una descripción de los métodos y atributos presentes en cada una de las tablas de la base de datos.

Por otra parte, se obtuvo el modelo de datos del sistema, en el cual se muestran todos los objetos que están presentes en la base de datos, así como la relación que existe entre ellos. Se proporcionaron además, las vistas de la implementación realizadas de acuerdo al patrón de arquitectura MVC; donde se

mostró el diagrama de componentes y la separación de los paquetes, mostrando los objetos que residen en cada uno de ellos.

## CAPÍTULO 4. REALIZACIÓN DE LAS PRUEBAS A LAS FUNCIONALIDADES: REALIZAR TEST IPP Y LA CONFIGURACIÓN DEL MÓDULO LOGOPEDIA.

En la actualidad, la calidad es un tema importante en el desarrollo de algún producto o servicio; ya que se puede indicar que algún software posee la misma, cuando satisface las necesidades y expectativas del cliente o usuario. Para dar continuidad al ciclo de vida del software y habiendo implementado las funcionalidades: realizar Test IPP y la configuración del Módulo Logopedia, se procede a realizarle las pruebas pertinentes para verificar el correcto funcionamiento de las mismas. Es por ello, que en el presente capítulo se describirá el método de caja negra para probar las funcionalidades del sistema propuesto, haciendo uso de la técnica de partición equivalente. Además, se definirán y detallarán los casos de pruebas asociados a las funcionalidades implementadas, así como la descripción de las tablas que conforman un caso de prueba.

### 4.1 Pruebas de caja negra.

Para obtener un software excelente es necesario que se tenga un buen modelo de pruebas. Con las pruebas no se puede asegurar la ausencia de defectos en un software, pero sí se puede demostrar que presenta defectos. Un buen diseño de prueba, evidencia diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. Para garantizar la efectividad de las pruebas, deben ser realizadas por un equipo independiente al que realizó el sistema.

“Las pruebas de caja negra, se llevan a cabo sobre la interfaz del software y son completamente indiferentes al comportamiento interno y la estructura del programa. Con estas se pretende demostrar que las funcionalidades del sistema son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que se mantiene la integridad de la información externa.”(44)

*“Las principales características de estas pruebas son:*

- ✓ Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- ✓ Son realizadas sin el conocimiento interno del producto.

- ✓ No validan funciones ocultas (por ejemplo funciones implementadas, pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados." (45)

En la presente investigación, para probar el correcto funcionamiento de las funcionalidades implementadas, se realizarán pruebas aplicando el método de caja negra, utilizando como técnica la partición equivalente; la cual, divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de casos de prueba para la partición equivalente, se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia, representa un conjunto de estados válidos o inválidos para condiciones de entrada. Típicamente, una condición de entrada puede ser un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (si o no).

**4.2 Descripción de los casos de prueba.**

"Los casos de prueba, representan los datos que se utilizarán como entrada para ejecutar el software a probar. Determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Por lo tanto, durante la tarea de generación de casos de prueba, se han de confeccionar los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba, debe ir acompañada del resultado que ha de producir el software al ejecutarlo."(46)

Entre los componentes con los que cuentan las tablas que comprenden a los casos de prueba, se encuentran: las secciones (SC), que se refieren a las diferentes funcionalidades que se encuentran dentro del caso de uso; los escenarios (EC), que son los diferentes resultados que se obtendrán cuando se realicen las operaciones dentro de la sección a la que pertenecen. *(Ver tabla 12)*

**Tabla 12. Descripción de las secciones que se van a probar en el caso de uso Gestionar Nomenclador Edad y de los escenarios que están presentes en cada sección.**

| Nombre de la sección | Escenarios de la sección | Descripción de la funcionalidad |
|---|---|---|
| SC 1: Adicionar Nomenclador Edad. | EC 1.1: Adicionar Nomenclador Edad exitosamente. | Se agrega un nuevo Nomenclador Edad con los datos correspondientes. |
| | EC1.2: Existen campos incompletos | No se agrega el Nomenclador Edad porque existen campos incompletos. |
| | EC 1.3: Existen campos incorrectos. | No se agrega el *Nomenclador Edad* porque existen campos incorrectos. |
| SC 2: Ver datos del Nomenclador Edad. | EC 2.1: Ver datos del Nomenclador Edad exitosamente. | Se visualiza los datos del Nomenclador Edad escogido. |
| | EC 2.2: Salir del Nomenclador Edad exitosamente. | Se visualiza el listado de los nomencladores de Edad. |
| SC 3: Modificar Nomenclador Edad. | EC 3.1: Modificar Nomenclador Edad exitosamente. | Se modifica un nuevo Nomenclador Edad con los datos correspondientes. |
| | EC3.2: Existen campos incompletos | No se modifica el Nomenclador Edad porque existen campos incompletos. |
| | EC3.3: Existen campos incorrectos. | No se modifica el *Nomenclador Edad* porque existen campos incorrectos. |
| SC 4: Eliminar Nomenclador Edad. | EC4.1: Eliminar el Nomenclador Edad. | Se elimina el Nomenclador Edad exitosamente. |
| | EC4.2: No Eliminar Nomenclador Edad. | No se elimina el Nomenclador Edad. |
| SC5: Buscar Nomenclador Edad. | EC5.1: Busca el Nomenclador Edad exitosamente. | Lista el nomenclador Edad a buscar, sino encuentra por los parámetros especificados sale un mensaje informando: “No se encontró información que cumpla con los |

