Universidad de las Ciencias Informáticas

Facultad 3

Módulo de Autopsias Perinatales para el Sistema Informático de Registro y control en Anatomía Patológica para la Red Hospitalaria de Cuba.

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

Autor:

Daymari Martínez Rodríguez.

Tutores:

Ing. Pedro Arango Astorga

MsC. Mailen Edith Escobar Pompa

Asesor:

DR. CS. José Hurtado De Mendoza Amat

La Habana, agosto de 2020

"Año 62 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser la única autora del trabajo de diploma con título “Módulo de Autopsias Perinatales para el Sistema Informático de Registro y Control en Anatomía Patológica para la Red Hospitalaria de Cuba” y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación con carácter exclusivo.

_____

Daymari Martínez Rodríguez

Autor

_____

Ing. Pedro Arango Astorga

Tutor

_____

MSC. Mailen Edith Escobar Pompa

Tutor

Dedicatoria

A mi familia y amigos, a todos los profesores, personas que me apoyaron durante el tránsito de la carrera, y a mi novio.

Resumen

Las autopsias son uno de los procedimientos médicos donde más datos se recopila de un paciente fallecido, de la anatomía humana y sus patologías. Dicha información puede ser usada en investigaciones científicas y para la toma de decisiones médicas. Para el almacenamiento y procesamiento de esta información existe un Sistema informático de Registro y Control en Anatomía Patológica para la Red Hospitalaria de Cuba desarrollado en el año 2014, que se adapta a estándares cubanos y usados por la Organización Mundial de la Salud, donde se gestiona la información referente a las autopsias de adultos, pero no recoge los datos de las autopsias perinatales, que se diferencia en muchos aspectos de las realizadas a una persona en etapa de adultez. El objetivo de este trabajo es desarrollar un módulo de autopsias perinatales que permita la obtención y tratamiento de la información de dicha necropsia. Para ello se realizó un estudio de diferentes sistemas de gestión médicos en todo el mundo, y de las tecnologías más apropiadas adaptables a los recursos tecnológicos de los hospitales del país donde, se siguió la línea de trabajo del sistema principal y se elaboraron un conjunto de reportes dinámicos que permite que el usuario pueda definir qué información requiere en cada reporte, bajo un criterio de búsqueda deseado. Este módulo permite el máximo aprovechamiento de la información generada durante la autopsia perinatal y facilita su uso en un momento dado.

Palabras claves: autopsias, perinatal, necropsia, reportes dinámicos.

Abstract

Autopsies are one of the medical procedures where more data is collected from a deceased patient, from the human anatomy and its pathologies. Such information can be used in scientific research and for medical decision making. For the storage and processing of this information, there is a Computer System for the Registration and Control of Pathological Anatomy for the Cuban Hospital Network developed in 2014, which is adapted to Cuban standards and used by the World Health Organization, where the information regarding adult autopsies is managed, but does not collect data from perinatal autopsies, which differs in many aspects from those performed on a person in adulthood. The objective of this work is to develop a perinatal autopsy module that allows obtaining and processing the information of said autopsy. For this, a study of different medical management systems around the world was carried out, and of the most appropriate technologies adaptable to the technological resources of the country's hospitals, where the line of work of the main system was followed and a set of dynamic reports that allow the user to define what information is required in each report, under a desired search criteria. This module allows maximum use of the information generated from the corpse of the perinatal baby and facilitates its use at any given time.
Keywords: autopsies, perinatal, necropsy, dynamic reports.

ÍNDICE

ÍNDICE

ÍNDICE

Índice de Tablas

ÍNDICE

Índice de Figuras

INTRODUCCIÓN

Introducción

El objetivo final de un sistema de salud es prolongar y mejorar en calidad la vida del hombre. La muerte es, por tanto, su mayor fracaso. Analizarla y aprender de ella debe ser una actividad obligada y sistemática (Hurtado de Mendoza Amat 2009).

Cuba es reconocida internacionalmente como ejemplo para otros países por los avances logrados en su sistema de salud. Dentro de las fortalezas de este sistema se encuentran: el desarrollo de la asistencia primaria al acercar al médico a la familia, los bajos índices de mortalidad infantil y materna, el alargamiento de la esperanza de vida, la erradicación y control de numerosas enfermedades, elevada instrucción médica y sanitaria de la población, el desarrollo de la industria biofarmacéutica y otros. Sin embargo, es poco reconocido su índice de autopsias ya que no se considera la importancia de las mismas.

En el país, el índice de autopsias es uno de los más elevados del mundo, cerca del 60 % en fallecidos hospitalarios durante los últimos 20 años (Hurtado de Mendoza Amat, Montero González, and Ygualada Correa 2013). Sin embargo, como consecuencia de lo anterior, la información recogida no siempre está disponible en un tiempo prudencial para la toma de decisiones, muchas veces ni siquiera se introduce en un sistema.

