UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Módulo para el registro de los pacientes de Nefrología

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

Autor: Javier Gómez Montero

Tutores:

Msc.Máxora Rorayma Castro Pérez

Ing. Rubier Sorio Pazos

Ing. Karina Bárbara Martinez Casas

DECLARACIÓN DE AUTORÍA:

Declaro ser autor de la presente tesis y reconozco a la XETID, Empresa de Tecnologías de la Información para la Defensa, los derechos patrimoniales de la misma, con carácter exclusivo.

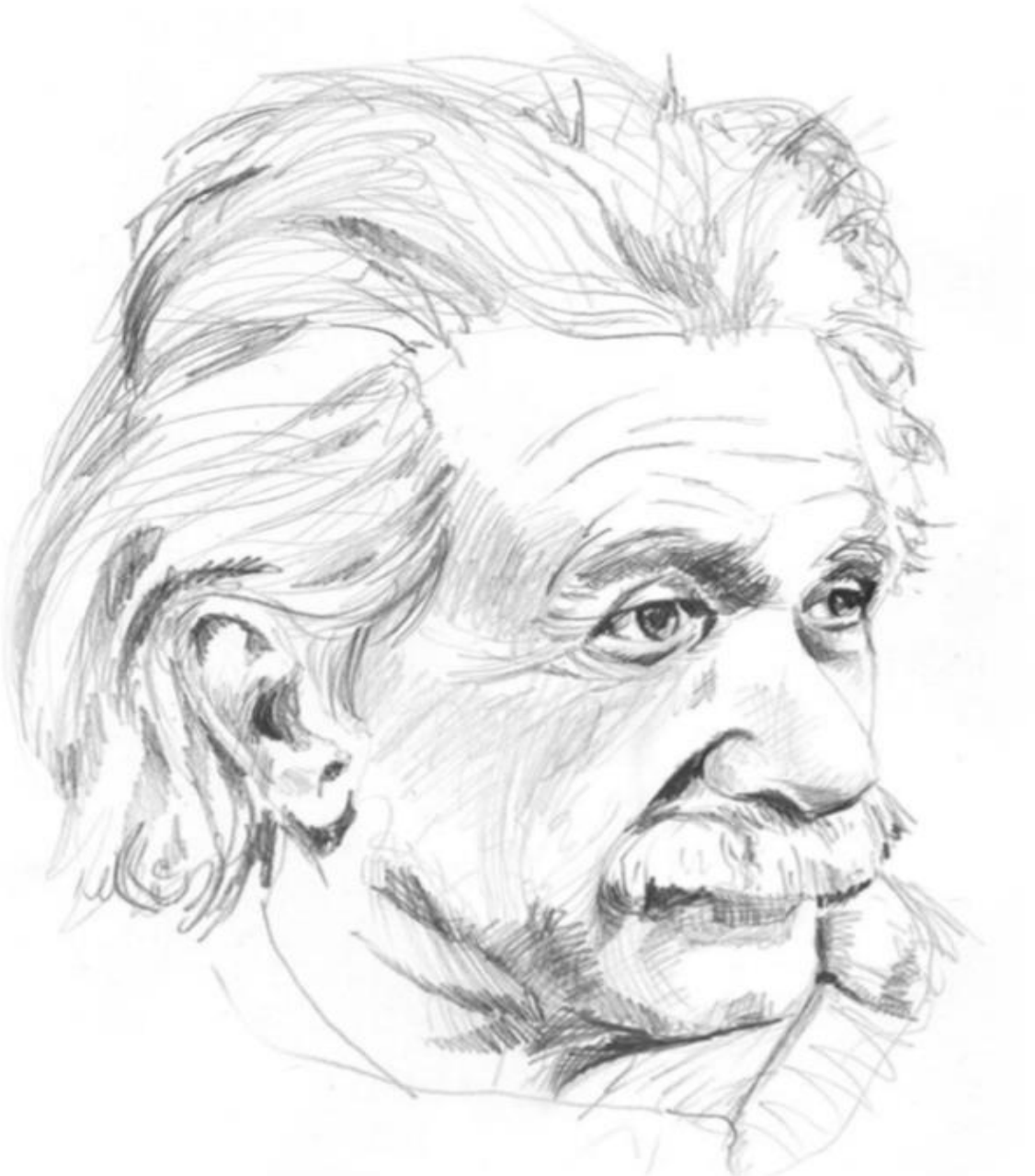
Para que así conste firmo la presente a los __ días del mes _____ del año _____.

Javier Gómez Montero

_____

Firma de Autor

Msc. Máxora Rorayma Castro Pérez	Ing. Rubier Sorio Pazos	Ing. Karina Bárbara Martinez Casas
_____	_____	_____
Firma de Tutor	Firma de Tutor	Firma de Tutor



"La vida es una preparación para el futuro; y la mejor preparación para el futuro es vivir como si no hubiera ninguno"

Albert Einstein

RESUMEN

El registro de pacientes de nefrología es imprescindible para mantener un control de los datos de los ciudadanos que padecen de enfermedades renales en Cuba. El registro de los pacientes se realiza actualmente en formato duro, manualmente y pertenece exclusivamente al hospital que brinda el servicio, si el paciente tuviera la necesidad de acudir a otro centro de salud, el médico no tendría acceso a estos registros que contribuirían a su correcta atención. Además como la información es guardada en formato duro, corre riesgo de deterioro y se dificulta su búsqueda. Partiendo de esta situación problemática se diseñó el módulo para la gestión de pacientes de nefrología SINEF, con el cual se gestionará la información individual de cada uno de los pacientes, facilitando la disponibilidad de la información. Para ello se llevó a cabo una recolección de los requerimientos y necesidades del cliente a través de encuentros directos con el mismo. Para el desarrollo de la solución se utilizó Visual Paradigm como herramienta para el modelado de los procesos pertenecientes a la ingeniería, UML como lenguaje de modelado, PostgreSQL como gestor de base de datos, PgAdmin III como herramienta para la administración de base de datos, Netbeans como entorno de desarrollo integrado, Zeolides como marco de trabajo de desarrollo, AngularJs y Zendframework. Su funcionamiento fue evaluado mediante la aplicación de pruebas, proceso que culminó con la aceptación por parte del cliente. Se obtuvo como resultado el completo desarrollo de la aplicación.

Palabras clave: nefrología, paciente, software informático, SINEF (Sistema Informático para Nefrología)

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El auge de la informática y su uso en la vida cotidiana, ha provocado su extrapolación hacia otros sectores como el sector de la salud, incrementando la eficiencia del mismo con equipamientos y métodos innovadores que van desde la fabricación de poderosos medicamentos, hasta la monitorización de los signos vitales de un paciente mientras es intervenido quirúrgicamente. Con los adelantos tecnológicos que se han puesto al servicio del sector de la salud, los médicos cuentan con una mayor cantidad de información, lo que facilita el diagnóstico sobre la enfermedad que padece determinado paciente.

La Empresa de Tecnologías de la Información para la Defensa (XETID), cuya actividad central es contribuir a la informatización de la sociedad cubana, cuenta con el Centro de Integración, que está actualmente está involucrado en el desarrollo de Sistema Informático para Nefrología (SINEF), un sistema que gestiona la información de los pacientes de nefrología en el Ministerio de Salud Pública de Cuba (MINSAP).

El Sistema Informático para Nefrología (SINEF), es un sistema de alcance nacional de Gestión de Pacientes por modalidades de tratamiento en la especialidad de Nefrología en Cuba. Este sistema es el encargado de la fiscalización y consulta de esta actividad que incluye el registro en tiempo real del 100% de los ciudadanos con insuficiencia renal crónica, la lista de espera de trasplante de riñón, el registro de Enfermedad Renal Crónica (ERC).

Los formularios de trabajos médicos desempeñan un rol muy importante, gracias a los cuales se obtienen las bases de conocimiento para estudios en el sector de la salud, que permiten apoyar al personal de dicho sector en la toma de decisiones, así como en las actividades docentes, investigativas y extra sectoriales. Además, permiten establecer líneas de conducta para la reducción de la mortalidad, facilitan los trabajos de los grupos de riesgos en prevención primaria, posibilitan las comparaciones de los datos de los pacientes con períodos anteriores y con los de otros pacientes de otras regiones dentro o fuera del país.

Muchos de los datos sobre el paciente son obtenidos una vez que se realiza su registro en una institución de salud. El registro de los pacientes es una de las principales tareas y procesos a realizar por los trabajadores de la salud, y se debe prestar atención en cuanto a la fiabilidad de la información y el llenado de las historias clínicas.

El registro de los pacientes de nefrología, en estos momentos, se realiza en formato duro, manualmente y pertenece exclusivamente al hospital que brinda el servicio, lo cual provoca falta de disponibilidad de la información, pues si el paciente tuviera la necesidad de atenderse en otro centro de salud, el médico no tendría acceso a estos registros que contribuirían a su correcta atención. Además como la información se encuentra guardada en papel, corre más riesgo de deterioro y se dificulta su búsqueda.

La situación planteada anteriormente permitió identificar el siguiente **problema de investigación**: ¿Cómo contribuir al registro y disponibilidad de la información de los pacientes de nefrología en los hospitales del Ministerio de Salud Pública de Cuba? Se definió como **objeto de estudio**: procesos de recolección de datos médicos, delimitándose como **campo de acción**: procesos para la recolección de datos médicos sobre pacientes de nefrología. **Objetivo general:** Desarrollar un módulo el registro de los pacientes de nefrología en los hospitales del Ministerio de Salud Pública del sistema de salud cubano que contribuya a reducir su pérdida y deterioro.

Objetivos específicos:

- Investigar los fundamentos teóricos y metodológicos referidos al registro de los pacientes de nefrología.
- Diseñar el módulo para el registro de los pacientes de nefrología.
- Implementar el módulo para el registro de los pacientes de nefrología.
- Evaluar el correcto funcionamiento de la solución.

Con la meta de cumplir con los objetivos específicos planteados se establecen las siguientes **tareas de investigación**:

- Definición de los elementos teóricos relacionados con el registro de los pacientes de nefrología.
- Análisis de diferentes soluciones informáticas que permiten el registro de los pacientes de nefrología.
- Caracterización de la metodología y las tecnologías utilizadas para el desarrollo de la solución.
- Identificación de los requisitos de la solución a desarrollar.
- Diseño del módulo para el registro de los pacientes de nefrología.

- Codificación del módulo para el registro de los pacientes de nefrología.
- Realización de las pruebas necesarias para identificar posibles errores en la solución.

Los **métodos de investigación** utilizados para dar solución a la presente investigación son:

Teóricos:

- Histórico-Lógico: Posibilitó realizar un estudio de los principales conceptos relacionados con el registro de los pacientes desde su surgimiento hasta la actualidad, así como de algunos de los diferentes sistemas informáticos para nefrología existentes hasta la fecha.
- Analítico-Sintético: Permitió llevar a cabo un análisis bibliográfico para establecer las bases teóricas en relación al desarrollo de los sistemas informáticos para nefrología.
- Modelación: Facilitó la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.

Empíricos:

- Entrevista: Se realizó a través de encuentros con el cliente para conocer la necesidad del desarrollo de la propuesta de solución y definir sus requisitos funcionales.

El contenido del presente trabajo, se estructuró en los siguientes tres capítulos:

Capítulo 1. Fundamentación teórica sobre la gestión de información del registro de los pacientes de nefrología: expone los elementos teóricos de la investigación. Se realiza el estudio e investigación de soluciones informáticas existentes para la gestión y formalización del registro de los pacientes de nefrología. Se determinan la metodología de desarrollo, herramientas y tecnologías que se utilizan para desarrollar la solución propuesta.

Capítulo 2. Análisis y diseño del módulo para Nefrología: contendrá toda la información referente al análisis y diseño de la aplicación, contando en sí con los diagramas de clases, requisitos funcionales y no funcionales que debe cumplir, así como propuesta y diseño de la arquitectura del sistema para lograr una mayor aceptación por parte de los usuarios.

Capítulo 3. Implementación, pruebas y validación del módulo para el registro de los pacientes de nefrología: describirá el proceso de implementación y prueba de la solución informática para módulo para el registro de los pacientes de nefrología. Contará con el diagrama de despliegue, así como la descripción de algunos casos de prueba.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace una breve descripción de los conceptos fundamentales necesario para la comprensión del presente trabajo de tesis, así como un estudio de las metodologías y herramientas utilizadas para el registro de los pacientes de nefrología. También se hace una caracterización de las herramientas y metodología de software empleadas en el desarrollo del módulo. Se aclara que es una decisión del cliente que el sistema se implemente desde cero y que tanto las herramientas como las metodologías son las establecidas por el centro XETID al que está vinculada esta investigación, por lo que no se valoran otras alternativas.

1.2 Conceptos Fundamentales

Registro de pacientes

Un r**egistro de pacientes** es un fichero, documento o conjunto organizado de datos de salud de personas que padecen una enfermedad, ligada a una identificación personal de las mismas. En su formato más sencillo puede consistir en una colección de fichas de papel recogidas por un médico y guardadas en una caja. (Barcelona, 2018)

Enfermedad Renal Crónica (ERC)

La enfermedad renal crónica (ERC) o insuficiencia renal crónica (IRC) es una pérdida progresiva (por tres meses o más) e irreversible de las funciones renales, cuyo grado de afección se determina con un filtrado glomerular. Como consecuencia, los riñones pierden su capacidad para eliminar desechos, concentrar la orina y conservar los electrolitos en la sangre.

Los síntomas de un deterioro de la función renal son inespecíficos y pueden incluir una sensación de malestar general y una reducción del apetito. A menudo, la enfermedad renal crónica se diagnostica como resultado del estudio en personas en las que se sabe que están en riesgo de problemas renales, tales como aquellos con presión arterial alta o diabetes y aquellos con parientes con enfermedad renal crónica. La insuficiencia renal crónica también puede ser identificada cuando conduce a una de sus reconocidas complicaciones, como las enfermedades cardiovasculares, anemia o pericarditis. (Rockville Pike, 2018)

Categoría Enfermedad Renal Aguda

Es una ERC, en una etapa que todavía puede ser reversible. Cuando los riñones dejan de funcionar de repente, durante un período de tiempo muy corto (generalmente dos días o menos), se denomina como lesión renal aguda (LRA). La LRA a veces se refiere como falla del riñón aguda o falla renal aguda. Es muy grave y requiere tratamiento inmediato.

Categoría ERC-5 (o IRT)

Dígase de aquella Enfermedad Renal Crónica que ha alcanzado la quinta etapa. Insuficiencia renal terminal (IRT), o insuficiencia renal, es la etapa final de la función renal. También llamada etapa 5 de la ERC, la IRT indica que ninguno de los dos riñones ya trabaja lo suficiente como para mantener el cuerpo saludable y químicamente equilibrado.