| Nombre de la sección | Escenarios de la sección | Descripción de la funcionalidad |
|---|---|---|
| | | criterios de búsqueda". |

En la tabla 13, se describen las variables que se encuentran declaradas en la funcionalidad que se va a probar. En ella, se describe el tipo, su clasificación y el nombre que recibe la misma.

**Tabla 13. Descripción de las variables declaradas en el caso de uso Gestionar Nomenclador Edad.**

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Código | Campo de texto | No | Se admiten números y letras. Min=1 Max= 20 |
| 2 | Valor | Campo de texto | No | Se admiten números y letras. Min=1 Max= 20 |

Por otra parte, se describen los diferentes EC de una SC determinada y la relación que tienen con las variables declaradas, en lo que se denomina matriz de datos *(ver tabla 14)*; donde se mencionan los diferentes valores que toma la variable de acuerdo al EC. Los valores que puede tomar son:

- V: Válido.
- I: Inválido.
- NA: No aplica.

**Tabla 14. Matriz de datos de la sección Adicionar Nomenclador Edad.**

| Escenario | Variable 1 *(Código)* | Variable 2 *(Valor)* | Respuesta del Sistema | Resultado de la Prueba | Flujo Central |
|---|---|---|---|---|---|

| Escenario | Variable 1 *(Código)* | Variable 2 *(Valor)* | Respuesta del Sistema | Resultado de la Prueba | Flujo Central |
|---|---|---|---|---|---|
| EC1.1: Adicionar Nomenclador Edad exitosamente. | *V* ***(CI89)*** | *V* | *El sistema adiciona un nuevo nomenclador Edad.* | | • *Menú Logopedia.*<br>• *Submenú Configuración.*<br>• *Vínculo Edad.*<br>• *Vínculo Adicionar nomenclador Edad.* |
| EC1.2: Existen campos incompletos. | *I* ***(null)*** | *V* | *El sistema muestra un carácter especial (asterisco rojo sobre el campo de entrada incompleto).* | | • *Menú Logopedia.*<br>• *Submenú Configuración.*<br>• *Vínculo Edad.*<br>• *Vínculo Adicionar nomenclador Edad.*<br>• *Mensaje de error.* |
| | *V* | *I* | | | |
| | *I* | *I* | | | |
| EC1.3: Existen campos incorrectos. | *I* ***($#)*** | *V* | *El sistema muestra un carácter especial (asterisco rojo sobre el campo de entrada incorrecto).* | | • *Menú Logopedia.*<br>• *Submenú Configuración.*<br>• *Vínculo Edad.*<br>• *Vínculo Adicionar nomenclador Edad.*<br>• *Mensaje de error.* |
| | *V* | *I* | | | |
| | *I* | *I* | | | |

El probador debe verificar el correcto funcionamiento de la aplicación de acuerdo a los parámetros establecidos en la fase de implementación, detectando y documentando los fallos que presente la misma. Este proceso de prueba se lleva a cabo en diferentes iteraciones, cada iteración muestra la revisión del sistema y la corrección de los fallos detectados.

La Tabla 15 muestra los resultados obtenidos con la ejecución de las pruebas, al realizar solamente una iteración de este proceso. Las mismas fueron realizadas por la analista Yennia Ramírez Martínez. En la 1ra columna se encuentran las funcionalidades que fueron probadas, mientras que la 2da columna muestra la cantidad de No Conformidades (NC) detectadas, la 3ra columna representa el total de recomendaciones realizadas en la funcionalidad y por último, la 4ta columna expresa el porciento de las NC resueltas.