Dada la diversidad del campo de las autopsias, se clasifican en tres tipos: la autopsia judicial o médico-legal, la autopsia clínica y la autopsia perinatal(Salud 2020a) . La última hace referencia a el estudio médico o forense del cuerpo de un feto mayor de 22 semanas de gestación o de 500 g de peso, un niño nacido muerto o un niño nacido vivo y muere menor de 28 días de nacido (Mendoza 2019a).

La mortalidad perinatal es un indicador de los riesgos de muerte ligados a la reproducción y es importante porque permite tener reflejo de la atención prenatal, durante el parto y posparto, actúa como demarcador tanto de la calidad del servicio de salud materno infantil como de la condición nutricional de la madre y el entorno en que vive. El periodo perinatal, a pesar de su relativo corto tiempo de duración, tiene una influencia decisiva en la calidad de vida del individuo, en el desarrollo físico, neurológico y mental, condicionando fuertemente su futuro.

La Organización Mundial de la Salud (OMS) informa que la mortalidad ligada al embarazo y parto constituye más de la mitad de la mortalidad infantil (Ticona Rendón and Huanco Apaza 2011). La mortalidad perinatal es un problema de salud pública, su reducción es una de las metas de los Objetivos de Desarrollo del Milenio (ODM) (salud 2020b). Tanto los

nacidos muertos como las defunciones neonatales tempranas son difíciles de identificar y es por ello que el cálculo de este indicador es complicado por falta de datos precisos.

A pesar de las oportunidades y beneficios que pueden dar un eficiente aprovechamiento de la información generada por la autopsia perinatal, muchas veces es afectado por la insuficiente estructuración de los datos acopiados, incluso su olvido en archivos físicos no siempre bien conservados. En este sentido, la utilización ordenada y masiva de las tecnologías de la información y las comunicaciones (TIC) juega un papel esencial en la búsqueda del máximo aprovechamiento de la información de la autopsia.

En el año 2014 se creó un Sistema Informático de Registro y Control de Anatomía Patológica para la Red Hospitalaria de Cuba que contiene la gestión de autopsias sin importar la clasificación de su tipo. Sin embargo, por la diversidad anatómica y en procedimiento que existe entre los adultos y los fetos o recién nacidos se hace necesario hacer un análisis diferenciado de las autopsias perinatales de las realizadas a adultos y, por tanto, incluirla, de manera diferenciada. Dicha información, hasta el momento se procesa y almacena de forma manual, por lo cual la gestión actual de estos datos presenta un conjunto de limitaciones:

- Los datos a gestionar no siempre se encuentran disponibles al alcance de los especialistas en un momento dado.
- Se generan errores en el proceso de recogida de información que no son fáciles de corregir.
- La información contenida en papel corre el riesgo de pérdida o deterioro en el tiempo
- El proceso es lento.

En la búsqueda de una solución factible, se formula el siguiente **problema a resolver:** El sistema actual de registro y control en Anatomía Patológica afecta el análisis de la información generada por las autopsias perinatales realizadas en la red hospitalaria de Cuba.

En concordancia con lo anterior, se toma como **objeto de estudio:** La gestión de la información en los departamentos de Anatomía Patológica, centrando el **campo de acción** en el proceso de registro y control de autopsias perinatales en la red hospitalaria de Cuba.

INTRODUCCIÓN

En respuesta al problema, se define como **objetivo general:** desarrollar un módulo para el sistema informático para el registro y control en Anatomía Patológica que favorezca el análisis de la información generada por el proceso de autopsias perinatales.

. Lo anterior deriva en los siguientes **objetivos específicos:**

1. Formalizar el marco teórico conceptual de la investigación a partir de los referentes teóricos que garanticen el máximo aprovechamiento de los estudios patológicos de las autopsias perinatales.
2. Obtener el modelo de diseño del módulo de autopsias perinatales para el sistema de registro y control en Anatomía Patológica.
3. Implementar el módulo de autopsias perinatales para el sistema informático de registro y control en Anatomía Patológica a partir del modelo de diseño.
4. Obtener la validación de la solución propuesta mediante pruebas de calidad de software y pruebas de aceptación a expertos en Anatomía Patológica.

Para dar cumplimiento al objetivo, se deberán desarrollar las siguientes **tareas de investigación:**

Tareas de la investigación:

1. Revisión de la bibliografía para la actualización de tendencias actuales en estudios patológicos de autopsias perinatales.
2. Determinación de la metodología, herramientas y tecnologías a utilizar para el desarrollo del módulo del sistema.
3. Definición de las principales funcionalidades del módulo del sistema.
4. Obtención del modelo de diseño del módulo del sistema.
5. Obtención del modelo de datos del módulo del sistema.
6. Implementación de las funcionalidades del módulo del sistema.
7. Validación de la efectividad de la solución propuesta a partir de pruebas de aceptación con expertos en Anatomía Patológica.