Categoría Trasplante Renal

Es cuando se decide resolver la afección del paciente mediante un trasplante de riñón. El riñón que se implanta se coloca en la parte lateral del abdomen, por debajo del ombligo. Puede colocarse, de forma indiferente, en el lado izquierdo o derecho. Por lo tanto la cirugía se realiza por la parte delantera, quedando la cicatriz quirúrgica en esta zona.

1.3 Aplicaciones web que gestionan información de salud

Antiguamente el proceso de gestión de los datos de los pacientes se almacenaba en formato duro, y para ello, se realizaba a mano. Esto provocaba problemas de fiabilidad y disponibilidad de la información. Con el avance de las tecnologías se comienza a digitalizar el proceso antes mencionado, así surgen sistemas de gestión de gestión de datos para la información de los pacientes, introduciéndose el concepto de **HIS** que son los sistemas de gestión hospitalaria por sus siglas en inglés. A continuación se realiza un estudio de algunos de estos sistemas teniendo en cuenta: aceptación del cliente, licencia de pago, si son de código abierto, gestionan la información y datos de los pacientes, son aplicaciones web.

Aquar Software Nefrología

Un *software* que ofrece la gestión informática, para todos los procesos de negocio, desde la gestión de pacientes, citas, informes, consentimientos, cuestionarios de salud, gestión económica hasta todo lo que se necesite. Es adaptable, lo cual facilita el trabajo de facultativos, doctores, asociados, personal de atención al paciente / cliente, recepcionistas, auxiliares de clínica. Incluye desde las funciones más

básicas a las más complejas: Gestión de citas, clientes, datos de filiación, gestión de recepción y sala de espera, hoja de trabajo para los profesionales, gestión económica privada y de sociedades, (ventas, compras, almacenes, laboratorios, stock) y una Completa Historia Clínica dividida en: Anamnesis, Episodios / Diagnósticos y Seguimientos. Sin embargo hay varios factores que lo descartan como candidato, es un software con licencia de pago y esto supone un freno para la mentalidad de soberanía tecnológica del país. Es un sistema que se basa en Windows/Mac Os, esto provoca desconfianza al no poder analizar el código fuente de la misma. (Software, 2018)

Nexadia Expert

El sistema Nexadia es una solución informática renal a la mayoría de los requerimientos de gestión de los datos de tratamiento de diálisis y su interconexión con el resto del entorno sanitario. Da respuesta al incremento de los datos monitorizados durante el tratamiento renal sustitutivo liberando a los equipos de enfermería del trabajo administrativo al que se ven sometidos para que puedan invertir su tiempo en lo realmente importante: el cuidado del paciente.

Está formado por dos programas individuales que se complementan a la perfección: el programa de monitorización del tratamiento Nexadia®monitor, y el sistema de gestión de datos de diálisis Nexadia®expert.

Nexadia®expert es un Sistema de Gestión de Datos de Paciente para la gestión clínica de alto nivel. Con este programa se pueden manejar y archivar todos los datos de paciente y su terapia generados en una Unidad de Hemodiálisis:

Manejo de datos clínicos y datos de paciente (HD y PD). Prescripciones médicas individualizadas. Comunicación a sistemas médicos externos como laboratorios, Sistemas Hospitalarios de Información (HIS), etc. Evaluación gráfica de datos. Personalización del entorno de programa: definición de campos. Visualización sencilla de toda la información médica. (Braun, 2018)

Este sistema no cumple con las características que se busca pues es de licencia de pago, no es de código abierto, además del gasto que supondría su implantación, la cual según estudios de la universidad de California encontró que en promedio los hospitales están gastando 544,000 dólares por implantar el sistema y 78,500. (SineMed System, S.A., 2016) (B Braun, 2017).

Sistema Médico en Línea:

Es una aplicación web pensada y creada para médicos de todas las especialidades que requieran llevar el control de consulta de sus pacientes, citas e informes médicos en una forma rápida, sencilla y eficiente. El Sistema Médico en Línea (SML) cuenta con las siguientes ventajas:

- ✓ **Movilidad y Accesibilidad:** Podrá acceder a la información sin restricciones de horario y desde cualquier sitio.
- ✓ **Fácil Implementación:** Con el SML sólo se necesita registrar y solicitar un plan que se adapte al consultorio.
- ✓ **Seguro y Confiable:** Todos los datos serán respaldados diariamente en los servidores dedicados.(SML, 2018)

Este Sistema no fue aceptado por el cliente, existe la imposibilidad de analizar el código fuente.

MediCloud:

MediCloud.me es una plataforma gratuita, diseñada para el fácil almacenamiento en la nube de información médica de pacientes de diferentes profesionales de la salud. Dado que la información se almacena en la nube, la misma puede ser consultada desde cualquier dispositivo con internet, en cualquier momento y en cualquier parte del mundo. Al mismo tiempo, la plataforma ofrece a los usuarios la facilidad de administrar sus clínicas de forma digital, conectar con diferentes proveedores para automatizar procesos de servicio y pertenecer a una comunidad científica latinoamericana (Medicloud, 2018).

La infraestructura usada para la implementación de este sistema no es la ideal, no fue aceptado por el cliente MINSAP.

En Cuba también se ha indagado en el tema de aplicaciones que gestionen datos de pacientes, algunas de la cuales se describen a continuación.

Alas Nefrored:

La propuesta perfecciona el espacio informativo en la red que difunde el trabajo de la Sociedad Cubana de Nefrología y ofrece un atractivo sistema de gestión de contenidos que permite interactuar con el usuario. A través de esta plataforma, tendrán acceso a recursos de información que les posibilitarán actualizarse e intercambiar sobre el quehacer científico en esta área. La amplitud que abarcan sus contenidos se muestra en secciones como las de Actualidad, Recursos de Información y Apuntes históricos. En tanto,

Espacio público servirá para el intercambio de preguntas frecuentes e información sobre las principales dudas de pacientes, cuidadores y población en general, mientras que en la sección de convocatorias conocerán todos los eventos y cursos previstos. También está disponible la relación de instituciones, centros y especialistas que atienden los servicios del Programa Nacional de la especialidad en el país. Otras secciones del sitio son: Noticias Al Día, Enlaces de Interés, Fechas señaladas, Recursos de información y NefroRed, que gestiona la información generada en los servicios de nefrología, diálisis y trasplantes del territorio dirigidos al Centro Coordinador del Programa de Enfermedad Renal Crónica. (Infomed, 2018)

En el año 2003-04 se comienza con la tarea de crear en software Alas Nefrored por parte del departamento CESIM o Centro de Informática Médica, dicha petición fue realizada por el MINSAP. Dicho sistema se culmina en el año 2015 y se entrega al cliente, el cual rechaza la propuesta de solución.

A partir del análisis realizado se llega a la conclusión de que, independientemente de la petición del cliente de que se implemente una solución desde el principio, los sistemas analizados aportan conocimientos necesarios que son de interés para el MINSAP tales como la forma de gestionar los pacientes y la cantidad de información que se muestra, siendo *Aquar Software* el mejor exponente de esta última característica. En cuanto a la forma de mostrar el listado de pacientes, Nexadia Expert fue el principal aportador. Se tomaron como base para el resto de la aplicación los restantes Sistemas.

Tabla 1. Análisis de las funcionalidades de las herramientas estudiadas. (Elaboración propia)

Criterios / Sistemas	Código abierto	Gestionan la información y datos de los pacientes	Son aplicaciones web	Son de licencia de pago
Aquar Software Nefrología	No	Sí	Sí	Sí
Nexadia Expert	No	Sí	Sí	Sí
Sistema Médico en Línea	No	Sí	Sí	No
MediCloud	No	Sí	Sí	No
Alas Nefrored (UCI)	Si	Sí	Sí	No

1.4 Proceso de desarrollo de software

Un proceso de desarrollo de software en un conjunto de actividades que se llevan a cabo para garantizar hasta cierto punto la calidad del mismo. Dicho proceso cuenta siempre con varias fases las cuales varían en dependencia de la metodología escogida.

La Empresa de Tecnologías de la Información para la Defensa (XETID) utiliza la metodología de desarrollo propuesta en el manual Prodesoft en su versión 1.5, la cual expresa los puntos necesarios para el desarrollo del ciclo evolutivo de un software.

El ciclo de vida compuesto por 5 fases: **inicio**, **modelación**, **construcción**, **explotación experimental** y **despliegue**, que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo producto de software, hasta aquel en que el producto deja definitivamente de ser utilizado por el último de sus usuarios (Pedesoft, 2018).

Durante la **fase de Inicio** se logra una visión preliminar de la problemática a resolver y se definen los recursos relevantes para la ejecución del proyecto. Es decir, se describen los objetivos y el alcance del proyecto, se identifican los involucrados y ejecutores (entidades involucradas), se estima de manera general las actividades a realizar durante todo el ciclo de desarrollo del proyecto (Cronograma General), se establece la estrategia a seguir para realizar la modelación del negocio, la captura de requisitos y de ser necesario se estiman los recursos materiales que deberán ser adquiridos (Pedesoft, 2018)

En la **fase de Modelación** se capturan las partes esenciales del sistema, donde se identifican los procesos de negocio fundamentales y se aceptan los requisitos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación aprobada por los implicados en el proyecto. El hito fundamental de esta fase es la liberación de la arquitectura de sistema, datos y despliegue (Pedesoft, 2018).

En la **fase de Construcción** se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. Las fases anteriores sólo dieron una arquitectura básica que es aquí refinada de manera incremental conforme se construye el producto. En esta fase todas las características, componentes, y requisitos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto (Pedesoft, 2018).

Durante la **fase de Explotación Experimental** se convierte la versión liberada del producto en una solución estable, donde se eliminan los errores que surgen durante las pruebas y se obtiene una certificación funcional y de seguridad del producto (Pedesoft, 2018)

En la **fase de Despliegue** se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, culminando de ser preciso con transferencias tecnológicas (Pedesoft, 2018).

1.5 Lenguajes de modelado

UML versión 8

El Lenguaje Unificado de Modelado (UML) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan (UML, 2017).

Sus funciones son resumibles en cuatro fundamentales:

Visualizar: expresa de una forma gráfica un sistema de forma que otro lo puede entender.

Especificar: especifica cuáles son las características de un sistema antes de su construcción.

Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.

Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

El Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML), (14), permitirá:

- ✓ Modelar utilizando técnicas orientadas a objetos.
- ✓ Especificar todas las decisiones de análisis, diseño e implementación.
- ✓ Documentar todos los artefactos de un proceso de desarrollo.

Se utiliza para desarrollar, mantener, evaluar servicios y sistema de software que satisfagan todos los requisitos del usuario. Valorar todas las necesidades del cliente y especificar los requisitos de software

Con el Lenguaje Unificado de Modelado se realizaron los diagramas siguientes:

- ✓ Diagrama de Clases
- ✓ Diagrama de Componentes

- ✓ Diagrama de Despliegue

1.6 Herramientas de modelado

Las herramientas de modelado de sistemas informáticos permiten crear un "simulacro" del sistema, a bajo costo y riesgo mínimo. A bajo costo porque es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real. Además, minimizan los riesgos, porque los cambios que se deban realizar, se pueden realizar más fácil y rápidamente sobre el modelo que sobre el sistema ya implementado (Herramienta de Modelado, 2017).

Visual Paradigm v15.1

Visual Paradigm es una herramienta *CASE*: Ingeniería de *Software* Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Visual Paradigm, 2017)

Esta herramienta de modelado fue utilizada para la confección de los diagramas de actividades, los diagramas de proceso de negocio, los diagramas conceptuales, los diagramas de modelado de base de datos, los diagramas de requerimientos y los prototipos de interfaz de usuarios.

1.7 Lenguajes de programación

PHP v5.4

"*Hypertext Preprocessor*" es un lenguaje interpretado de alto nivel embebido en páginas *HTML* y ejecutado en el servidor. El cliente solamente recibe el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los ficheros con *PHP.* (…) *PHP* también soporta el uso de otros servicios que usen protocolos como *IMAP*, *SNMP*, *NNTP*, *POP3*, *HTTP* y derivados (AULBACH, WINSTEAD, TORBEN WILSON, LERDORF, ZMIEVSKI, & AHTO)

HTML v5

HTML5 es la última versión de *HTML*. El término representa dos conceptos diferentes:

- Se trata de una nueva versión de *HTML*, con nuevos elementos, atributos y comportamientos.

- Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance (HTML 5, 2017)

JavaScript v1.8

JavaScript es el lenguaje interpretado orientado a objetos desarrollado por *Netscape* que se utiliza en millones de páginas *web* y aplicaciones de servidor en todo el mundo. *JavaScript* puede funcionar como lenguaje procedimental y como lenguaje orientado a objetos (HTML 5, 2017).

Estos lenguajes de programación son utilizados para funcionalidades específicas, el lenguaje *HTML* 5 es utilizado para la confección visual de la aplicación, se escoge este para que el sistema tenga un mejor rendimiento en el procesamiento, *JavaScript* es utilizado para la verificación de las validaciones de los requisitos funcionales y PHP 5 es aplicado para realizar el desarrollo de la aplicación.

AngularJs v 1.63

AngularJS, es un *framework* de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. (Technology, 2010)

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos.

AngularJS Material

AngularJS Material es un marco de componentes de la Interfaz de Usuario (UI) y una implementación de referencia de la especificación de diseño de materiales de Google. Este proyecto proporciona un conjunto de componentes UI reutilizables, bien probados y accesibles basados en el diseño de materiales.

1.8 Sistemas gestores de base de datos

PostgreSQL v9.4

PostgreSQL es un sistema de gestión de bases de datos de código abierto que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre apoyada por organizaciones comerciales. Dicha comunidad es denominada el *PGDG* (*PostgreSQL Global Development Group*) (Gestores de BD, 2018) (PostgreSQL, 2018). Todo el proceso del manejo de las bases de datos fue hecho con esta herramienta por la gran comodidad que brinda y su gran aceptación además de ser el estándar en el departamento Integración de XETID.

1.9 Administrador de base de datos

PgAdmin III v1.22.2

Herramienta de código abierto para la administración de bases de datos *PostgreSQL*. Diseñada para responder a las necesidades de los usuarios, desde escribir consultas *SQL* simples hasta desarrollar bases de datos complejas. La aplicación incluye un editor *SQL* con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar comandos programados y soporte para la replicación. La conexión al servidor puede hacerse mediante la familia de protocolos de internet (*TCP/IP* por sus siglas en inglés) y encriptarse mediante el protocolo de Capa de Conexión Segura (*SSL* por sus siglas en inglés) para mayor seguridad (PGADMIN, 2017)

1.10 Entorno Integrado de Desarrollo (I.D.E.)

NetBeans v8.2

NetBeans IDE es un entorno de desarrollo: una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en *Java* - pero puede servir para otro lenguaje de programación. Existe además un número importante de módulos para extender el *NetBeans IDE*. *NetBeans IDE* es un producto libre y gratuito sin restricciones de uso (Netbeans, 2018).