**Tabla 15. Tabla de resultados obtenidos con la realización de las pruebas.**

| Funcionalidades | NC | Recomendaciones | Porciento de NC resueltas |
|---|---|---|---|
| Crear Test IPP | 3 | 0 | 100% |
| Ver Detalles Test IPP | 3 | 1 | 100% |
| Gestionar Nomenclador Edad | 1 | 0 | 100% |
| Gestionar Nomenclador Áreas | 0 | 0 | 0 |
| Gestionar Nomenclador Clasificación | 0 | 0 | 0 |
| Gestionar Nomenclador Clasificación Palabra | 1 | 1 | 100% |
| Gestionar Nomenclador Ítem | 1 | 1 | 100% |
| Gestionar Nomenclador Percentil | 1 | 0 | 100% |
| Gestionar Nomenclador Palabras | | | |
| Gestionar Nomenclador Listas | 1 | 0 | 100% |

| | | | |
|---|---|---|---|
| Palabras | | | |
| Gestionar Nomenclador Tipo Prueba | 0 | 0 | 0 |
| Gestionar Nomenclador Lenguaje | 0 | 0 | 0 |
| Gestionar Nomenclador Lenguaje Valor | 0 | 0 | 0 |
| Gestionar Nomenclador Observaciones Test | 1 | 0 | 100% |
| Gestionar Nomenclador Observaciones Test Valor | 1 | 0 | 100% |

El uso de mecanismos y métodos que verifiquen el correcto funcionamiento de un sistema informático, permite obtener un software de calidad y que se ajuste, a las necesidades de los clientes. Es por ello, que una vez desarrolladas las funcionalidades del Test IPP y la configuración del Módulo Logopedia, perteneciente al SENDN, se utilizó el método de Caja Negra, empleando la técnica de partición equivalente para comprobar que la aplicación efectuaba las operaciones de manera correcta. La realización de los casos de prueba sirvió de guía para el personal encargado de realizar las pruebas a las funcionalidades implementadas, viabilizando el proceso de detección de las No Conformidades.

En este capítulo se ha comprendido la importancia de las pruebas para detectar que no existan errores en el software. La mayoría de los usuarios de programas extensos no se interesan en los detalles del funcionamiento del programa y lo que buscan son las respuestas. Es por ello, la importancia que reviste la utilización de pruebas para desarrollar un software con la máxima calidad, para brindar un servicio de excelencia. Se diseñaron 15 casos de pruebas para 15 funcionalidades implementadas; los cuales fueron ejecutados en 1 iteración, encontrándose un total de 13 NC las cuales fueron resueltas en su totalidad.

**CONCLUSIONES.**

Con la realización del presente trabajo y el cumplimiento de las tareas propuestas, se arribaron a las siguientes conclusiones:

- La valoración crítica del diseño propuesto por los analistas, para el desarrollo de las funcionalidades demostró su ineficiencia, al no cumplir con las especificaciones del sistema lo que provocó la necesidad de su refinamiento.
- Se implementaron las funcionalidades: realizar Test IPP y la configuración del Módulo Logopedia, lo que permitirá aumentar la eficiencia en los procesos de atención y evaluación del neurodesarrollo, llevados a cabo en la especialidad de Logopedia en el Programa Renacer Contigo, lográndose una herramienta que podrá ser utilizada en cualquier centro médico del país, en el que se realicen los estudios de esta especialidad.
- La realización de las pruebas de caja negra aplicadas a las funcionalidades implementadas, posibilitaron mejorar el funcionamiento y estética de la aplicación. Obteniéndose un prototipo funcional que posibilitará mejorar los servicios de atención y evaluación en la especialidad de Logopedia, brindados a los pacientes de cero a cinco años de edad, egresados del Hospital Pediátrico “William Soler”, de La Habana.

**RECOMENDACIONES.**

Con el objetivo de complementar y mejorar la solución propuesta, además de ofrecer un mejor servicio a usuarios y pacientes, se recomienda:

- Agregarle al Módulo Logopedia, funcionalidades que permitan generar reportes de las consultas y de los test que se desarrollan.
- Implementar funcionalidades que muestren en los resultados finales, las categorías del Test IPP en las cuales el niño tiene posibles afectaciones, para permitir la obtención de un diagnóstico más completo y preciso.
- Adicionarle a la configuración del Módulo Logopedia, nomencladores que permitan ver el resultado de la consulta, así como los resultados de los test; facilitando la gestión de los mismos.
- Proponer a la Dirección del Hospital Pediátrico William Soler, realizar el despliegue del Módulo Logopedia en las instituciones pediátricas, con el objetivo de realizar los estudios de atención y evaluación del neurodesarrollo en esta especialidad.

**REFERENCIAS BIBLIOGRÁFICAS.**


1. **Romeo, Y., Martínez, M. y Hernández, M. D.** *Solución Informática para el Centro Nacional de Balance Alimentario: Análisis y diseño de la aplicación Web para la captura y envío de los datos requeridos por el CNBA.* La Habana: s.n., 2008.