Además, para asegurar el correcto desarrollo de la investigación, se deberá hacer uso de métodos científicos. Entre ellos, de corte teórico:

- **Análisis histórico-lógico:** para el estudio de la evolución y situación actual de la práctica de la autopsia perinatal a nivel mundial, y en Cuba de forma particular; así como la creciente necesidad de una gestión cada vez más eficiente de la información generada y las soluciones encontradas a este problema.
- **Analítico-sintético:** para un estudio crítico de la mayor cantidad de trabajos relacionados con el tema en cuestión, así como soluciones informáticas existentes a nivel nacional e internacional. Con ello se pretende identificar elementos relevantes, tendencias y tecnologías más comunes en el desarrollo del módulo de sistemas informáticos para la gestión y procesamiento de la información en Anatomía Patológica, lo cual servirá como punto de partida para la solución propuesta.
- **Modelación:** para un mejor entendimiento de los procesos a informatizar. Provee una abstracción de la realidad mediante la creación de determinados artefactos que facilitan la comunicación con el cliente, la mejor visualización del sistema y su posterior implementación.

Como método empírico:

- **La entrevista:** juega un papel esencial en el desarrollo del presente Trabajo de Diploma, específicamente la entrevista no estructurada, dada su flexibilidad y capacidad de obtener conocimiento de un tema directamente de los especialistas. El empleo de la entrevista permitirá identificar las necesidades reales de informatización en los departamentos de Anatomía Patológica en Cuba, así como el entendimiento de los procesos a informatizar. Para una mejor comprensión de este documento, a continuación, se describe brevemente cada uno de los capítulos en que se encuentra estructurado.

Capítulo 1. Fundamentación teórica: describe los principales conceptos asociados al dominio del problema, así como las relaciones que se establecen entre ellos. Expone un análisis de las soluciones existentes en el mercado, resaltando las tendencias y tecnologías más comunes en su desarrollo. Finalmente, se argumentan las bases del desarrollo del módulo en cuestión.

Capítulo 2. Análisis y Diseño: refleja cada uno de los pilares del desarrollo del módulo propuesto, la especificación de sus funcionalidades, la arquitectura, los modelos de diseño y de datos, así como el uso de patrones.

INTRODUCCIÓN

Capítulo 3. Implementación y Pruebas: refiere la estrategia de pruebas definida a partir de la metodología de desarrollo, con el objetivo de garantizar la validación de la solución propuesta. Se analizan los niveles de pruebas ejecutados y los resultados alcanzados. Finalmente, se presentan evidencias de cómo la solución propuesta puede contribuir al análisis de los datos recogidos de las autopsias perinatales.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Capítulo 1: Fundamentación Teórica

El presente capítulo recoge los elementos teóricos que fundamentan el desarrollo de un módulo de autopsias perinatales para el Sistema Informático de Registro y Control en Anatomía Patológica de la Red Hospitalaria de Cuba. Se realiza un análisis de la situación actual y perspectivas de la autopsia perinatales en Cuba y el mundo, así como la importancia de una eficiente gestión de la información que esta genera. Además, se reflejan las características esenciales de las diferentes soluciones informáticas existentes en el mercado; dejando claro, de manera general, las tendencias actuales en el desarrollo de este tipo de sistemas. Finalmente, se exponen los elementos más relevantes de la metodología, lenguajes y herramientas definidas para el desarrollo.

1.1 Descripción de objeto de estudio

La Patología de las ciencias naturales que se dedica al estudio de las enfermedades, a su vez, la Anatomía Patológica es la rama de la Patología que estudia las alteraciones morfológicas, macro y microscópicas, que producen las enfermedades en las células, tejidos u órganos (Mendoza 2019b).

El método fundamental de la Anatomía Patológica es la autopsia. Permanente enriquecedora de la calidad en la medicina, constituye el estudio más completo del enfermo y la enfermedad. Las autopsias pueden ser de dos tipos: judiciales o médico-legales y clínicas. Las autopsias clínicas provienen de:

a) Pacientes ingresados en el hospital. Son las autopsias clínicas hospitalarias.

b) Pacientes no ingresados en el hospital. Son las autopsias clínicas extrahospitalarias. Dentro de esta categoría se encuentran las autopsias que provienen de Hospitalización Domiciliaria, de los propios domicilios remitidas por médicos de atención, otros centros u hospitales. Con respecto a los casos de los fallecidos en el Servicio de Urgencias, el típico paciente que ingresa “cadáver”, la mayoría de las veces va a ser subsidiario de autopsia judicial, específicamente cuando el paciente carece de antecedentes patológicos que justifiquen el desenlace. El Colegio de Patólogos Americano (CAP) también incluye dentro del ámbito médico legal a los fallecidos en las primeras 24 horas de acudir al hospital, de manera que el límite de tiempo para considerar una autopsia como judicial o clínica se establece habitualmente en las 24 horas; menos de ese tiempo se considera una autopsia judicial más que una autopsia clínica.