1.11 Marco de trabajo

Un marco de trabajo es un conjunto de conceptos, prácticas, herramientas, librerías y criterios para enfocar una problemática particular, para enfrentar y resolver nuevos problemas de índole similar.

Ejemplos de marcos de trabajo utilizados actualmente u ofrecidos por organismos o empresas de normalización son:

- *Resource Description Framework*, un conjunto de reglas del *World Wide Web Consortium* para describir cualquier recurso de internet, como un sitio *Web* y su contenido.
- *Internet Business Framework*, un grupo de programas que forman la base tecnológica para el producto *mySAP* de *SAP*, la compañía alemana que comercializa una línea de productos de gestión de recursos empresariales.
- *Sender Policy Framework*, un enfoque definido y una programación para hacer más seguro el correo electrónico.
- *Zachman framework*, una estructura lógica destinada a proporcionar una representación integral de una empresa de tecnología de la información que es independiente de las herramientas y métodos utilizados en cualquier negocio de tecnologías de la Información en particular.

Algunas de las características esenciales de los marcos de trabajo son:

- Propone un modelo de colaboración.
- Reutiliza el diseño y el código.
- Incorpora el conocimiento para el dominio que el Marco de trabajo fue diseñado (Framework, 2018).

AngularJS, es un *framework* de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. (Technology, 2010)

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos.

AngularJS Material

AngularJS Material es un marco de componentes de la Interfaz de Usuario (UI) y una implementación de referencia de la especificación de diseño de materiales de Google. Este proyecto proporciona un conjunto de componentes UI reutilizables, bien probados y accesibles basados en el diseño de materiales.

Zeolides v2.2.0

El marco de trabajo para el desarrollo de aplicaciones *web* es el responsable de proveer un entorno de ejecución para los proyectos con funcionalidades y características óptimas de desarrollo. Brinda prestaciones que aseguran un alto grado de calidad e integración de las aplicaciones realizadas en dicha plataforma tecnológica. Aspectos como la seguridad, el monitoreo de trazas, la gestión del personal, los reportes, entre otros, son contenidos entre las funcionalidades de la solución. Por lo que permite la obtención de soluciones con un alto grado de robustez, dadas por el número de funcionalidades brindadas durante el desarrollo de soluciones *web* (Zeolides, 2017).

Zend Framework v1.9.7

Zend Framework (ZF) es un *framework open source* para *PHP* desarrollado por *Zend. ZF* implementa el patrón Modelo Vista Controlador (MVC), es 100% orientado a objetos y sus componentes tienen un bajo acoplamiento por lo que los puedes usar en forma independiente (ZendFramework, 2017).

Doctrine v1.2.1

El *Doctrine Project* es el hogar de varias bibliotecas *PHP* principalmente enfocadas en el almacenamiento de bases de datos y el mapeo de objetos. Los proyectos centrales son un *Object Relational Mapper* (mapeo de objetos relacionales) y la *Database Abstraction Layer* (Capa de abstracción de la base de datos) sobre la que se basa. Doctrine se ha beneficiado enormemente de los conceptos de *Hibernate ORM* y los ha adaptado para adaptarse al lenguaje *PHP* (Doctrine, 2017).

1.12 Servidor para aplicaciones web

Un servidor de aplicaciones es un programa de servidor en un equipo en una red distribuida que proporciona la lógica de negocio para un programa de aplicación. El servidor de aplicaciones se ve frecuentemente como parte de una aplicación de tres niveles, que consta de un servidor gráfico de interfaz de usuario (*GUI*), un servidor de aplicaciones (lógica empresarial) y un servidor de bases de datos y transacciones (Servidor de aplicaciones, 2017).

Apache HTTP Server v2.0

Apache es un proyecto de código abierto y uso gratuito, multiplataforma, muy robusto y se destaca por su seguridad y rendimiento. Por otro lado, viene con una licencia sin restricciones y se caracteriza por implementar características de uso frecuente (ApacheServer, 2018).

1.13 Software de integración Jenkins

Jenkins es un software de Integración continua *open source* escrito en Java. Proporciona integración continua para el desarrollo de software. Es un sistema corriendo en un servidor que es un contenedor, como *Apache Tomcat*. Soporta herramientas de control de versiones como CVS, *Subversion* y puede ejecutar proyectos basados en *Apache Ant* y *Apache Maven*, así como scripts de *shell* y programas *batch* de Windows. (Kohsuke, 2016)

1.14 Conclusiones del capítulo

Después del desarrollo de este capítulo se puede arribar a las siguientes conclusiones:

El estudio de los principales sistemas de gestión hospitalaria existentes en la actualidad posibilitó entender las principales características y funcionalidades que estos deben cumplir. El estudio de los elementos teórico relacionados con la gestión de los datos de los pacientes de nefrología permitió una mejor comprensión de los procesos a informatizar. La caracterización de tecnologías para el desarrollo de la propuesta de solución, permitió obtener una base tecnológica sólida para la implementación de la misma, basado en tecnologías libres que garantiza la soberanía tecnológica desde el punto de vista a la gestión hospitalaria. Se realiza un estudio del estado del arte tomándose varios sistemas similares, ya sea en el ámbito nacional e internacional, tomando las características positivas de los mismos para el módulo a implementar.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO PARA EL REGISTRO DE PACIENTES DE NEFROLOGÍA

2.1 Introducción

Durante este capítulo, utilizando diferentes estrategias y concepciones metodológicas, se darán razones que respalden la propuesta de solución. Se desarrollan las fases iniciales de la metodología de desarrollo propuesta en el manual Prodesoft: modelación y construcción; se presentan los diferentes artefactos que genera, que ayudará a entender de manera sencilla cómo funciona el proceso de digitalizado por parte del módulo, así como la descripción de los conceptos encerrados en él. Incluye También los prototipos de interfaz, así como diagramas de negocio y de base de datos, el levantamiento de los requisitos funcionales y no funcionales obtenidos luego de analizar las necesidades del cliente, una descripción de los mismos.

2.2 Propuesta de solución

El Sistema Informático para Nefrología (SINEF), es un sistema de alcance nacional de Gestión de Pacientes por modalidades de tratamiento en la especialidad de Nefrología en Cuba. Dentro del SINEF está el módulo para la gestión de los datos de los pacientes de nefrología posee varias funcionalidades tales como adicionar, buscar y modificar los datos de los pacientes de nefrología. Esto permitirá al personal de la salud acceder a la información almacenada de los pacientes de nefrología de todo el país en tiempo real, lo cual puede ser crucial en los diagnósticos de los doctores sobre los pacientes del área antes mencionada. El módulo no elimina pacientes por no detectarse dicha funcionalidad, en el levantamiento de requisitos.

2.3 Modelado del negocio

En este epígrafe se estará dando forma al negocio mediante el modelado, además de definir los principales conceptos del proceso de adicionar los datos de los pacientes y las relaciones que se establecen entre estos conceptos.

Modelo conceptual

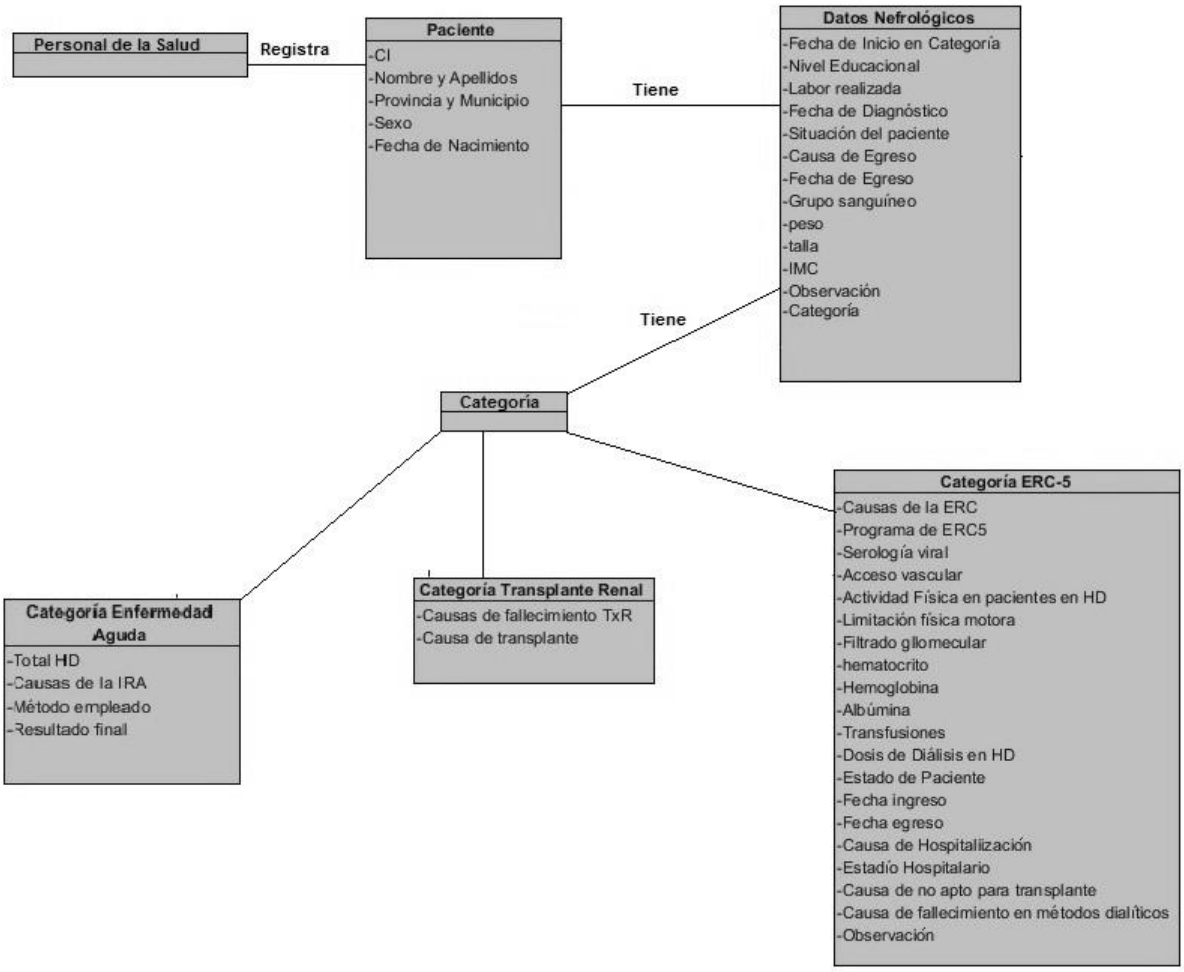


Figura 1. Modelo Conceptual del negocio de la propuesta de solución. *(Elaboración propia)*

Siguiendo los procesos que se buscan informatizar, se crea un modelo conceptual para la representación de los distintos conceptos que engloba la gestión de los datos de los pacientes de nefrología y la relación entre dichos conceptos, que a su vez son los que componen el diagrama Modelo Conceptual.

Datos nefrológicos: Son los datos generales de cualquier paciente de nefrología, independientemente de la categoría de su enfermedad.

Categoría Enfermedad Aguda: Es una ERC, en una etapa que todavía puede ser reversible.

Categoría ERC-5: Dígase de aquella Enfermedad Renal Crónica que ha alcanzado la quinta etapa.

Categoría Trasplante Renal: Es cuando se decide resolver la afección del paciente mediante un trasplante de riñón.

Pacientes: Ciudadano que debe ser atendido de manera médica.

Personal de la Salud: Es el trabajador que se encarga de los procesos para con el paciente.

Descripción de los procesos de negocios.

Con este mapa de procesos se persigue el objetivo de describir el proceso que sigue la Dirección Nacional del Ministerio de Salud Pública en cuanto al registro de pacientes.

Primeramente el paciente acude al cetro de atención médica, luego de que el personal médico calificado para ello lo registre, los datos del paciente quedan en el banco de datos.

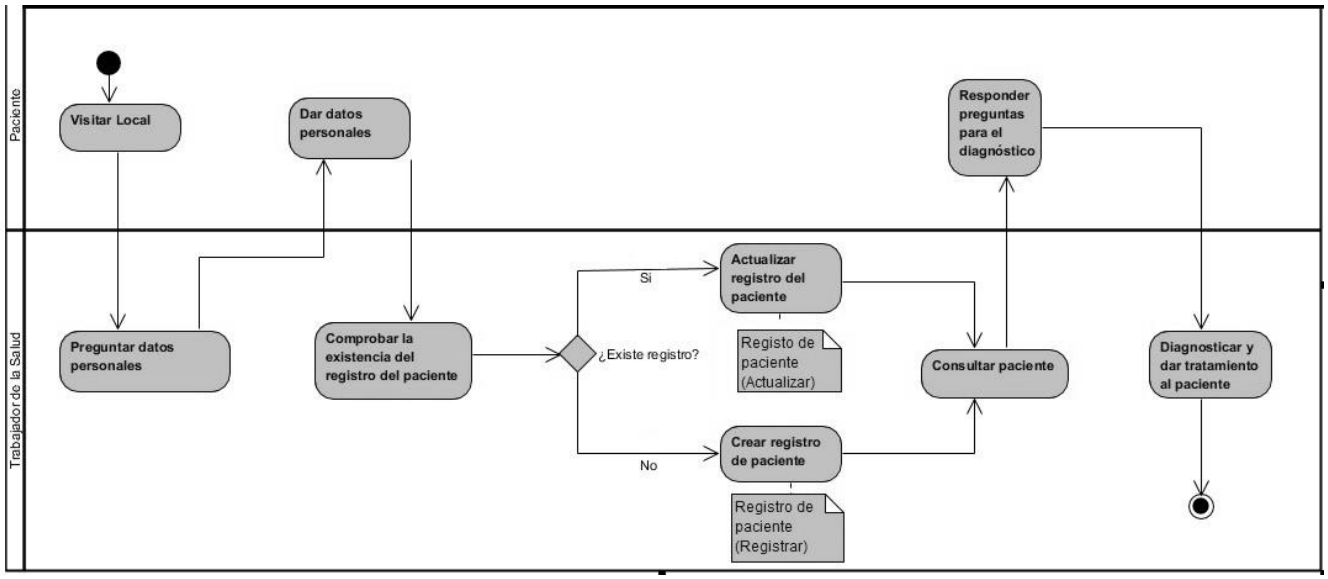


Figura 2. Diagrama de actividades del proceso de negocio gestionar paciente. *(Elaboración propia)*

2.4 Requisitos

En este epígrafe se evidencia el trabajo con los requisitos funcionales y no funcionales que trazaron la línea conductora a seguir, como se obtuvieron y los métodos usados para ello. También se incluye el diagrama de actividades el cual es necesario para una mayor comprensión del módulo a desarrollar.