2. SALUS. [En línea] [Citado el: 12 de febrero de 2011.] http://www.softwaresalus.com/ .

3. Docentes innovadores.net. *Estudio de los suelos en Uruguay.* [En línea] [Citado el: 12 de febrero de 2011.] http://www.docentesinnovadores.net/uncontenido.asp?id=3084 .

4. **Castro, F.** *Reunión del 19/8/2002. Tropa de futuro.* La Habana: Oficina de publicaciones del Consejo de Estado, 2003. págs. 25-36.

5. **Mosqueda, M. y Oliver, N.** *Diseño del Módulo de Logopedia Del Sistema de Evaluación del Neurodesarrollo en Niños.* La Habana: s.n., 2010.

6. Información de Neurociencia y Neurología. *Concepto de neurodesarrollo.* [En línea] http://neurodesarrollo.com/ .

7. **Manzano, M. e Inguanzo, G.** *Prueba de Pesquisaje del Desarrollo del Lenguaje (PPDL).* 2008.

8. **Manzano, M., Piñeiro, A. e Inguanzo, G.** *Prueba de Vocabulario Peabody (PVP).Manual de usuario.* La Habana: s.n., 2009.

9. *Inventario de las Primeras Palabras (IPP).* La Habana: s.n., 2009.

10. Metodologías de desarrollo de software. *Rup.* [En línea] [Citado el: 20 de enero de 2011.] http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html .

11. **Booch, G., Rumbaugh, J. y Jacobson I.** *El Proceso Unificado de Desarrollo de Software.* La Habana: Félix Varela, 2004. pág. 13. Vol. 1.

12. EcuRed. *Flujos de Trabajo de RUP.* [En línea] [Citado el: 25 de febrero de 2011.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo .

13. El proceso unificado de desarrollo. [En línea] [Citado el: 26 de febrero de 2011.] http://www.mitecnologico.com/Main/ProcesoDeDesarrolloUnificado .

14. **Booch, G., Rumbaugh, J. y Jacobson I.** *El Lenguaje Unificado de Modelado.* 2000. págs. 50-68.

15. UML Lenguaje unificado de Modelado. *Blog de tecnología.* [En línea] 8 de 8 de 2009. [Citado el: 2 de marzo de 2011.] http://www.gratisblog.com/index.php?itemid=130280 .

16. Visual Paradigm para UML. [En línea] [Citado el: 2 de diciembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p .

17. Lenguaje de Programación. [En línea] [Citado el: 10 de enero de 2011.] www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r32267.DOC .

18. **Álvarez, M.A.** Qué es Java. [En línea] [Citado el: 21 de enero de 2011.] http://www.desarrolloweb.com/articulos/497.php .

19. MANUAL DE JAVA. [En línea] http://www.webtaller.com/manual-java/caracteristicas-java.php .

20. JBoss- Definición. [En línea] [Citado el: 23 de noviembre de 2010.] http://www.scribd.com/doc/19026497/JBOSS .

21. *Características de JBoss.* [En línea] [Citado el: 23 de noviembre de 2010.] http://www.scribd.com/doc/51232547/23/Jboss .

22. *PostgreSQL.* [En línea] [Citado el: 25 de noviembre de 2010.] http://danielpecos.com/docs/mysql_postgres/x15.html .

23. Ambiente de Desarrollo Integrado (IDE). [En línea] [Citado el: 25 de enero de 2011.] http://foro.ignetwork.net/showthread.php?15188-IDE-Entorno-integrado-de-desarrollo-%28Concepto-importante%29 .

24. EcuRed. *Eclipse, entorno de desarrollo integrado.* [En línea] 13 de diciembre de 2010. [Citado el: 25 de enero de 2011.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado .

25. Un Lenguaje para la Especificación y Validación de Arquitecturas de Software. *Tesis doctoral.* [En línea] diciembre de 2000. http://www.lcc.uma.es/~canal/papers/tesis/index.html .

26. **Pressman.** *Ingeniería de software, un enfoque practico.* 6. s.l.: MC GRAW HILL. Pág. 992. ISBN: 970-10-5473-3.

27. Java Server Faces. *JSF.* [En línea] http://amap.cantabria.es/confluence/display/BASE/JSF .

28. **Sánchez, J. M.** Richfaces. *Introducción a RichFaces.* [En línea] [Citado el: 28 de enero de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro

29. Madeja. *Ajax4JSF.* [En línea] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Ajax4JSF

30. Cantabria. *Facelets.* [En línea] http://amap.cantabria.es/confluence/display/BASE/Facelets .

31. *Guía Breve de XHTML.* [En línea] http://www.w3c.es/divulgacion/guiasbreves/XHTML .

32. *Qué es JBoss Seam?* [En línea] [Citado el: 8 de diciembre de 2010.] http://es.debugmodeon.com/articulo/que-es-jboss-seam .