La autopsia perinatal hace referencia a la practicada a un feto muerto en la fase fetal intermedia (22 semanas) y en la fase fetal tardía (28 días de nacido) (Mendoza 2019a). La tasa de mortalidad perinatal es un indicador importante de la salud pública. Su cálculo y la observación de su evolución en el tiempo permite saber la efectividad de las estrategias sanitarias enfocadas en su reducción,

además de posibilitar comparativas entre regiones y países. Tener datos fiables permite identificar grupos sociales con mayor riesgo de sufrir la pérdida o muerte durante el embarazo o durante el periodo perinatal. Además, en combinación con la investigación de la causa de la muerte, facilita el desarrollo de estrategias de prevención clínicas y sociales. Evitar la muerte perinatal es importante porque tiene impactos psicológicos y sociales para las familias y los profesionales sanitarios, además de costes económicos sustanciales.

El examen postmortem en caso de muerte perinatal es esencial y de gran trascendencia para la información a los padres, el consejo genético y la planificación y cuidado del siguiente embarazo. La autopsia perinatal es un instrumento diagnóstico que reafirma a los obstetras y pediatras en su práctica clínica y que les sirve, utilizada como control de calidad, para modificarla y mejorarla. Este estudio puede influir en la detección de defectos congénitos para confirmar y hacer más completo los diagnósticos antenatales del siguiente embarazo.

Por la trascendencia de estos resultados para el consejo genético, se realizará un informe por escrito resultado de la autopsia en un plazo no superior a un mes, salvo que la misma requiera estudios especiales. Dicho informe deberá constar de un breve resumen de historia, datos macroscópicos (peso, talla, sexo), presencia o no de malformaciones, hallazgos fundamentales de la disección y descripción de hallazgos fundamentales histológicos.

1.2 Situación actual y perspectivas de la autopsia en Cuba y en el mundo

Las autopsias perinatales se dividen en autopsias fetales (realizadas a fetos) y autopsias neonatales (realizadas a recién nacidos). En Cuba no hay obstáculos legales para la realización de autopsias perinatales en la etapa fetal, solo en las de fallecidos neonatales se requieren el permiso expreso de los padres. No obstante, en su conjunto, el índice de autopsias es superior al 80% debido al gran interés de los centros hospitalarios en su realización y a una gran comprensión de los padres de la importancia de la misma. Uno de los objetivos de la autopsia es satisfacer la legítima duda de los padres en cuanto al desenlace desfavorable y otro es informar al médico de asistencia. También es objetivo, actualizar las estadísticas vitales de la institución y el país, sin olvidar su valor como fuente de docencia e investigación y en último término, y no por ello menos importante, constituirse en instrumento de control de calidad de los servicios.

No es posible olvidar que una autopsia puede confirmar un diagnóstico, añadir diagnósticos, cambiar diagnósticos o no ser concluyente. Todo esto queda evaluado en el seno de los Comités de Análisis de la Mortalidad Intrahospitalaria que forman parte de los Comités de Calidad de la Atención. Pero, por supuesto para que la autopsia pueda colmar estas expectativas y ser confiable

debe ser realizada con un estándar mínimo de calidad y es notable que aún en países desarrollados haya hasta un 47% de autopsias inadecuadas por ser realizada por patólogos sin experiencia en Patología Perinatal.

Sin embargo, diferentes autores como: (Hurtado de Mendoza Amat and Álvarez Santana 2008; Hurtado de Mendoza Amat 2009; Hurtado de Mendoza Amat, Montero González, and Ygualada Correa 2013), refieren un alarmante descenso del Índice de Autopsias durante las últimas décadas en países escandinavos, de Europa del Este, España, Reino Unido, Francia, India, China, Japón, Australia, Canadá y Estados Unidos (EE.UU); todos por debajo del 15%. En la mayoría de los países subdesarrollados las propias limitaciones de su sistema de salud pública los coloca en una posición casi excluyente(Hurtado de Mendoza Amat, Álvarez Santana, and Borrajero Martínez 2008).

En 2008 se publicó el resultado de una encuesta (Salud 2020a)[1] realizada a 59 patólogos de 21 países asistentes al XXV Congreso Latinoamericano de Anatomía Patológica celebrado en La Habana, en octubre de 2005. Los resultados muestran que, más de la mitad de los índices de autopsias declarados fueron del 5 % o menos y solo 2 patólogos costarricenses refirieron 20 % o más. En Colombia en el año 2013 se analizaron características morfológicas durante autopsias fetales, así como caracterizaciones de la placenta y hallazgos fenotípicos de importancia para diagnósticos correctos. España es uno de los países con más avances prácticos en la realización de autopsias perinatales pues su protocolo contiene la realización de diferentes tipos de exámenes: examen externo, fotografías, radiografía, examen interno y examen histológico.

En países como Chile se han realizado estudios estadísticos en los casos de muerte intrauterina. El análisis conjunto de los datos clínicos maternos con la autopsia perinatal y el estudio de la placenta, permiten aclarar la causa de la mayoría de las muertes perinatales. De este modo los padres se informan y orientan adecuadamente y en un futuro embarazo se pueden tomar las medidas necesarias para evitar una muerte intrauterina. Para ello se analizaron retrospectivamente 299 muertes fetales ocurridas entre las 22 y 42 semanas de gestación en un período de 5 años. Se incluyeron 279 casos con estudio histopatológico de la placenta y autopsia fetal logrando identificarse las principales causas primarias de muerte perinatal.