2.4.1 Fuentes y técnicas para la obtención de requisitos

Las entrevistas al personal de la salud y la documentación que el Ministerio de Salud Pública pone al alcance de todos en la red nacional, fueron las principales fuentes para la obtención de los requisitos que se presentarán posteriormente.

La técnica usada fue **análisis de documentación,** la cual consiste, como su nombre indica, en analizar documentación que ya existe sobre un tema x para la elaboración de los requisitos, ya sean funcionales o no funcionales (PMOINFORMATICA, 2018).

2.4.2 Especificación de requisitos

A continuación se describen los requisitos funcionales y no funcionales usados para la creación del módulo para la gestión de los datos de los pacientes de nefrología.

2.4.2.1 Requisitos funcionales

RF 1 Adicionar paciente

RF 2 Modificar Paciente

RF 3 Buscar paciente

RF 4 Mostrar listado de pacientes

RF 5 Mostrar Seguimiento

2.4.2.2 Requisitos no funcionales del sistema

Para el levantamiento de requisitos no funcionales de hardware, se toman los requisitos del Sistema Informático para Nefrología (SINEF) (XETID, 2017), teniendo en cuenta la cantidad de usuarios y los estándares de despliegue de la empresa XETID para el marco de trabajo Zeolides se emiten los siguientes requisitos recomendados.

RNF Hardware

Para las PC clientes:

RNF 1 Procesador: Pentium 4 o superior.

RNF 2 RAM: 1 GB o superior.

Para los servidores web:

RNF 3 Procesador: Doble núcleo o superior.

RNF 4 RAM: 4GB o superior (para 60 usuarios o más).

RNF 5 Disco duro: 300 GB o superior.

RNF 6 Tarjeta de Red: 1 GB.

Para los servidores de base de datos:

RNF 7 Procesador: Doble núcleo o superior.

RNF 8 RAM: 4GB o superior.

RNF 9 Disco duro: 500GB o superior.

RNF 10 Tarjeta de Red: 1 GB.

Usabilidad:

RnF 11: El sistema debe ser una aplicación web.

RnF 12: La aplicación debe mostrar una interfaz agradable e intuitiva para el usuario a través de la disponibilidad de un botón de ayuda en cualquier momento, además de la presencia de tooltips para cualquier acción. El usuario podrá elegir entre una serie de temas para el color de la aplicación, así puede usar el matiz que encuentre agradable. Para el diseño de la interfaz se sigue el documento Estándar de diseño de interfaces para las aplicaciones de gestión Versión 1.0, emitido por la empresa XETID

Seguridad:

RnF 13: El sistema debe ser tolerante a fallos, y mostrar solo la información necesaria para orientar al usuario. El sistema se recupera de cualquier error mediante las validaciones implementadas y la información que muestra es solo estrictamente la necesaria.

RnF 14: Solo tendrán acceso a la aplicación las personas que se encuentren registradas en la misma.

Eficiencia:

RnF 15: El sistema debe permitir que los usuarios interactúen con él de manera concurrente. Podrán estar conectados al mismo tiempo, cerca de 100 y soporta alrededor de 255 peticiones concurrentes.

Funcionalidad:

RnF 16: La aplicación debe gestionar y requerir información de usuarios para su uso.

2.4.3 Descripción de requisitos

El sistema está compuesto por 5 requisitos funcionales, a continuación, se expone el diagrama de actividades para el RF 1 Adicionar paciente de nefrología, que servirá de ejemplo para la descripción de este requisito.

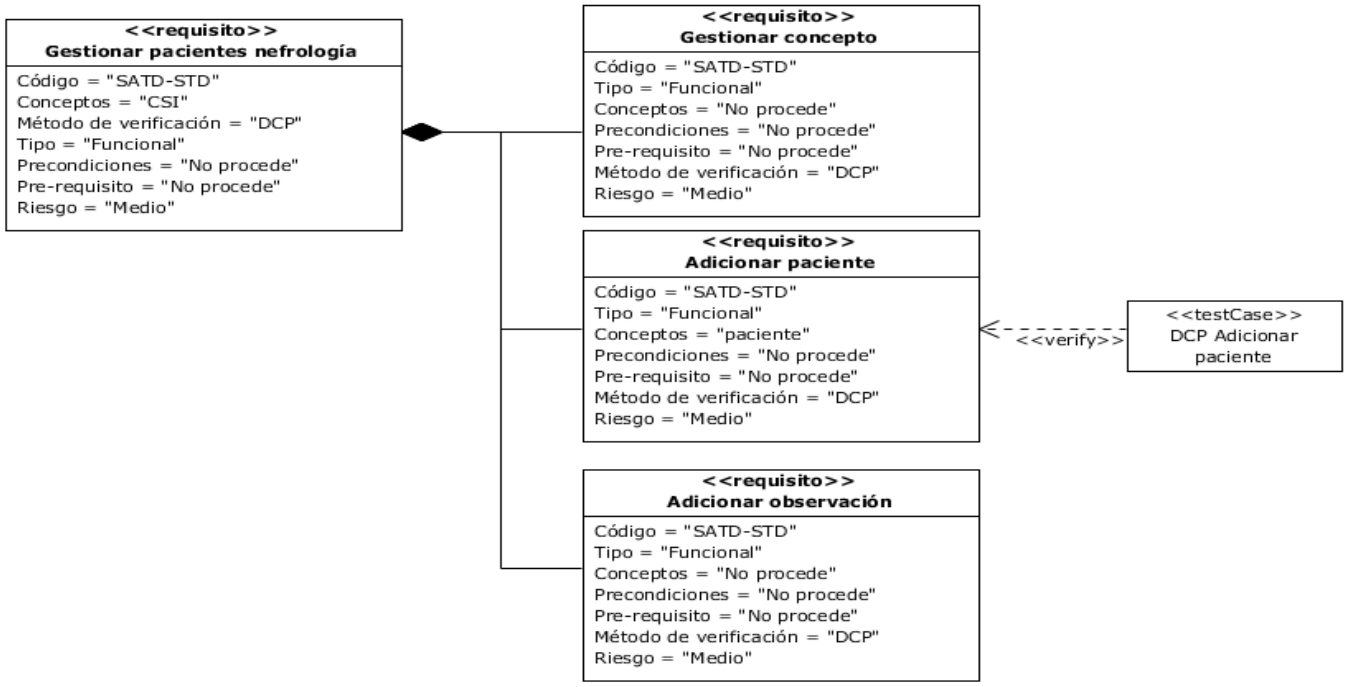


Figura 3. Diagrama de Requisitos caso genérico gestionar concepto. (Elaboración propia)

2.5 Análisis y diseño

Durante este epígrafe se estarán dando forma a los requisitos funcionales del sistema mediante diseño arquitectónico, el modelado de datos y modelado del diseño. También se tocarán temas como los patrones diseño y estándares de base de datos.

2.5.1 Diseño arquitectónico

El diseño arquitectónico debe satisfacer las necesidades del cliente, tanto en lo estético como en lo tecnológico. Entendiendo al diseño como proceso creativo encausado hacia una meta determinada, existen ciertas bases que apoyen su desarrollo y su creatividad. El correcto uso de un diseño

arquitectónico permite una mayor extracción del potencial de este y una mejor comunicación entre los componentes del sistema.

Para esta solución se propone usar una arquitectura por capas con el patrón de Modelo Vista Controlador que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario (Coplien). Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (Visual Paradigm, 2017).

De manera genérica, los componentes de MVC se podrían definir como sigue:

El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la "vista" aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al "modelo" a través del "controlador" (Visual Paradigm, 2017).

El **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su "vista" asociada si se solicita un cambio en la forma en que se presenta el "modelo" (por ejemplo, desplazamiento o *scroll* por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la "vista" y el "modelo" (Visual Paradigm, 2017).

La **Vista**: Presenta el "modelo" (información y *lógica de negocio*) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho "modelo" la información que debe representar como salida.

2.5.2 Modelado de datos

En el modelado de la base de datos el centro Integración utiliza un híbrido entre bases de datos relacionales y no relacionales, esto se observa en la relación entre tablas utilizando las llaves foráneas

como método de unificación entre las mismas y teniendo como meta ganar en rendimiento se usa ampliamente json donde se almacenan los valores de los datos nomenclados minimizando el uso de sentencias INNER JOIN debido a que el json almacena los datos que se necesitarían.

A continuación, se muestran los diagramas entidad relación definidos para el módulo de registro de pacientes de La Dirección Nacional del Ministerio de Salud Pública. Ver Figura 4 Diagrama entidad relación caso registro de pacientes.

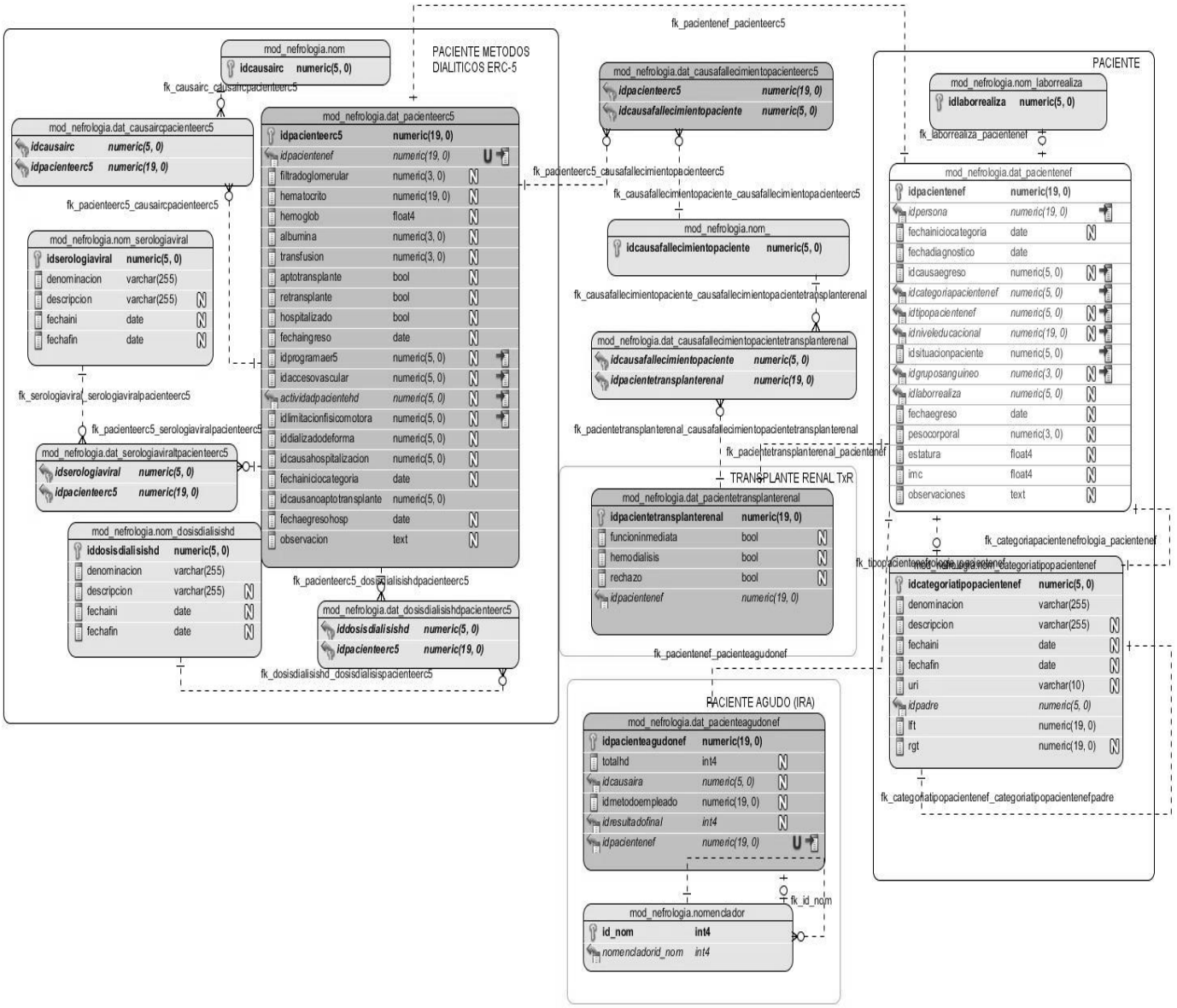


Figura 4. Diagrama entidad relación caso registro de pacientes. (Elaboración propia)

2.5.2.1 Estándar de base de datos

Se decide usar como estándar de base de datos la llave subrogada, pues además de cumplir con los parámetros de buenas prácticas, es un estándar bastante cómodo a la hora de trabajar con él y aporta varias ventajas tales como: mejorar el particionamiento eficiente de los datos físicos y crear una separación de modelos multidimensionales para facilitar el control de cambios así como también mejora el rendimiento de operaciones y permite leer más datos con menos operaciones de entrada y salida gracias a que el tamaño de los índices es menor.

La **clave subrogada** o ***surrogate key*** es el identificador único de una tabla que no se deriva de los datos de la aplicación, generalmente no es visible al usuario y se suele construir a partir de una secuencia autogenerada. El uso del patrón permite que las tablas sean más fáciles de consultar a partir del identificador, pues los tipos de datos son iguales en cada una de las tablas.

2.5.2.2 Plan de respaldo de la base de datos

Un plan de respaldo de bases de datos no es más que un modo de salvar la información que se tiene en los bancos de información hacia otro lugar, de manera periódica, para evitar su pérdida en caso de que ocurra algún tipo de evento imprevisto o falla en el sistema. Dicho período puede ser en intervalos de tiempo que van desde lo diario hasta semanal. Puede ser de periodos mayores a lo semanal pero por cuestiones de seguridad no se recomienda. Siguiendo lo antes planteado se decide hacer respaldos diarios de la base de datos.

Políticas de procedimiento:

- Se realizará una salva de todos los directorios del servidor de manera periódica.
- Los respaldos serán conservados según los acuerdos de las propias áreas donde se realizaron estos.
- Se seguirán los documentos rectores de la realización de los respaldos para los mismos.
- Todas las áreas deben contar con su respectivo plan de contingencia como lo plantea su plan de seguridad informática.
- Cada usuario es responsable de respaldar su información personal.