33. *Hibernate.* [En línea] [Citado el: 20 de enero de 2011.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBYQFjAA&url=http%3A%2F%2Fwww.unife.edu.pe%2Fing%2Fdesarrollo.doc&rct=j&q=Hibernate%2C%20caracteristicas&ei=g0gGTaOWE4a0sAO69bm-DQ&usg=AFQjCNEYWSGaKUVBmr9ZxZx3x-Pwfl78lQ&cad=rja .

34. *Enterprise JavaBeans (EJB3).* [En línea] [Citado el: 20 de enero de 2011.] http://sistemas.uniandes.edu.co/~isis3702/dokuwiki/lib/exe/fetch.php?media=principal:isis3702-ejb3.pdf .

35. MADEJA- JPA. [En línea] [Citado el: 21 de enero de 2011.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/JPA .

36. *Java Platform Enterprise Edition (JavaEE 5).* [En línea] [Citado el: 1 de febrero de 2011.] http://publib.boulder.ibm.com/wasce/V2.1.0/es/java-ee-5-certified.html .

37. **Visconti, M. y Astudillo, H.** Fundamentos de Ingeniería de Software. *Departamento de Informática Universidad Técnica Federico Santa María.* [En línea] [Citado el: 20 de febrero de 2011.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf .

38. Ídem a 37.

39. **Montano, C. A. y Rovira, R.** *Implementación de las hojas de anestesia estándar y cardiovascular del quirófano del Sistema de Información Hospitalaria alas HIS.* La Habana: s.n., 2010.

40. Diagramas de Interacción. [En línea] 26 de marzo de 2002. [Citado el: 5 de febrero de 2011.] http://www.mcc.unam.mx/~cursos/Objetos/Cap18/cap18.html .

41. *Ídem a 15.*

42. Fundamentos sobre el Modelo de Datos. [En línea] [Citado el: 1 de marzo de 2011.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm .

43. *El modelo dinámico y de implementación.* [En línea] [Citado el: 5 de marzo de 2011.] http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf .

44. *RUP. Las pruebas de caja negra.* [En línea] [Citado el: 20 de abril de 2011.] http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf .

45. *Características de las Pruebas de caja negra. [En línea] [Citado el: 20 de abril de 2011.]* http://www.ecured.cu/index.php/Importancia_y_objetivos_del_flujo_de_trabajo_Prueba_de_RUP.

46. **Castro, O. y Carrasco, Z.** *Implementación de los procesos de observación y administración del módulo Emergencias del Sistema de Información Hospitalaria alas HIS.* La Habana: s.n., 2010.

**BIBLIOGRAFÍA.**


1. Ajax4JSF. [En línea] [Citado el: 25 de noviembre de 2010.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Ajax4JSF.

2. Ambiente de Desarrollo Integrado (IDE). [En línea] [Citado el: 12 de diciembre de 2010.] http://www.freehttp://foro.ignetwork.net/showthread.php?15188-IDE-Entorno-integrado-de-desarrollo-%28Concepto-importante%29 downloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p/.

3. **Arevalo, M.E.** María Eugenia Arevalo's Blog- Introducción al Patrón de Arquitectura por Capas. [En línea] 3 de diciembre de 2010. [Citado el: 10 de diciembre de 2010.]
4. Atención Temprana en la Provincia de Cienfuegos. [En línea] [Citado el: 3 de octubre de 2010.] http://www.sld.cu/sitios/rehabilitacion-%20temprana/temas.php?idl=131&idv=20923.

5. **Booch, G., Rumbaugh, J. y Jacobson I.** *El Proceso Unificado de Desarrollo de Software.* La Habana: Félix Varela, 2004. Vol. 1.

6. **Booch, G., Rumbaugh, J. y Jacobson I.** *El Lenguaje Unificado de Modelado.* 2000

7. BPMN Business Process Modeling Notation. [Online] [Cited: diciembre 15, 2010.] http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf .

8. *Características de las Pruebas de caja negra. [En línea] [Citado el: 20 de abril de 2011.]* http://www.ecured.cu/index.php/Importancia_y_objetivos_del_flujo_de_trabajo_Prueba_de_RUP.

9. **Casalánguida H., D.J.**. Una Metodología Orientada a Aspectos para Desarrollo de Aplicaciones Web. Proceedings of ASSE 2007.

10. **Castro, O. y Carrasco, Z**. *Implementación de los procesos de observación y administración del módulo Emergencias del Sistema de Información Hospitalaria alas HIS.* La Habana: s.n., 2010.

11. **Conceptos Básicos de Computación**. [En línea]. [Citado el: 15 de enero del 2009.]