En EE UU se descubrió que la autopsia y el examen histológico de la placenta son útiles en la confirmación o exclusión de causas de muerte en el 42,4% y el 64,6% de casos, respectivamente, pero que su utilidad alcanza más del 90% de los casos en varios contextos clínicos. Otro estudio

[1] **Encuesta realizada a 59 patólogos de 21 países en 2008:** Argentina, Austria, Brasil, Canadá, Colombia, Costa Rica, Chile, Ecuador, España, Guatemala, Honduras, Italia, México, Nicaragua, Países Bajos, Panamá, Paraguay, Perú, República Dominicana, Suiza y Venezuela.

que empleaba un sistema de análisis escalonado (evaluación clínica y pruebas de laboratorio estándar, seguidas por un examen histológico de la placenta y, finalmente, la autopsia) encontró una causa de muerte en el 74,3% de los casos. Aún más llamativo fue el hallazgo de que la combinación de la autopsia junto con el examen de la placenta cambió la gestión clínica en casi la mitad de los casos (Ovalle S. et al. 2005).

En un estudio de (Park et al. 2013), que incluía casos de menos de 20 semanas, la autopsia reveló resultados nuevos en el 37% de los casos y en, aproximadamente, una tercera parte de los casos dio lugar a un cambio en el asesoramiento genético. Pero en la mayoría de los países los estudios avanzan muy lentamente y es los datos no se aprovechan por la inexistencia de un sistema automatizado. En un artículo de una revista de patología, publicado por un numero colectivo internacional de autores, se reconoce el trabajo realizado por Cuba en la incorporación de la informática en la patología, resaltando particularmente el uso del SARCAP y las investigaciones que de él se han derivado.

A pesar de las dificultades, Cuba cuenta con las condiciones propicias para mantener e incrementa la posición cimera alcanzada en el mundo. Mucho puede y debe hacerse para refirmar esta fortaleza del sistema de salud nacional (Jardines Méndez 1995). Es necesario profundizar en la situación actual y las perspectivas para elevar los índices de autopsias perinatales. Para mejorar la calidad y lograr el mayor aprovechamiento de la autopsia, el Dr. Cs. José Hurtado de Mendoza Amat concede un importante papel a la necesidad de ampliar y consolidar el Registro Nacional de Autopsias a partir de la explotación eficiente de un Sistema Automatizado de Registro y Control en Anatomía Patológica (Hurtado de Mendoza Amat 2009; Hurtado de Mendoza Amat, Montero González, and Ygualada Correa 2013), una vez que la información de muchas de las autopsias que se realizan hoy, no puede ser procesada porque no es introducida en el sistema, y en consecuencia, se desaprovecha.

1.3 La gestión de la información en las autopsias perinatales en Cuba

El examen postmortem en caso de muerte perinatal es esencial y de gran trascendencia para la información a los padres, el consejo genético y la planificación y cuidado del siguiente embarazo. La autopsia perinatal es un instrumento diagnóstico que reafirma a los obstetras y pediatras en su práctica clínica y que les sirve, utilizada como control de calidad, para modificarla y mejorarla.

La información recogida durante la autopsia se divide en dos: diagnóstico macroscópico y diagnóstico microscópico. Durante el primero se recogen los datos que describen el estado de cada uno de los sistemas y órganos del cuerpo y en el segundo, se recogen los datos de los procesos

que dieron lugar a la muerte, basado en un análisis del diagnóstico macro y, finalmente, se obtiene como resultado la causa de muerte. Todos los datos almacenados se rigen por un protocolo que unifica las formas de trabajo de los diferentes servicios de Anatomía Patológica al examen postmortem perinatal, el cual es mostrado en los Anexos del 1 al 6.

Todos esos datos se recogen de forma manual lo cual dificulta el acceso a la información desde cualquier lugar remoto a las oficinas del patólogo, se hace tedioso y lento la búsqueda de un caso específico, así como modificar o comparar los casos existentes. Por tanto, la información no siempre está disponible cuando se necesita, y los documentos pueden perderse, romperse o deteriorase con el tiempo, por lo cual se hace necesario la existencia de un sistema informático para registrar y gestionar esta información.

1.4 Análisis de las soluciones existentes en el mundo

En la actualidad, existen una gran cifra de empresas que se dedican al desarrollo de sistemas informáticos para la gestión de información hospitalarias (HIS, por sus siglas en inglés). Diversas soluciones están disponibles en el mercado, unas más usuales y otras para departamentos específicos. Los sistemas de información son herramientas básicas que dan respuesta a la labor asistencial de los servicios de anatomía patológica, que va dirigida a la elaboración de diagnósticos, evaluaciones pronosticadas y selección de dianas terapéuticas. Además, sirven de apoyo para las funciones docentes, investigadoras y de calidad de los servicios de anatomía patológica.