2.5.3 Modelado del diseño

A continuación se muestra la relación entre cada componente de los requisitos, lo cual es de gran importancia para su entendimiento.

2.5.3.1 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces (Ingeniería del software, 2018).

Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Erich Gamma, 2015).

Durante el desarrollo de este trabajo se utilizaron patrones de diseño como el patrón modelo-vista-controlador o MVC, *GRASP* (*General Responsibility Assignment software Patterns*, traducido como: Los patrones generales de asignación de responsabilidad de *software*) y *GOF* (*Gang of Four*, traducido al español Grupo de Cuatro). A continuación una breve descripción de dichos patrones.

Patrones *GOF* utilizados:

Singleton (instancia única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto. Se emplea en las consultas a la base de datos a través de Doctrine, dicha consulta es ejecutado por el objeto de la clase doctrine sin tener en cuenta el contenido de la misma. A continuación se muestra un ejemplo de este patrón:

```
        INNER JOIN mod_estructuracomp.nom_filaestruc fest ON (fest.idfila = est.idestructura)
        INNER JOIN mod_unidadsalud.nom_tipounidadsalud tus ON ((u.metadatos->>'tipounidadsalud')::NUMERIC = tus.idtipounidadsalud)
        LEFT JOIN mod_unidadsalud.nom_subordinacionunidadsalud sus ON ((u.metadatos->>'subordinacion')::NUMERIC = sus.idsubordinacionun
        INNER JOIN mod_datosmaestros.nom_dpt dpt ON (dpt.iddpt = est.iddpt)
        LEFT JOIN mod_estructuracomp.dat_estructura pol ON (pol.idestructura = (u.metadatos->>'policlinico')::NUMERIC)"
      . "Inner Join mod_workflow.nom_estado e on (mn.idestado = e.idestado) "
      . "WHERE true ";
    if (isset($filter->us)) {
        $idunidadsalud = $filter->us;
        $sqlNefro .= " and mn.idunidadsalud = $idunidadsalud ";
    }
    if (isset($filter->mes)) {
        $fechaainformar = $filter->mes;
        $sqlNefro .= " and lower(mn.fechaainformar->>'mes') Like '%" . strtolower($fechaainformar) . "%'";
    }
    $nefro = Doctrine_Manager::getInstance()->getCurrentConnection()->fetchAll($sqlNefro);
    return array('success' => true, 'data' => $nefro, 'count' => count($nefro));
} catch (Exception $exc) {
    return array('success' => false, 'msg' => $exc->getMessage());
}
```

Figura 5. Uso de consultas a la Base de Datos con Doctrine. (Elaboración propia)

Patrones *GRASP* utilizados:

Patrón creador:

El patrón creador ayuda a identificar que clase debe ser la responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto. La clase NefrologiaminsapController es la encargada de la creación de los objetos NefrologíaminsapModel.

```
class NefrologiaminsapController extends ZendExt_Controller_Secure {

    private $model;

    public function init() {
        parent::init();
    }

    public function apiRestNefrologiaAction() {
        try {
            $this->model = new NefrologiaminsapModel();
            $result = array();
            switch ($this-> request->getParam('query')) {
```

Figura 6 Instancia de NefrologiaminsapModel en NefrologiaminsapController. (Elaboración propia)

Patrón Controlador:

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método usado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. En el fragmento de código de la estructura condicional *switch* de la función apiRestNefrologíaAction dentro de la clase NefrologíaminsapController, los datos que solicita el usuario son suministrados por distintas clases, de acuerdo a la posibilidad de la misma para suplir la demanda.

```
class NefrologiaminsapController extends ZendExt_Controller_Secure {

    private $model;

    public function init() {
        parent::init();
    }

    public function apiRestNefrologiaAction() {
        try {
            $this->model = new NefrologiaminsapModel();
            $result = array();
            switch ($this->_request->getParam('query')) {
                case 'getAll':...
                case 'get':...
                case 'insert':...
                case 'delete':
                    $ids = json_decode($this->_request->getParam('params'));
                    $result = $this->model->delNefrologia($ids);
                    break;
                case 'getUs':
                    $filter = $this->_request->getParam('filter');
                    $result = $this->model->getUs($filter);
                    break;
                case 'getEstado':...
            }
            echo json_encode($result, options: JSON_NUMERIC_CHECK);
        } catch (Exception $e) {...}
    }
```

Figura 7. Método de la clase NefrologiaminsapController. (Elaboración propia)

Patrón Bajo Acoplamiento:

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre clases.

```
public function getAll($start, $limit, $filter, $restrincion)
{
    try {
        $sqlNefro = "Select est.denominacion as us, mn.fechaainformar->>'mes' as fechaainformar, mn.fecharegistro, e.denomina
            . "From mod_nefrologia.dat_modelonefrologia mn "
            . "Inner Join mod_unidadsalud.dat_unidadsalud u on (mn.idunidadsalud = u.idunidadsalud)
             INNER JOIN mod_estructuracomp.dat_estructura est ON (u.idunidadsalud = est.idestructura)
             INNER JOIN mod_estructuracomp.nom_filaestruc fest ON (fest.idfila = est.idestructura)
             INNER JOIN mod_unidadsalud.nom_tipounidadsalud tus ON ((u.metadatos->>'tipounidadsalud')::NUMERIC = tus.idtipoun
             LEFT JOIN mod_unidadsalud.nom_subordinacionunidadsalud sus ON ((u.metadatos->>'subordinacion')::NUMERIC = sus.id
             INNER JOIN mod_datosmaestros.nom_dpt dpt ON (dpt.iddpt = est.iddpt)
             LEFT JOIN mod_estructuracomp.dat_estructura pol ON (pol.idestructura = (u.metadatos->>'policlinico')::NUMERIC)"
            . "Inner Join mod_workflow.nom_estado e on (mn.idestado = e.idestado) "
            . "WHERE true ";
        if (isset($filter->us)) {...}
        if (isset($filter->mes)) {
            $fechaainformar = $filter->mes;
            $sqlNefro .= " and lower(mn.fechaainformar->>'mes') Like '%" . strtolower($fechaainformar) . "%'";
        }
        $nefro = Doctrine_Manager::getInstance()->getCurrentConnection()->fetchAll($sqlNefro);
        return array('success' => true, 'data' => $nefro, 'count' => count($nefro));
    } catch (Exception $exc) {
        return array('success' => false, 'msg' => $exc->getMessage());
    }
}
```

Figura 8 Ejemplo del patrón bajo acoplamiento en la clase NefrologiaminsapModel. (Elaboración propia)

2.5.3.2 Diagramas de clases del diseño

Un **diagrama de clases** en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos. (UML, 2017).

A continuación, se muestra un ejemplo de diagrama de clases del diseño con estereotipos *web*, el resto se podrán apreciar en los anexos.

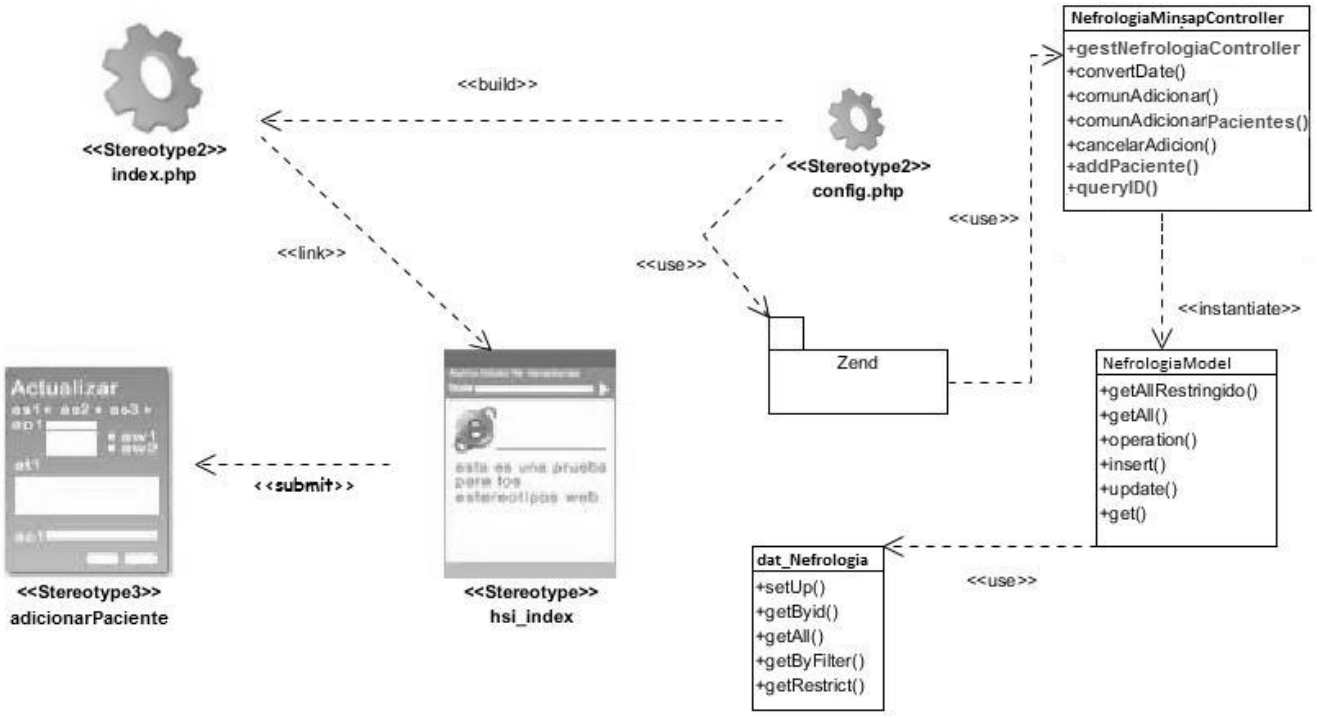


Figura 9. Diagrama de clases del diseño con estereotipos web caso adicionar. (Elaboración propia)

2.6 Prototipos de interfaz de usuario

La representación de los prototipos de interfaz de usuario (PIU) se realizó siguiendo la metodología utilizada para la elaboración de este trabajo investigativo.

A continuación, podemos ver el PIU para el requisito buscar paciente.

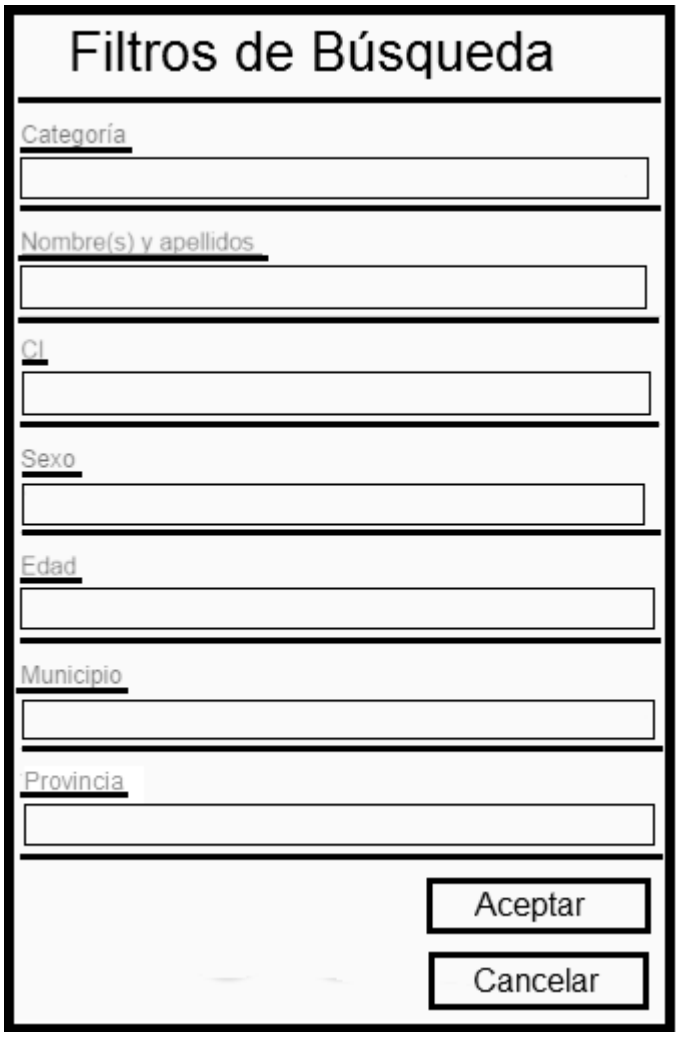


Figura 10. PIU 1 Buscar Paciente. (Elaboración propia)

2.7 Conclusiones del capítulo

Se proponen las siguientes conclusiones:

Con la definición de los requisitos funcionales y no funcionales, se logró comprender con mayor profundidad las capacidades del sistema. Con la modelación del sistema de acuerdo a los requerimientos del cliente, se logra una mejor visión de cómo se debe plasmar en la solución lo que se define en la propuesta. Se cumple con las buenas prácticas de programación, la reutilización de código y se define la estructura interna del software utilizando los patrones GOF y GRASP.

CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBAS Y VALIDACIÓN DEL MÓDULO PARA EL REGISTRO DE LOS PACIENTES DE NEFROLOGÍA

3.1 Introducción

Durante este capítulo se estará describiendo la implementación de la propuesta de solución, así como el valor de los requisitos obtenidos en el capítulo anterior. También se estará dando forma a los diagramas y a las principales interfaces de usuario. Como parte del desarrollo de este capítulo se estarán mostrando los resultados obtenidos en las pruebas de software realizadas a la propuesta de solución.

3.2 Implementación

A partir del diseño de la aplicación, se entra en la fase de implementación de la propuesta de solución formulada durante el capítulo anterior. Los objetivos de esta etapa estarán centrados en la implementación.