12. **Desongles, J.** Técnicos Auxiliares de Informática. Temario parte II. s.l.: MAD-Eduforma, 2006. ISBN 8466554068.

13. Diagramas de Interacción. [En línea] 26 de marzo de 2002. [Citado el: 5 de febrero de 2011.] http://www.mcc.unam.mx/~cursos/Objetos/Cap18/cap18.html .

14. Eclipse, entorno de desarrollo integrado – EcuRed:. [En línea] 13 de diciembre de 2010. [Citado el: 15 de diciembre de 2010.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

15. *El modelo dinámico y de implementación.* [En línea] [Citado el: 5 de marzo de 2011.] http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf .

16. Enterprise JavaBeans (EJB3). [En línea] [Citado el: 1 de diciembre de 2010.] http://sistemas.uniandes.edu.co/~isis3702/dokuwiki/lib/exe/fetch.php?media=principal:isis3702-ejb3.pdf

17. Facelets. [En línea] [Citado el: 25 de noviembre de 2010.] http://amap.cantabria.es/confluence/display/BASE/Facelets

18. **Franky ,M.C.** Java EE 5 (sucesor de J2EE):el reto de volver a empezar**.** Junio de 2007

19. Fundamentos sobre el Modelo de Datos. [En línea] [Citado el: 1 de marzo de 2011.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm .

20. **GARRETT, J. J.** AJAX un nuevo acercamiento a Aplicaciones Web, [2007]. [Citado el: 10 de febrero del 2009] Disponible en: http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php

21. **Geoffrey Sparks, Sparx Systems.** Una Introducción al UML. El Modelo de Casos de Uso. www.sparxsystems.com.ar - www.sparxsystems.cl

22. **Gómez, R. P.** Herramientas Case. [Online] [Cited: diciembre 18, 2010.] http://www.monografias.com/trabajos14/herramicase/herramicase.shtml.

23. **Grupo de Atención Temprana.** LIBRO BLANCO DE LA ATENCIÓN TEMPRANA. [En línea] [Citado el: 5 de octubre de 2010.] http://www.acondroplasiaperu.com/images/descargas/4.pdf.

24. Guía Breve de XHTML. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.w3c.es/divulgacion/guiasbreves/XHTML

25. **Håkon, L., Bert, B. Cascading Style Sheets**. Designing for the Web, Third Edition, Addison Wesley Professional.

26. **Herrera, C.** Comparativa entre EJB 3 y Spring Framework. [Online] 10 17, 2007. [Cited: diciembre 15, 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EJB3vsSpring#1.2.1.Enterprise%20JavaBeans|outline .

27. Hibernate. [En línea] [Citado el: 28 de noviembre de 2010.] http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBYQFjAA&url=http%3A%2F%2Fwww.unife.edu.pe%2Fing%2Fdesarrollo.doc&rct=j&q=Hibernate%2C%20caracteristicas&ei=g0gGTaOWE4a0sAO69bm-DQ&usg=AFQjCNEYWSGaKUVBmr9ZxZx3x-PwfI78lQ&cad=rja

28. http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/

29. http://haideeperez75.blogspot.com/feeds/posts/default/3466693163993704565

30. **IH-SW-DR-087 ALAS-HIS_Elementos comunes_Modelo de casos de uso del sistema.**

31. Información de Neurociencia y Neurología. [En línea] [Citado el: 7 de octubre de 2010.] http://neurodesarrollo.com/.

32. Java Platform Enterprise Edition (JavaEE 5). [En línea] [Citado el: 22 de noviembre de 2010.] http://publib.boulder.ibm.com/wasce/V2.1.0/es/java-ee-5-certified.html.

33. Java Server Faces. [En línea] [Citado el: 22 de noviembre de 2010.] http://amap.cantabria.es/confluence/display/BASE/JSF .

34. **JavaHispano. [EN linea] .** [Citado el: 10 de febrero del 2009] http://www.javahispano.org/contenidos.type.action?type=NEWS&menuId=NEWS

35. JBoss- Definición. [En línea] [Citado el: 8 de diciembre de 2010.] http://www.scribd.com/doc/19026497/JBOSS

36. **Kiran, G., Michael, L. y Bruce, W.** Introducción a BPM para Dummies.

37. **LABRA, J. E. Curso:** Tecnología Web, [2007]. [Citado el: 12 de febrero del 2009] Disponible en:http://www.di.uniovi.es/~labra/cursos/masterMTI/PDF/ServWeb.pdf

38. **LARGMAN, C. UML Y PATRONES,** Introducción al análisis y diseño orientado a objetos.

39. Lenguajes de Programación. [Online] 2009. [Cited: diciembre 16, 2010.] http://www.lenguajes-de-programacion.com/programacion-java.shtml .