Los dos países que resaltan en el desarrollo y comercialización de sistemas de información en Anatomía Patológica son: España y Estados Unidos. Una encuesta referida en el “Libro Blanco de Anatomía Patológica en España 2019” revela que en este país el 98,7% de los centros tienen un programa informático específico para Anatomía Patológica. El sistema más ampliamente implantado es Patwin, en sus diferentes versiones, seguido de Vitropath y Gespath y el grado de satisfacción mayor es para (Patwin 4.9), seguido de Gespath y Novopath (Isabel 2019).

Por otro lado, en Estados Unidos las empresas que han distribuido más números de contratos para sitios que operan los sistemas de anatomía patológica los cuales tienen los siguientes nombres: Easy (596), Cerner Corp. (177), Computer Trust Corp. (106), Epic (123), Novopath (245), Psyche Systems Corp. (107), SCC Soft Computer (139), Sunquest (357), Technidata (150+), VitalAxis (112) (Cap 2019).

A continuación, se muestra una comparación de los sistemas de gestión de información para anatomía patológica antes mencionados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Tabla 1 Resumen de las características más importantes de los sistemas más conocidos con soporte para la gestión de Anatomía Patológica a nivel internacional.

Nombre	Contienen la gestión de autopsias perinatales	Plataforma window	Aplicación de escritorio	Codificación CIE 10	Software Libre
Patwin	X	X	X		
Vitropath		X	X	X	
Novopath	X	X		X	
Gespath	X	X			
Cerner CoPathPlus	X	X			
AP Easy		X		X	
WinSURGE		X			
Beaker Anatomic Pathology	X	X			
NovoPath	X	X			
WindoPath E.ssential		X			
SoftPathDx	X	X			
Sunquest CoPathPlus		X	X		
Sunquest PowerPath		X	X		
TD-Histo/Cyto		X			
VitalDx LIS		X		X	

De manera general, los datos evidencian un predominio de software privativo en el mercado internacional, el uso de SNOMED CT como sistema de codificación, arquitectura cliente servidor y dependencia de la plataforma Windows. Lo anterior supone el pago de costosas licencias para su adquisición y actualización, sin contar las limitaciones de Cuba para el uso de SNOMED y la contrariedad con las políticas cubanas de migración a software libre que representa el propio sistema y su dependencia del sistema operativo Windows.

Por lo regular, las empresas no venden programas informáticos, los “licencian”. El comprador de un producto no adquiere el derecho de utilizarlo como guste, al adquirir una licencia, por el contrario, solo se paga por utilizarlo según las condiciones fijadas por el productor. Es común entonces que la solución a los problemas sea actualizar el software a una nueva versión, teniendo que pagar por ello; e incluso, la interrupción en la asistencia técnica a productos “antiguos” según su proveedor.

Además, para garantizar el acceso y la adecuada utilización de los datos, es necesario que su codificación no esté ligada bajo licencia propietaria a un proveedor en exclusiva. No es cuestión de la titularidad de un estándar, sino de su disponibilidad para terceros, como en el caso de SNOMED CT.

1.4.1 Situación en Cuba

En el año 1985 se desarrolló el Sistema Automatizado de Registro y Control en Anatomía Patológica **(SARCAP),** en el hospital “Dr. Luis Díaz Soto”. Posteriormente aplicado en otros departamentos de Anatomía Patológica del país, con el objetivo de registrar y controlar de forma automatizada la información obtenida de las biopsias y autopsias, para facilitar su procesamiento y recuperación con fines docentes, científicos, asistenciales y administrativos.

Desarrollado sobre un ambiente MS-DOS y desatendido desde el año 2002, el sistema posee un conjunto de limitaciones que se considera necesario resolver, para lo cual es preciso rehacerlo completamente. Dadas sus características, el SARCAP no ha podido ser actualizado a la última revisión de la Clasificación Internacional de Enfermedades (CIE-10) de la Organización Mundial de la Salud. La codificación actual de los diagnósticos resulta redundante en algunos casos e insuficiente en otros para lograr una especificación detallada de la enfermedad, especialmente cuando resulta conveniente complementarla con los ejes morfológico y topográfico, así como el uso de comodines (Pedro Arango 2014a).

El mismo, restringe la cantidad de causas indirectas y contribuyentes a la muerte, pasando por alto la posibilidad de recoger el resumen de historia clínica del fallecido, la descripción de los hábitos externos e internos (órganos por aparato o sistema), el peso y medida de los órganos. El SARCAP carece de un subsistema adecuado para el procesamiento de la información de autopsias perinatales (incluidas las neonatales) que por sus características reciben un tratamiento diferenciado de las autopsias de adultos y pediatría (Hurtado de Mendoza Amat 2009).