3.2.1 Modelo de implementación

En este subepígrafe se definen varios diagramas necesarios para una mayor comprensión de la aplicación, como son: diagrama de componentes y diagrama de despliegue

3.2.1.1 Diagrama de Componentes

El diagrama de componentes representa, como su nombre lo indica, los componentes en los cuales es dividido el sistema de software al cual se le diseñó dicho diagrama, mostrando, de cierta manera, las relaciones y dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. (Pressman, 2018)

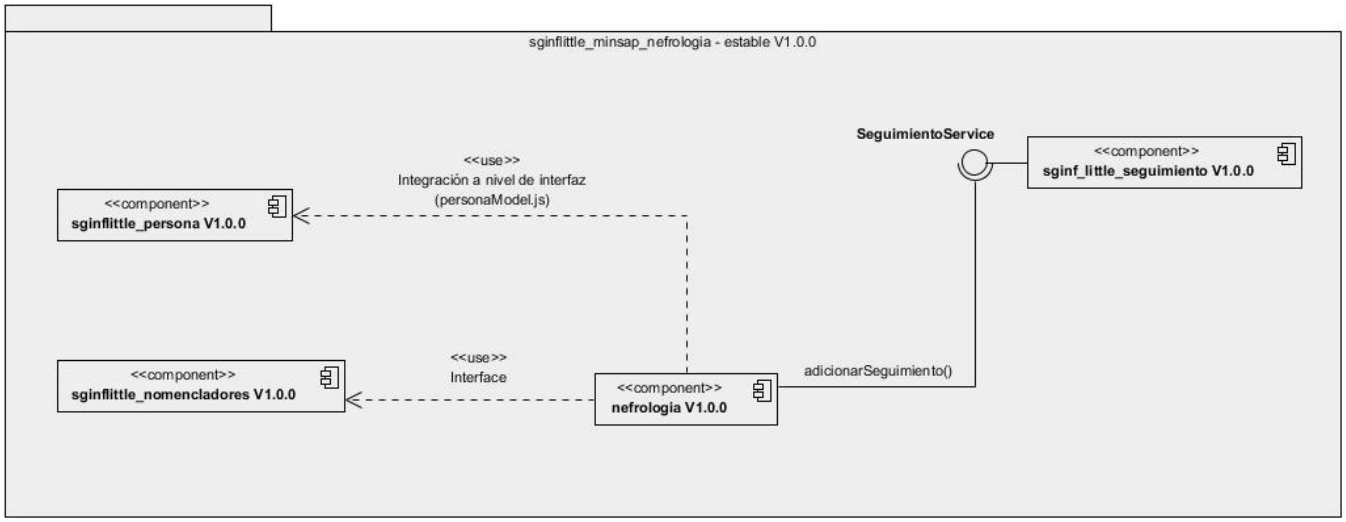


Figura 11. Diagrama de Componentes. (Elaboración propia)

3.2.1.2 Diagrama de Despliegue

En el siguiente diagrama de despliegue se toma como nodo, en el cual se originan las peticiones, **PC-Cliente** mediante protocolo HTTPS, al sistema alojado en el servidor de aplicaciones web. El sistema se comunicará con el servidor de Base de Datos PostgreSQL a través del protocolo TCP/IP.

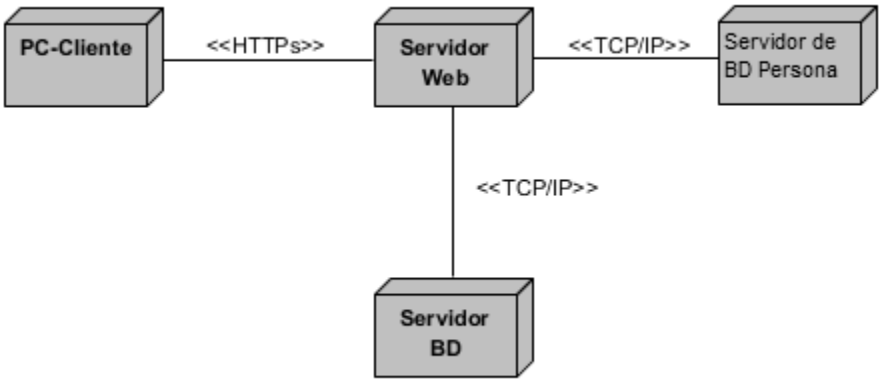


Figura 12. Diagrama de Despliegue. (Elaboración propia)

3.2.2 Estándares de implementación

Algunos lo llaman *Coding by convention* (**codificación** por convención) o simplemente *Coding Standards* (**estándares de codificación**), es un paradigma de programación que busca reducir el número de decisiones que el desarrollador tiene que tomar al momento de escribir su código. (2016)

Los estándares de implementación se basan en la unificación del trabajo para un mayor entendimiento del código desarrollado. Los mismos se proponen para contribuir a una sola idea de desarrollo para esto se utilizaron los siguientes estándares de codificación:

Algunos de los estándares reflejados en estos documentos son:

- ✓ Para la nomenclatura de las clases e interfaces se utilizará el estilo de capitalización UpperCamelCase con el cual se capitaliza la primera letra de cada palabra.
- ✓ El nombre de una clase comienza con la primera letra en mayúscula.

  Ejemplo: (class NefrologiaminsapController extends ZendExt_Controller_Secure)
- ✓ El nombre de los atributos de una clase comienza con la primera letra en minúscula.

  Ejemplo: ($this- model = new NefrologiamnsapModel();)
- ✓ Los nombres de las clases pertenecientes al dominio deben empezar con la palabra “Dat” y el de las clases del modelo deben tener la palabra “Model” al final.

  Ejemplo: (NefrologiaminsapModel.js)

En las clases controladoras:

- Los nombres de los métodos serán definidos con la palabra “Action” al final.

  Ejemplo: (vm.gestActions = function(action) {…})
- Cada método será público.
- El nombre de cada clase controladora debe terminar con la palabra “Controller”.

  Ejemplo: (NefrologiaminsapControllller.js)
- Se utilizará el método Post para las llamadas.

3.2.3 Interfaz gráfica de usuario

Una interfaz gráfica de usuario o GUI por sus siglas en inglés de ***Graphic User Interface,*** no es más que una forma de mediar entre el usuario y el código de la aplicación, para facilitar la interacción entre usuario y aplicación.

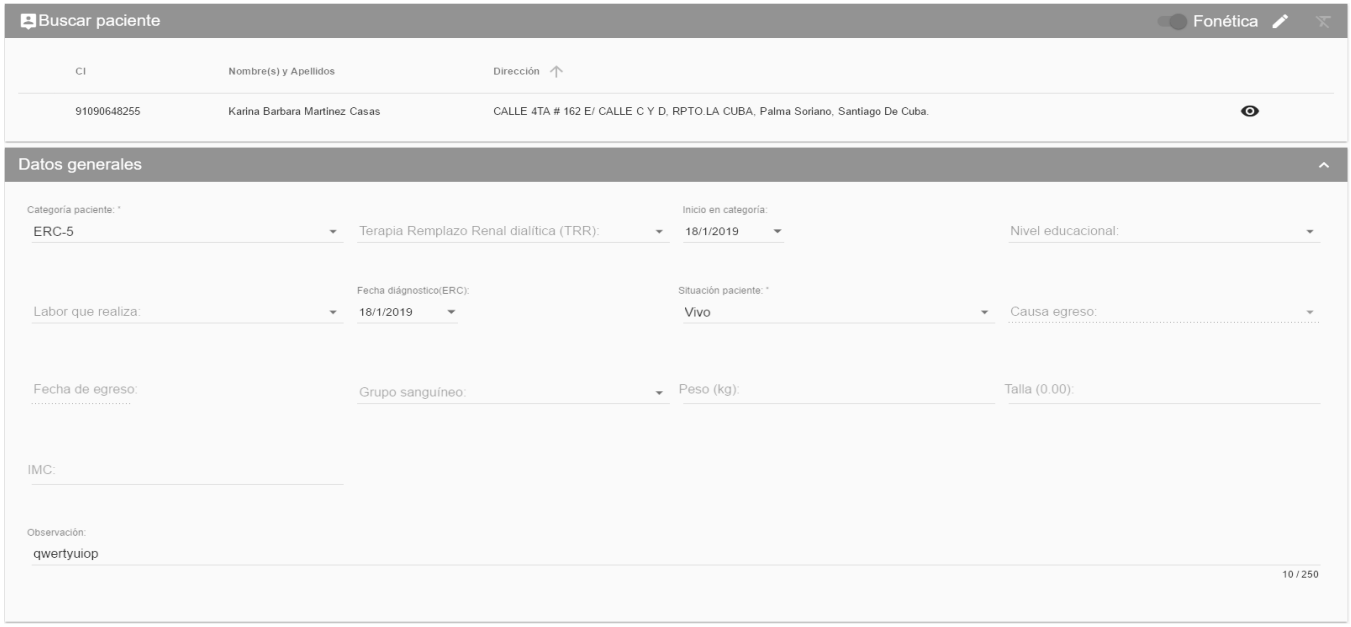


Figura 13. Interfaz de Insertar paciente. (Elaboración propia)

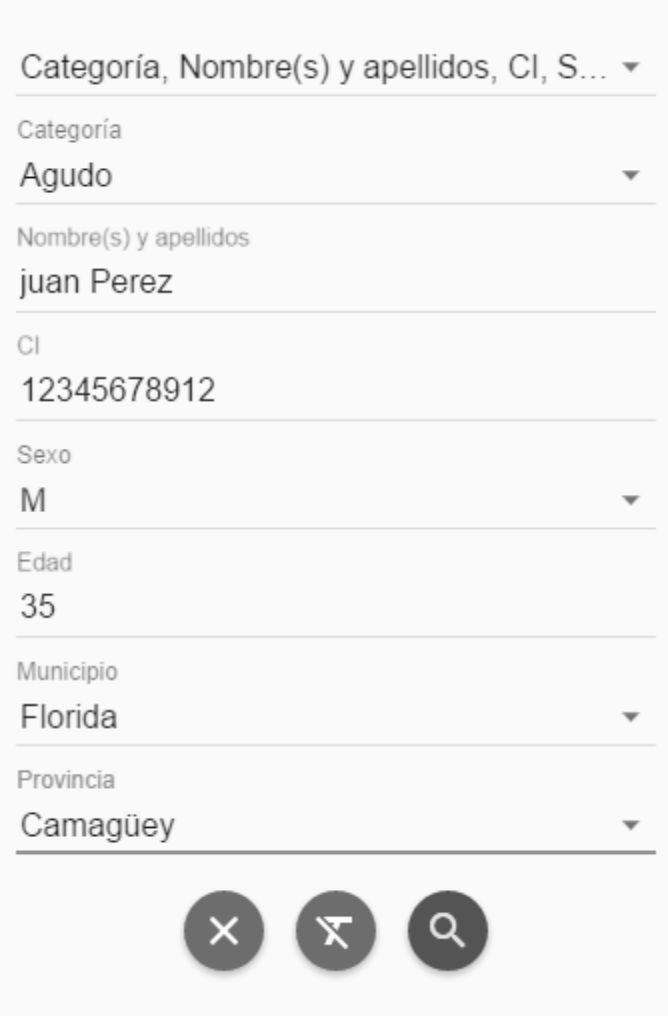


Figura 14. Interfaz de Buscar paciente. (Elaboración propia)

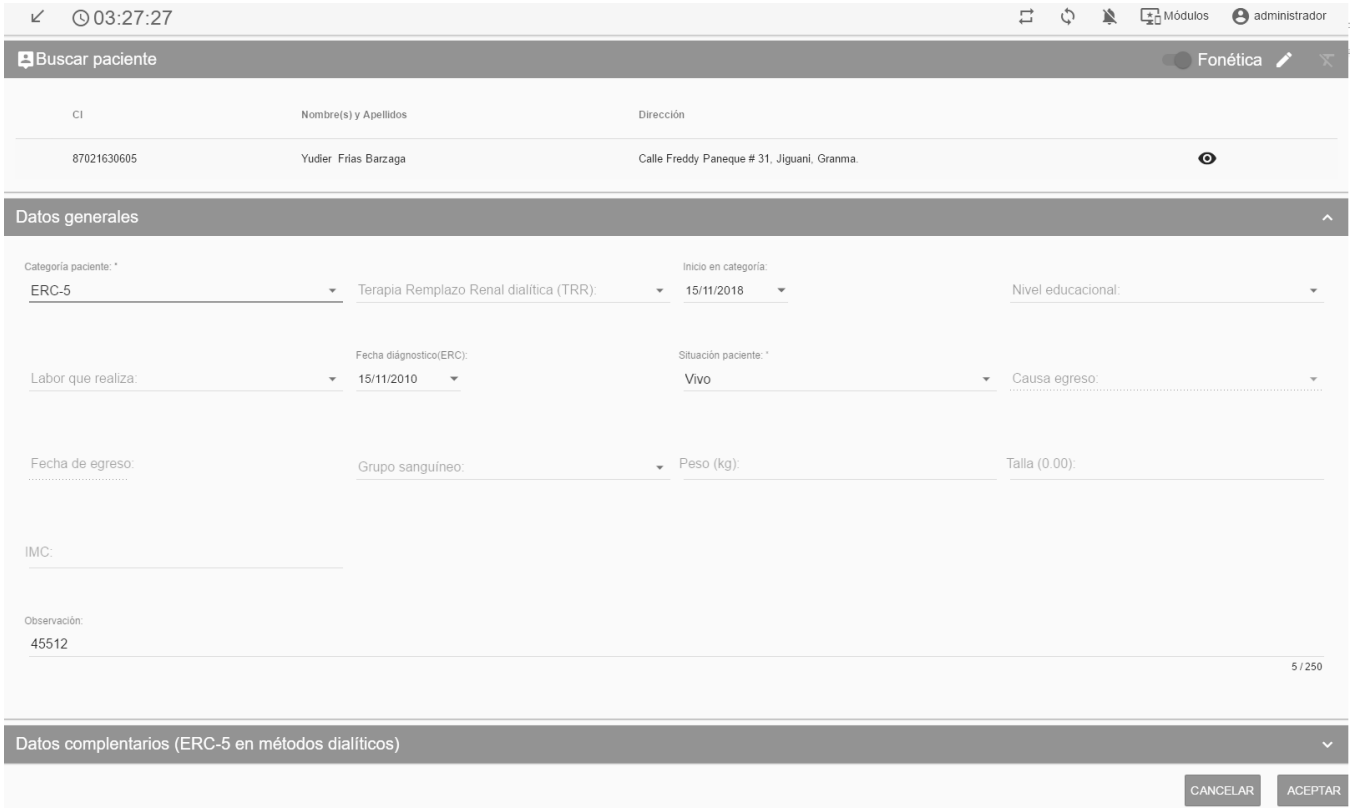


Figura 15. Interfaz de Editar paciente. (Elaboración propia)

Figura 16. Interfaz de Listar paciente. (Elaboración propia)

3.3 Pruebas de Software

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba debidamente seleccionados, por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el

propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo.

Tabla 2. Estrategia de pruebas. (Elaboración propia)

Nombre de la prueba	Método	Técnica
Funcionales	Caja Negra	Partición equivalente
Integración	Caja Negra	Casos de prueba
Sistema (carga y estrés)	Caja Negra	Automática mediante la herramienta JMETER
Unidad	Caja Blanca	Diagrama Piramidal y Gráfico de Abstracción/Inestabilidad por la herramienta Jenkins.