40. **León, E.** Tutorial Visual Paradigm for UML. [Online] [Cited: diciembre 15, 2010.] http://www.scribd.com/doc/36636137/Tutorial-Visual-Paradigm .

41. MADEJA- JPA. [En línea] [Citado el: 1 de diciembre de 2010.] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/JPA

42. **Madrid, V. J.** Introducción a BPMN. [En línea] [Citado el: 5 de diciembre de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=bpmn

43. MANUAL DE JAVA - CARACTERISTICAS DE JAVA . [En línea] [Citado el: 5 de diciembre de 2010.] http://www.webtaller.com/manual-java/caracteristicas-java.php

44. **MARIN D. M. E.** Fundamentos del Sistema de Salud Pública en Cuba para estudiantes de Informática. La Habana. 2006.

45. **Montano, C.A. y Rovira, R.** *Implementación de las hojas de anestesia estándar y cardiovascular del quirófano del Sistema de Información Hospitalaria alas HIS.* La Habana: s.n., 2010.

46. **Mulas, F.** LA ATENCIÓN TEMPRANA: QUÉ ES Y PARA QUÉ SIRVE. [En línea] [Citado el: 5 de octubre de 2010.] http://www.invanep.com/descargas/documentos/atencion/atencion.temprana.mulas.milla.pdf .

47. MVC - Modelo Vista Controlador.[En línea] [Citado el: 16 de diciembre de 2010.] http://www.programacionweb.net/articulos/articulo/?num=505.

48. **Pérez, M.C.** La intervención logopédica a través de las nuevas tecnologías. [En línea] [Citado el: 3 de octubre de 2010.] http://www.cedown.org/articulos/logopedia/nuevas-tecnologias.htm.

49. **Plataforma De Aplicaciones Jboss Enterprise.** [Citado el: 10 de marzo del 2009] www.redhat.es/jboss

50. **Pons, C. y otros.** Introducción a las Bases de Datos. El Modelo Relacional. s.l. : Thomson Learning Ibero, 2005. ISBN 8497323963.

51. PostgreSQL. [En línea] [Citado el: 10 de diciembre de 2010.] http://danielpecos.com/docs/mysql_postgres/x15.html

52. Qué es JBoss Seam?. [En línea] [Citado el: 26 de noviembre de 2010.] http://es.debugmodeon.com/articulo/que-es-jboss-seam

53. reEduca.com. [En línea] 9 de 9 de 2009. [Citado el: 5 de octubre de 2010.] http://www.reeduca.com/estimulacion-temprana.aspx?cat=201.

54. *RUP. Las pruebas de caja negra.* [En línea] [Citado el: 20 de abril de 2011.] http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf

55. **Sánchez, J.** Monografias.com. [En línea] [Citado el: 1 de noviembre de 2010.] http://www.monografias.com/trabajos75/formacion-equipo-%20interdisciplinario/formacion-equipo-interdisciplinario.shtml.

56. **Sánchez, J.M.** Introducción a RichFaces. [En línea] [Citado el: 22 de noviembre de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro.

57. Un Lenguaje para la Especificación y Validación de Arquitecturas de Software. [En línea] diciembre de 2010. [Citado el: 15 de diciembre de 2010.] http://www.lcc.uma.es/~canal/papers/tesis/index.html.

58. **Visconti, M. y Astudillo, H.** Fundamentos de Ingeniería de Software. *Departamento de Informática Universidad Técnica Federico Santa María.* [En línea] [Citado el: 20 de febrero de 2011.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf.

59. Visual Paradigm para UML. [En línea] [Citado el: 10 de diciembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/

**GLOSARIO DE TÉRMINOS.**

**Actividad:** es una unidad de trabajo que una persona que desempeñe un rol puede ser solicitado a que realice. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

**Ajax:** es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

**API:** conjunto de funciones residentes en bibliotecas (generalmente dinámicas), que permiten que una aplicación corra bajo un determinado sistema operativo.

**Artefacto:** elementos de información producidos, modificados, o usados por el proceso. Son los productos tangibles del proyecto y son usados por los trabajadores, para realizar nuevas actividades.

**Caché:** es un sistema especial de almacenamiento de alta velocidad. Puede ser tanto un área reservada de la memoria principal, como un dispositivo de almacenamiento de alta velocidad independiente.

**Case sensitive:** es una expresión usada en informática que se aplica a los textos en los que tiene alguna relevancia, escribir un caracter en mayúscula o minúscula

**Clase:** es una construcción que se utiliza como un modelo para crear objetos de ese tipo. El modelo describe el estado y el comportamiento que todos los objetos de la clase comparten.