En 2004 se dio a conocer **AvanPat 1.1** en el VI Congreso Virtual Hispanoamericano de Anatomía Patológica. Surge como una respuesta a la necesidad que el desarrollo creciente de la medicina cubana, en áreas de investigación como la anatomía patológica. El sistema almacena, organiza y gestiona la información relacionada con los estudios que se realizan en los departamentos de Anatomía Patológica, cuenta con un conjunto de plantillas prediseñadas para estudios (citologías de líquidos, vaginales y punción aspirativa con aguja fina, biopsia y necropsias) (Medell Gago 2004).

Posterior a su publicación, no existen referencias de la utilización del sistema. No se le dio seguimiento y no se tienen noticias de su empleo en centros de salud cubanos para la introducción de los datos de estudios anatomopatológicos realizados. En consecuencia, no ha recibido la validación de los especialistas en la práctica.

En el 2011 en la Universidad de las Ciencias Informáticas, desarrolló el Sistema de Información Hospitalaria **Alas HIS**, está concebido como una aplicación web a ser desplegada en cada centro

de salud, con una base de datos local encargada del almacenamiento y reporte de los datos generados por los distintos módulos instalados. Entre sus principales funcionalidades del Alas se encuentran:

- Gestión de protocolos de autopsias (descripción macroscópica y microscópica).
- Gestión de biopsias y citologías (descripción macroscópica y microscópica).
- Gestión de solicitudes de análisis.
- Gestión de muestras para análisis de biopsias y citologías.
- Seguimiento de las muestras durante todo su procesamiento.
- Registro de datos de entrada y salida del cadáver a la morgue.
- Visualización de la cantidad de análisis indicados por servicio o la cantidad de pacientes atendidos de un servicio dado, en un rango de fecha seleccionado.

Alas HIS ha venido ganando un prestigio internacional en los últimos tiempos, logrando ser desplegado en diferentes centros hospitalarios venezolanos. Sin embargo, la concepción integral del sistema y sus características particulares, hacen pensar que el mismo fue concebido para su exportación y no para ser desplegado en Cuba, cuyas deficiencias en la conectividad interdepartamental de los centros de salud y el alto grado de obsolescencia tecnológica hacen la solución no viable en la mayoría de las instituciones hospitalarias de la Isla. Hasta el momento, el sistema no ha podido ser desplegado en ningún centro de salud cubano.

Con la motivación de resolver los problemas presentes en el sistema SARCAP surge en el 2014 la primera versión de el Sistema Informático de Registro y Control en Anatomía Patológica para la Red Hospitalaria de Cuba (**SIRCAP**), creado sobre la base del original, con las tecnologías actuales y la tutoría del propio autor del SARCAP. Este sistema incluye un subsistema de autopsias para adultos. En los años siguientes ha continuado el desarrollo del SIRCAP, llegando a ser probado con éxito en el Hospital Clínico Quirúrgico "Hermanos Ameijeiras" de La Habana. Para una mejor especificación de los diagnósticos, el sistema promueve su codificación según los más recientes estándares de terminología impulsados por la Organización Mundial de la Salud.

La solución desarrollada, facilita el proceso de evaluación de la calidad en la atención médica, a partir del análisis y clasificación de la correlación clinicopatológica. Posibilita la configuración y aplicación reglas de control interno sobre los diagnósticos contribuyendo a elevar la calidad de los datos almacenados. Permite la creación de un amplio conjunto de reportes estadísticos sobre los datos, los cuales podrán ser empleados como soporte para la toma de decisiones, además de

contribuir con el descubrimiento y explicación de nuevas enfermedades, manifestaciones inusuales de enfermedades conocidas y complicaciones terapéuticas.(Pedro Arango 2014b)

De igual forma puede realizarse un control de calidad sobre los datos introducidos con el objetivo de garantizar el cumplimiento de la metodología establecida y la mayor calidad de la información. El sistema a partir de las reglas configuradas permite realizar correcciones automáticas a la información almacenada o mostrará los casos que deben ser revisados por los especialistas. (Pedro Arango 2014b)

Como resultado del análisis de los sistemas cubanos, en la Tabla 2 se muestra un resumen comparativo de sus principales características.

Tabla 2 Resumen de las características más importantes de los sistemas más conocidos con soporte para la gestión de Anatomía Patológica en Cuba.

Nombre	**Contienen la gestión de autopsias perinatales**	**Plataforma window**	**Aplicación de escritorio**	**Codificación CIE 10**	**Software Libre**
SARCAP		X	X	X	X
AvanPat		X	X	X	X
Alas HIS (Módulo AP)		X		X	
SIRCAP		X	X	X	X

Los sistemas anteriormente mencionados, a pesar de contar con una amplia diversidad de funcionalidades no incluyen la especialidad de autopsias perinatales. Por lo tanto, se puede concluir que no existen antecedentes específicos en Cuba de sistemas que trabajen las autopsias perinatales.