3.3.1 Métodos de Prueba

Las pruebas de caja blanca Se realizan sobre el código fuente de la aplicación a analizar, y según avanza el proceso de los casos de prueba, se determina donde existen errores para corregirlos en iteraciones posteriores.

Las pruebas de caja negra se toman una colección de condiciones de entradas, basadas en los requisitos funcionales del sistema y se prueban sobre el sistema, comprobando en cada caso la respuesta obtenida y la esperada. Este proceso se ejecuta en iteraciones para corregir los errores encontrados.

3.3.1.1 Tipos de prueba de caja negra

Pruebas funcionales

Prueba sobre el **requisito funcional** buscar paciente.

Tabla 3. Prueba realizada al requisito buscar paciente. (Elaboración propia)

Variables			
Id del escenario	**Escenario**	**C.I**	**Respuesta del sistema**
Ep.1.1	Buscar paciente de forma correcta	V [C.I (Con formato incorrecto y tamaño válido)]	Muestra una alerta de si encontró o no el paciente y lo muestra.
Ep.1.2	Buscar paciente de forma incorrecta	I [C.I (Con formato incorrecto y tamaño válido)]	El campo se muestra rojo y no realiza ninguna acción.
		I [C.I (Con formato correcto y tamaño incorrecto)]	El campo se muestra rojo y no realiza ninguna acción.
		I [C.I (Con formato y tamaño incorrecto)]	El campo se muestra rojo y no realiza ninguna acción.

Valores correctos:

C.I: Números y tamaño de 9 caracteres.

Tabla 4. Prueba realizada al requisito adicionar paciente. (Elaboración propia)

Variables			
Id del escenario	**Escenario**	**Carnet de Identidad(CI)**	**Respuesta del sistema**
Ep.1.1	Adicionar paciente de forma correcta	V [CI (rellenando con las opciones válidas)]	Muestra una alerta especificando que se adicionó el paciente.
Ep.1.2	Adicionar paciente sin completar los campos	I [CI (sin rellenar con opciones)]	Muestra un mensaje notificando la obligatoriedad del campo

Tabla 5. Prueba realizada al requisito modificar paciente. (Elaboración propia)

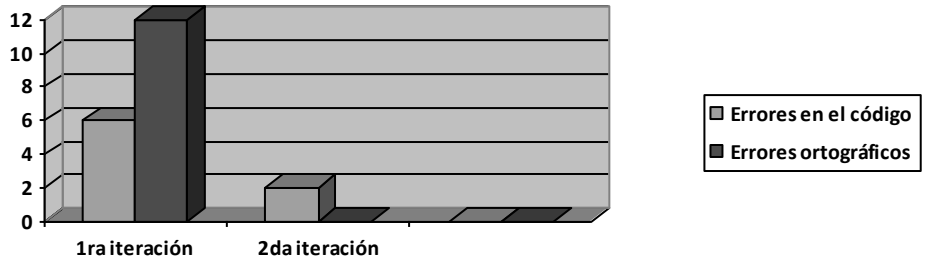
Variables			
Id del escenario	**Escenario**	**Nombre del paciente**	**Respuesta del sistema**
Ep.1.1	Modificar paciente de forma correcta	V [Nombre del paciente (seleccionando las opciones válidas)]	Muestra una alerta especificando que se modificó el nombre del paciente.

Ep.1.2	Modificar paciente de forma incorrecta	I [Nombre del paciente (sin seleccionar opciones)]	El campo no realiza ninguna acción.

Como resultado final de las pruebas se obtuvo que:

En una primera iteración de las pruebas se detectaron un total de 18 no conformidades de las cuales 12 eran sobre faltas de ortografía y las otras 6 en relación al código, quedando pendiente para una nueva iteración. Para la segunda iteración de las pruebas se corrigieron los errores encontrados en la primera iteración y no se detectaron nuevas no conformidades concluyendo las mismas con 2 iteraciones.

Tabla 6. Resumen sobre Pruebas unitarias. (Elaboración propia)



3.3.1.2 Pruebas de Integración

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software, una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan al unísono correctamente probándolos en grupo. Se centran principalmente en probar la relación entre los componentes, ya sea hardware o software.

En la validación de la solución se probó la integración del módulo para la gestión de pacientes de nefrología con los módulos Estructura, Seguridad, División Política Administrativa (D.P.A.) Persona y Datos Maestros. Las siguientes tablas muestran los casos de prueba de integración realizado entre los módulos. Al finalizar las pruebas de integración no se detectaron errores asociados a la interacción entre el módulo que se analiza con los módulos ya mencionados.

Tabla 7. Prueba de Integración módulos Estructura y Seguridad. (Elaboración propia)

Módulo al cual se integra	**Estructura y Seguridad**
Condiciones de Ejecución	Los módulos de Estructura y Seguridad almacenen los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba	Comprobar que los módulos HSF e HSI son capaces de realizar el acceso a funcionalidades y seguridad de negocio a partir de la información gestionada por los módulos de Estructura y Seguridad.
Entradas/Pasos de ejecución	Los módulos de Estructura y Seguridad introducen en la base de datos central los datos y los módulos HSF e HSI consultan estos datos y definen la seguridad del sistema.
Resultado esperado	Los usuarios tienen acceso a las funcionalidades de acuerdo al rol y sus responsabilidades.
Evaluación	Prueba satisfactoria.

Tabla 8. Prueba de Integración módulo Persona. (Elaboración propia)

Módulo al cual se integra	**Persona**
Condiciones de Ejecución	El módulo Persona almacene una nueva persona en la base de datos central y exista conexión con la misma.
Descripción de la prueba	Comprobar que los módulos HSF e HSI son capaces de visualizar las nuevas personas gestionadas por el módulo de Persona.
Entradas/Pasos de ejecución	El módulo Persona introduce en la base de datos central los datos de las nuevas personas y los módulos HSF e HSI consultan estos datos en el sistema.
Resultado esperado	Los usuarios tienen acceso a los datos introducidos por el módulo Persona.
Evaluación	Prueba satisfactoria.

Tabla 9. Prueba de Integración módulo Datos Maestros. (Elaboración propia)

Módulo al cual se integra	**Datos Maestros**
Condiciones de Ejecución	El módulo Datos Maestros introduce los datos en la base de datos central y exista conexión con la misma.
Descripción de la prueba	Comprobar que los módulos HSF e HSI son capaces de realizar el acceso a funcionalidades y seguridad de negocio a partir de la información gestionada por el módulo Datos Maestros.
Entradas/Pasos de ejecución	El módulo Datos Maestros introduce en la base de datos central los datos y los módulos HSF e HSI consultan estos datos en el sistema.

Resultado esperado	Los usuarios tienen acceso a los datos introducidos de acuerdo al rol y sus responsabilidades.
Evaluación	Prueba satisfactoria.

3.3.1.3 Pruebas de carga y estrés

En la ingeniería del software, las **pruebas de rendimiento** son las pruebas que se realizan, para determinar lo rápido que se ejecuta una tarea en el sistema bajo condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Las pruebas de rendimiento son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza por mejorar el rendimiento, englobándose en el diseño y la arquitectura de un sistema, antes incluso del esfuerzo inicial de la codificación. (O'Reilly)

Esta prueba fue realizada para 100 usuarios conectados en una 1ra iteración. Para una mejor comprensión de la tabla se detallan a continuación cada uno de los parámetros usados:

- **Muestras #:** indica la cantidad de usuarios haciendo peticiones de manera concurrente.
- **Media:** indica el máximo de tiempo de ejecución invertido para una petición.
- **Mediana:** significa que el 50% de las peticiones realizadas tardaron menos del valor reflejado.
- **Min:** indica el mínimo de tiempo de ejecución invertido para una petición.
- **Max:** indica el máximo de tiempo de ejecución invertido para una petición.
- **% Error:** indica la relación entre el total de peticiones y el número de peticiones que originaron errores.
- **Rendimiento:** hace referencia al número de peticiones que el servidor puede procesar en un segundo.
- **Kb/sec:** rendimiento medido en Kilobytes por segundo.

Tabla 10. Resultados de carga y estrés para 100 usuarios. (Elaboración propia)

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
3184 /std/st...	100	1757	1872	2263	849	2363	0,00%	40,9/sec	70,9
3186 /std/st...	100	378	372	697	24	860	0,00%	61,7/sec	35,7
3187 /std/st...	100	168	43	486	26	653	0,00%	91,8/sec	245,4
3185 /std/st...	100	59	38	101	26	320	0,00%	195,7/sec	1358,2
3188 /std/st...	100	34	33	39	25	184	0,00%	267,4/sec	852,5
3189 /std/st...	100	37	29	37	24	397	0,00%	137,4/sec	473,3
3190 /favico...	100	51	29	36	23	737	0,00%	69,6/sec	19,6
3191 /std/st...	100	88	36	187	26	745	0,00%	46,5/sec	419,5
3192 /std/st...	100	212	195	402	28	779	0,00%	35,7/sec	321,9
3195 /std/st...	100	569	550	885	154	1160	0,00%	25,4/sec	44,1
3193 /std/st...	100	864	885	1010	397	1249	0,00%	19,9/sec	34,5
3197 /std/st...	100	981	974	1097	815	1215	0,00%	18,2/sec	31,5
3194 /std/st...	100	1044	1059	1156	800	1229	0,00%	18,4/sec	31,9
3198 /std/st...	100	1063	1061	1208	760	1431	0,00%	18,3/sec	31,7
3196 /std/st...	100	1034	1028	1211	710	1427	0,00%	18,3/sec	31,7
3202 /std/st...	100	224	208	375	3	588	0,00%	21,9/sec	34,0
3199 /std/st...	100	929	896	1055	706	1608	0,00%	21,2/sec	36,7
3201 /std/st...	100	968	979	1173	212	1510	0,00%	25,2/sec	43,2
3200 /std/st...	100	922	965	1240	76	1679	0,00%	31,3/sec	51,4
3203 /std/st...	100	522	490	878	52	1259	0,00%	47,0/sec	73,1
3206 /succe...	100	0	0	0	0	1	100,00%	72,3/sec	120,0
3207 /succe...	100	0	0	0	0	0	100,00%	72,5/sec	120,4
Total	2200	541	411	1124	0	2363	9,09%	176,5/sec	471,7

En cada una de las iteraciones con la muestra de usuarios establecida la ocurrencia de errores es de 0.00% lo que demuestra que las peticiones son realizadas de forma satisfactoria en todo momento.

3.3.1.4 Tipos de prueba de caja blanca

La pirámide que se muestra representa en su parte derecha el nivel de acoplamiento existente entre las clases de los módulos, en su parte izquierda se hace referencia al tamaño de los módulos teniendo en cuenta las cantidades de paquetes (NOP), clases (NOC), métodos (NOM) y líneas de código (LOC), así como también la suma de complejidades ciclomáticas de todos los métodos de los módulos. (Kohsuke, 2016).

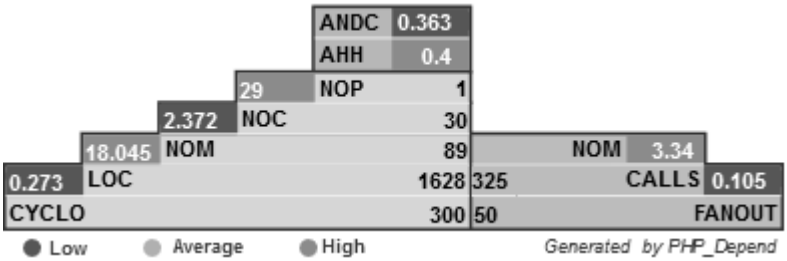


Figura 17. Gráfico piramidal (Módulo para gestión de pacientes de nefrología) (Elaboración propia).

El siguiente gráfico de Abstracción/Inestabilidad, se refiere en su eje vertical al acoplamiento entre paquetes o clases, en el cual mayor inestabilidad tendrá el paquete, mientras más cercano a uno sea el valor obtenido, indicando una mayor dependencia a otros. En su eje horizontal muestra la relación entre clases abstractas y clases concretas donde el valor uno muestra una mayor cantidad de clases e interfaces abstractas.

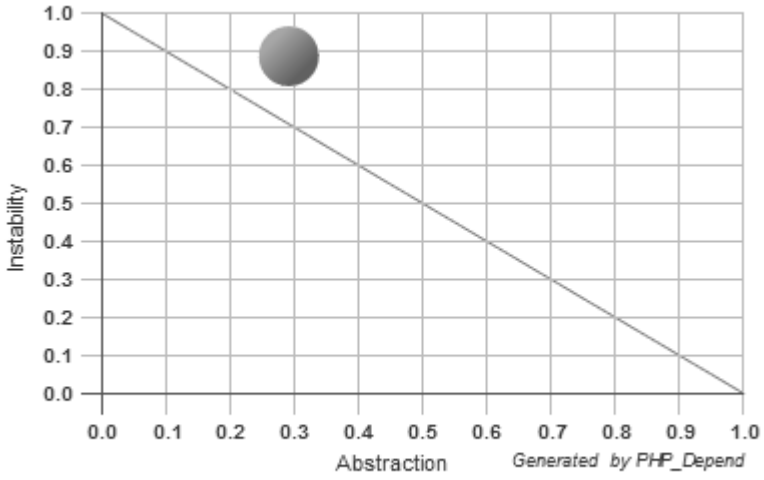


Figura 18. Gráfico de Abstracción/Inestabilidad . (Elaboración propia)

El módulo para la gestión de pacientes de nefrología poseen un alto nivel de herencia (AHH), las llamadas realizadas por cantidad de métodos (CALLS/NOM) junto al promedio de líneas de código por método (LOC/NOM) sobrepasan el máximo permisible.

Al analizar los gráficos arrojados por la herramienta se obtuvo que:

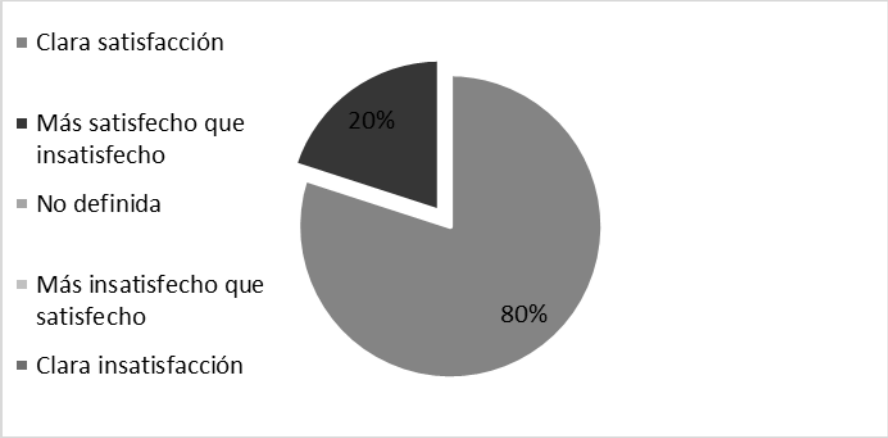
En el módulo actual la cantidad promedio de líneas de código por método (LOC/NOM) y las llamadas por cantidad de métodos (CALLS/NOM) supera el máximo permisible, además de poseer un nivel alto de herencia (AHH).