**Compilador:** es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.

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

**Edad cronológica:** se determina como resultado de restar la fecha de aplicación de test y la fecha de nacimiento del evaluado.

**Edad de vocabulario:** se determina a través de la puntuación obtenida en la realización de un test.

**Equipo interdisciplinario:** Puede definirse como un grupo de personas, desde una amplia gama de disciplinas, que trabajan juntas para asegurar la utilización integrada de las ciencias naturales y sociales y las artes en la planificación y la toma de decisiones para resolver un mismo problema. El equipo interdisciplinario, como todo grupo, se instaura desde la propuesta de una tarea común, la que dará cuerpo y determinará dominantemente las formas de su organización de la tarea planteada.

**Examen logofoniátrico:** es el examen que se realiza cuando el niño llega por primera vez a la consulta, evalúa el estado tanto físico, como clínico de los niños cuyas edades están comprendidas entre cero y cinco años.

**Firma digital:** esquema matemático que sirve para demostrar la autenticidad de un mensaje digital o de un documento electrónico. Una firma digital brinda al destinatario seguridad de que el mensaje fue creado por el remitente, y que no fue alterado durante la transmisión. Se utiliza comúnmente para la distribución de software, transacciones financieras y en otras áreas donde es importante detectar la falsificación y la manipulación.

**Flujo de trabajo:** es una relación de actividades que producen resultados observables.

**Framework:** estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

**Health Level Seven:** conjunto de estándares para el intercambio electrónico de información clínica. Proveer los mejores estándares globales para los dominios: clínico, asistencial, administrativo y logístico, con el fin de lograr una interoperabilidad real entre los distintos sistemas de información en el área de la salud.

**Herramientas CASE:** son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas, pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas

como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación, o detección de errores entre otras.

**Indentación:** de uso común en informática, que significa mover un bloque de texto hacia la derecha, insertando espacios o tabuladores para separarlo del texto adyacente. Se utiliza para mejorar la legibilidad del código fuente por parte de los programadores, teniendo en cuenta que los compiladores o intérpretes, raramente consideran los espacios en blanco entre las sentencias de un programa.

**Java Community Process:** comunidad de desarrollo de herramientas y aplicaciones para Java.

**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.

**Logopedia:** es la ciencia que estudia, previene, evalúa, diagnostica y trata los problemas de la comunicación en cualquiera de sus campos. Abarca la voz, el habla, el lenguaje, la escritura, la expresión corporal y la expresión facial.

**Máquina virtual:** es un software que emula a una computadora y puede ejecutar programas como si fuese una computadora real. Este software en un principio fue definido como: un duplicado eficiente y aislado de una máquina física.

**Metadata:** es la información sobre los datos que se alimenta, se transforma y existe en el data warehouse. Es un concepto genérico, pero cada implementación de la metadata usa técnicas y métodos específicos.

**Middleware:** software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores. El middleware, abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas.

**Modelo relacional:** es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

**Nomenclador:** conjunto de palabras técnicas propias de una ciencia. Término utilizado para nombrar los objetos que conforman la configuración del módulo.

**No Conformidad:** término empleado en la realización de las pruebas al software, se refiere a un fallo detectado en el sistema informático.

**Notación CamelCasing:** es común en Java, parecido al PascalCasing con la excepción de que la letra inicial del identificador debe estar en minúscula.

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

**Objeto:** se define como la unidad que en tiempo de ejecución, realiza las tareas de un programa. A un nivel más básico se define como la instancia de una clase.

**Open source:** término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código, que a las cuestiones morales y filosóficas, las cuales destacan en el llamado software libre.

**ORM:** 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 el utilizado en una base de datos, creando una base de datos orientada a objetos virtuales, por encima de la base de datos relacional.

**Orientado a objetos:** es una forma especial de programar, más cercana de cómo se expresan las cosas en la vida real que otros tipos de programación.

**Patrón Singleton:** está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella.

**POJO:** lenguaje de programación Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.

**Release:** Versión de una aplicación informática.

**Requerimientos Funcionales**: definen las funciones que el sistema será capaz de realizar. Describen además, las transformaciones que el sistema realiza sobre las entradas para producir salidas.

**Rol:** define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos, trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol, puede ser representado por varias personas.

**Script:** es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. El uso habitual de los scripts es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo, o con el usuario.

**SOA:** es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

**TCP/IP:** es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo: PC, minicomputadoras y computadoras centrales sobre redes de área local y área extensa.

**Usabilidad:** se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

**Web 2.0:** es la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través de la web y que están enfocadas al usuario final. Se trata de aplicaciones que generen colaboración y de servicios, que reemplacen las aplicaciones de escritorio.

**WSDL:** formato XML que permite tener una descripción de un servicio web. Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y los detalles de cómo se debe utilizar.