Validación de los resultados obtenidos.

Para evaluar la satisfacción con el módulo, se escogieron veinte trabajadores de la institución como muestra. A partir de dicha muestra para determinar el nivel de satisfacción individual y grupal se utilizó la Técnica de IADOV. Los resultados individuales de la satisfacción se resumen en la

.

Tabla 11. Índice de satisfacción individual



Para obtener el Índice de Satisfacción Grupal (ISG) se parte de asociar los diferentes niveles de satisfacción de los encuestados con una escala numérica que oscila entre +1 y – 1, de la siguiente forma:

Tabla 12. Índice de satisfacción grupal

Escala	Significado	Satisfacción individual	%
+1	Clara satisfacción	16	80
+0.5	Más satisfecho que insatisfecho	4	20
0	No definida	0	0
-0.5	Más insatisfecho que satisfecho	0	0
-1	Clara insatisfacción	0	0

Durante la investigación el Índice de Satisfacción Grupal tuvo un valor de: ISG = 0,90. Como se puede apreciar el valor del Índice es alto, de ahí que se perciba, además de la clara aceptación de la misma, un reconocimiento a su utilidad, en tanto los usuarios han emitido criterios donde evidencian su satisfacción por la contribución del módulo.

El siguiente cuestionario empleado para determinar el grado de satisfacción de los usuarios con respecto a la propuesta del módulo, cuenta con un total de cinco preguntas, de ella tres cerradas (1, 2 y 3) y dos abiertas (4 y 5), cuya relación ignora el sujeto. Estas tres preguntas cerradas se relacionan a través del Cuadro lógico de Iadov.

	P1: ¿Se siente satisfecho con el resultado obtenido con la aplicación ?								
	Si			No sé			No		
	P2: ¿Se siente usted que esta herramienta le es útil para agilizar su trabajo?								
¿Le gusta la forma en que se diseñó el módulo?	Si	No sé	No	Si	No sé	No	Si	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que disgusta	2	2	3	2	2	3	6	3	6
Me es indiferente	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	6	4	3	4	4
No me gusta	6	6	6	6	6	4	6	4	5
No puedo decir	2	3	6	3	6	3	6	3	4

Imagen 18. Cuadro Lógico de Iadov.

La encuesta cuenta con dos preguntas de carácter abierto. Estas son:

P1: ¿Está usted satisfecho con el resultado obtenido con la puesta en funcionamiento del módulo?

P2: ¿Siente usted que esta herramienta le es útil para agilizar su trabajo?

P3: ¿Le gusta la forma en que se diseñó el módulo?

P4: ¿Qué importancia le concede al módulo?

P5: ¿Qué aspectos a su juicio potencian el uso de este módulo?

Con respecto a la importancia que le conceden al módulo respondieron lo siguiente:

Permite registrar al paciente de manera más rápida, así como mantener un seguimiento y control de la información.

Facilita el control y el análisis de la información registrada.

Con relación a la pregunta cinco, señalaron que:

Mejorar la calidad de organización y seguimiento de los pacientes.

Resulta significativo en el análisis de estas opiniones, la preponderancia de aspectos positivos, lo cual sirve como fundamento del alto valor obtenido en el ISG.

3.4 Conclusiones del capítulo

En este capítulo se han abordado los elementos de la implementación del módulo para la gestión de pacientes, así como las pruebas realizadas al mismo y los resultados obtenidos; lo cual permite arribar a las siguientes conclusiones:

Con la elaboración del diagrama de componentes, se facilita la comprensión de la estructura general del sistema mediante sus componentes. El correcto uso de los estándares de codificación, permitió una mejor legibilidad y comprensión del código, facilitando su mantenimiento. La implementación del sistema permitió obtener una aplicación funcional y totalmente operativa. La ejecución de la estrategia de pruebas especificada, permitió detectar y corregir deficiencias presentes en la solución y ofrecer una aplicación con mayor calidad, seguridad y usabilidad. La evaluación del objetivo de la investigación, ofreció una valoración satisfactoria de la propuesta de solución.

CONCLUSIONES GENERALES

Una vez cumplidos los objetivos trazados en la presente investigación, se concluye lo siguiente:

- A través de la investigación y estudio de homólogos se fortalecen los procesos y características deseadas para dar solución a la problemática planteada.
- Con la ingeniería de requisitos y los artefactos que se generan, se alcanza un punto clave para la ejecución de la propuesta de solución.
- La realización de las pruebas de software, permitió comprobar el correcto funcionamiento de la aplicación.
- El módulo desarrollado permite la gestión de la información de los pacientes de nefrología de manera ordenada.
- La posibilidad de conocer y consultar en cualquier momento la información de los pacientes favorece la disponibilidad de la información.

GLOSARIO DE TÉRMINOS:

AHH: Altura promedio de jerarquía desde la clase raíz hasta las subclases que dependen de ellas.

ANDC: Número de clases derivadas.

Base de Datos (BD): Colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos es un sistema de archivos electrónico.

CALLS: Número de llamadas a métodos y funciones.

Código Abierto (*Open Source*): Término por el que se conoce al software que es desarrollado y distribuido libremente

CYCLO: Promedio de la complejidad ciclomática del código.

DSL: *Domain Specific Languages*, cualquier lenguaje que esté especializado en modelar o resolver un conjunto específico de problemas.

GHZ: Frecuencia en el que un cristal de cuarzo emite una señal de reloj que regula un ciclo de un circuito integrado síncrono, principalmente en microprocesadores (aunque no exclusivamente), sustituyendo al megahercio como unidad en que más frecuentemente se indica la velocidad de una CPU.

Hardware: Conjuntos de componentes que integran la parte física de una computadora.

HTML: *HyperText* Markup *Language* (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.

HTTP: *Hypertext Transfer Protocol* en español significa Protocolo de Transferencia de Hipertexto, es utilizado para establecer una conexión web a través de un navegador.

IOC: Inversión de control.

JSON: Acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos.

NOM: Se reflejan las clases cuyo número de métodos (*nom*) sea de 10 o mayor.

NOC: Reflejan los paquetes que posean los números de clases de 26 o mayor.

Software: Equipo lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware.

SGBD: Sistema Gestor de Base de Datos, tiene la función de controlar la entrada y salida de los datos de una base de datos.

SATD: Sistema de Apoyo a la Toma de Decisiones.

XHTML: Extensible *HyperText Markup Language* (lenguaje extensible de marcado de hipertexto), es el lenguaje pensado para sustituir a HTML como estándar para las páginas web.

RECOMENDACIONES

Tomando en cuenta los resultados de la investigación se emiten las siguientes recomendaciones.

- Actualizar la versión de PHP usada de versión 5.4 a una versión superior.
- Introducir analizadores estadísticos más potentes que cuenten con inteligencia artificial.
- Acelerar el proceso de despliegue para su explotación.

ANEXOS

Anexo 1 ver Figura 19 Diagrama de Entidad Relación tamaño completo

Figura 19 Diagrama de Entidad Relación tamaño completo

BIBLIOGRAFÍA

1. *¿Qué es un framefork?* (s.f.). Recuperado el 3 de enero de 2018, de https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza
2. *ApacheServer*. (6 de febrero de 2018). Obtenido de Versión 2.4 de la documentación del Servidor de HTTP Apache - Servidor Apache HTTP Versión 2.4.: http://httpd.apache.org/docs/current/.
3. AULBACH, E., WINSTEAD, J., TORBEN WILSON, L., LERDORF, R., ZMIEVSKI, A., & AHTO, J. (s.f.). *Manual de PHP*. S.l.: s.n.
4. B Braun. (2017). *B Braun condiciones de uso*. Obtenido de https://www.bbraun.es/es/condiciones-de-uso.html
5. Barcelona, H. S. (4 de 3 de 2018). *Sitio Oficial del Hospital Sant Joan de Déu Barcelona*.
6. Benemérita Universidad Autónoma de Puebla, Dr. Mario Rossainz López. (2016). *Cursos del Periodo de Primavera 2016.* Recuperado el 5 de 4 de 2016, de http://rossainz.cs.buap.mx/IngSw_2016/Proyecto/5_PruebasDeSw.pdf
7. *blog.eltallerweb.com*. (abril de 2016).
8. *BPMN vs UML*. (21 de noviembre de 2017). Obtenido de https://articulosit.files.wordpress.com/2014/01/bpm-vs-uml.pdf
9. Braun, B. (25 de 11 de 2018). *Nexadia® expert*.
10. *Doctrine*. (23 de noviembre de 2017). Obtenido de http://www.doctrine-project.org/.
11. Dr. Alejandro López Rodríguez, D. V. (Abril de 2002). *efdeporte*. (revista digital · Año 8 · N° 47 | Buenos Aires) Recuperado el 13 de 4 de 2018, de http://www.efdeportes.com/: http://www.efdeportes.com/efd47/iadov.htm
12. *DriCloud*. (6 de febrero de 2018). Obtenido de https://dricloud.com/
13. Erich Gamma, R. H.-A. (2015). *Design Patterns. Elements of Reusable Object-Oriented Software.*

14. *Framework*. (3 de enero de 2018). Obtenido de https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza

15. *Galén*. (12 de febrero de 2017). Obtenido de Sistema de Información para la gestión y coordinación de procesos en un servicio de Oncología.: http://www.lcc.uma.es/~lfranco/B12-Ribelles+Jerez++10.pdf.

16. Gálen. (25 de diciembre de 2018). *Sistema de Información para la gestión y coordinación de procesos en un servicio de Oncología*. Obtenido de http://www.lcc.uma.es/~lfranco/B12-Ribelles+Jerez++10.pdf.

17. *Gestores de BD*. (3 de enero de 2018). Obtenido de https://es.slideshare.net/nipas/10-sgbd.

18. *Herramienta de Modelado*. (21 de noviembre de 2017). Obtenido de http://www.alegsa.com.ar/Dic/herramienta_de_modelado.php.

19. *HTML 5*. (22 de noviembre de 2017). Obtenido de HTML5 - HTML | MDN: https://developer.mozilla.org/es/docs/HTML/HTML5.

20. Infomed. (18 de septiembre de 2018). *Indomed Red de salud en Cuba*. Obtenido de www.sld.cu

21. *Ingeniería del software*. (8 de febrero de 2018). Obtenido de http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

22. *JavaScript*. (22 de noviembre de 2017). Obtenido de Documentación web de MDN: https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript.

23. Kohsuke, K. (2016). *Hudson.*

24. *Medicina General Integral*. (6 de febrero de 2018). Obtenido de http://sparraparramecubana.blogspot.com/2011/04/medicina-general-integral.html

25. *MediCloud*. (6 de febrero de 2018). Obtenido de https://medicloud.me/

26. *NCBI Bookshelf*. (6 de febrero de 2018). Obtenido de ¿Qué es el historial de salud familiar? Y ¿Cómo me afecta? - ¿Cómo hablo con mi familia sobre Gaucher?: https://www.ncbi.nlm.nih.gov/books/NBK115576/

27. *Netbeans*. (3 de enero de 2018). Obtenido de https://netbeans.org/.

28. O'Reilly. (s.f.). *The Art of Performance Testing.* ISBN 978-0-596-53066-3.

29. Pedesoft. (25 de noviembre de 2018). *Pedesft| Intranet Corporativa XETID*. Obtenido de https://intranet.xetid.cu/content/prodesoft.

30. *PGADMIN*. (24 de noviembre de 2017). Obtenido de PgAdmin PostgrestSQL Tools: http://www.pgadmin.org/translation/

31. *PMOINFORMATICA*. (6 de febrero de 2018). Obtenido de PMOINFORMATICA. 7 Técnicas de levantamiento de requerimientos software.: http://www.pmoinformatica.com/2016/08/tecnicas-levantamiento-requerimientos.html.

32. *PostgreSQL*. (7 de febrero de 2018). Obtenido de https://www.postgresql.org

33. *Prodesoft*. (20 de noviembre de 2017). Obtenido de Prodesoft | Intranet Corporativa XETID.: https://intranet.xetid.cu/content/prodesoft.

34. *Real Academia de la Lengua Española.* (2018).

35. Rioja Salud. (28 de 11 de 2018). *www.larioja.es*. Obtenido de https://www.riojasalud.es/ciudadanos/catalogo-multimedia/nefrologia/la-lista-de-espera

36. Rockville Pike, B. M. (10 de 4 de 2018). *Mediline Plus*. Recuperado el 18 de 8 de 2018

37. Rumbaugh, J. J. (1998). *Lenguaje Unificado de Modelado, Manual de referencia.*

38. *Servidor de aplicaciones*. (25 de noviembre de 2017). Obtenido de http://searchdatacenter.techtarget.com/es/definicion/Servidor-de-aplicaciones.

39. SineMed System, S.A. (2016). *www.sinemed.com*. Obtenido de http://www.sinemed.com/recursos/docs/HIS.pdf

40. *SML*. (6 de febrero de 2018). Obtenido de Sistema Médico en Línea: http://smlmedico.com/

41. Software, A. (16 de 2 de 2018). *Aquar Software*.

42. Technology, M. M. (2010). *What is AngularJS?*

43. The DCI Architecture: A New Vision of Object-Oriented Programming. (s.f.). En T. R. Coplien.

44. *UML*. (21 de noviembre de 2017). Obtenido de El Lenguaje Unificado de Modelado: https://campusvirtual.ull.es/ocw/mod/resource/view.php?id=8373.

45. Varios. (2015). *AngularJS FAQ*.

46. *Visual Paradigm*. (21 de noviembre de 2017). Obtenido de https://www.visual-paradigm.com/.

47. *XAVIA - ucisis*. (1 de marzo de 2018). Obtenido de https://www.felti.org/sites/default/files/archivos/postulacion/ucisis.pdf.

48. XETID. (2017). *Levantamiento de requisitos para Sistema Informático para Nefrología (SINEF).*

49. *ZendFramework*. (23 de noviembre de 2017). Obtenido de http://www.maestrosdelweb.com/guia-zend/.

50. *Zeolides*. (20 de noviembre de 2017). Obtenido de Zeolides - Xetid.: https://www.xetid.cu/producto/33.