Ninguna de las soluciones existentes en el ámbito de los Sistemas de Información para Anatomía Patológica de Cuba gestiona la información de las autopsias perinatales, sin embargo, cada una tiene algo que aportar a la confección de una nueva propuesta que se adapte a las condiciones y necesidades de los departamentos de Anatomía Patológica en el país, de ahí la importancia de su análisis. El actual Sistema de Registro y Control de Anatomía Patología reúne las demandas tecnológicas necesarias para un sistema de salud cubano y forma parte del área de anatomía patológica, por tanto, la opción más apropiada para la gestión de la información perinatal, sería la inclusión de un módulo en el ya existente sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.5 Metodología, lenguajes y herramientas de desarrollo

El análisis de los elementos abordados en cada uno de los epígrafes anteriores, arrojó los fundamentos del conjunto de decisiones relacionadas con el desarrollo del módulo en cuestión, entre ellas, la elección de la metodología, lenguajes y herramientas de desarrollo. El módulo a desarrollar forma parte del sistema SICAP (ya que este no gestiona la información de las autopsias perinatales), y que por tanto tiene que haber correspondencia entre la línea base de desarrollo de la propuesta de solución con el del sistema al que se integrará.

1.5.1 Metodología de desarrollo

El proceso de desarrollo de software, es definido como el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software, tiene como finalidad la obtención de un producto que cumpla con las expectativas del cliente. En el logro de ese objetivo, cobra vital importancia la elección de la metodología de desarrollo apropiada. La misma, guía el proceso de desarrollo para alcanzar la satisfacción del cliente y del equipo. La elección de la metodología no es un proceso trivial, en ocasiones se torna una tarea bien difícil de llevar a cabo, debe elegirse aquella que mejor se ajuste a las características del equipo de desarrollo y las exigencias de los usuarios finales (Jacobson, Booch, and Rumbaugh 2000).

Las metodologías de desarrollo se pueden dividir en dos grandes grupos:

- **Metodologías pesadas:** se centran en la definición minuciosa de los procesos y tareas que se realizarán durante el ciclo de vida del software, generando una extensa documentación asociada a cada elemento del proceso de desarrollo (Charvat 2003).

  El **Proceso Unificado de Desarrollo** *(Rational Unified Process, RUP)* resulta el exponente por excelencia entre las metodologías pesadas. Es un marco de trabajo genérico que puede especializarse para una gran cantidad de sistemas de software, para diferentes áreas de aplicación y diferentes tipos de organizaciones; se basa en la construcción de componentes de software interconectados a través de interfaces bien definidas (Jacobson, Booch, and Rumbaugh 2000).

  El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes, estos constan de cuatro fases: inicio, elaboración, construcción y transición. Cada fase se divide en iteraciones. Cada ciclo produce una nueva versión del sistema y cada versión es un producto preparado para su entrega. El producto final incluye los requisitos, casos de uso,

especificaciones no funcionales, casos de pruebas, el modelo de arquitectura y el modelo visual (Booch, Rumbaugh, and Jacobson 1999).

- **Metodologías ágiles:** esbozan los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y responder a los cambios que puedan surgir a lo largo del proyecto, estas ofrecen una alternativa a los procesos de desarrollo de software tradicionales (Letelier and Penadé 2003).

  La **Programación Extrema** *(e**X**treme **P**rogramming, XP)* tal vez sea la más conocida y más extensamente usada de las metodologías ágiles, en sus inicios desafió numerosos dogmas convencionales asociados al desarrollo de software (Beck and Andres 2004). En XP todos los requisitos son expresados como escenarios (historias de usuarios), los cuales son implementados directamente como una serie de tareas. Esta metodología resulta apropiada para proyectos con requisitos imprecisos y cambiantes, en los que existe un alto riesgo técnico. Los programadores trabajan en parejas y se desarrollan pruebas para cada una de las tareas. Todas las pruebas deben ser ejecutadas satisfactoriamente cuando el nuevo código es integrado al sistema (Sommerville 2007).

Ambas metodologías persiguen el mismo fin, producir software con los estándares de calidad requeridos; aunque para ello, se sirven de herramientas y métodos diferentes.

RUP brinda numerosas ventajas, sin embargo, se torna rígida a la hora de introducir cambios a mediado del proyecto, cuestión a tener presente una vez que los flujos dentro del proceso de desarrollo no están bien definidos. Además, esta metodología genera una gran cantidad de artefactos que podrían suponer un atraso dada la reducida composición del equipo de desarrollo, así como el escaso tiempo de que se dispone para ejecutar todo el proceso.

La evaluación de las metodologías y la necesidad de una variante flexible a los cambios, preparada para ser ejecutada por un equipo pequeño y que genere los artefactos mínimos para la comunicación con el cliente, permitió identificar la **Programación Extrema (XP)** como la alternativa más acertada.

1.5.2 Lenguaje modelado

Un lenguaje de modelado provee un vocabulario y conjunto de reglas centradas en la representación conceptual y física de un sistema (Booch, Rumbaugh, and Jacobson 1999). El mismo es un lenguaje estándar para escribir planos de software, compuesto por diferentes elementos gráficos que se combinan para conformar los diagramas. Permite a los creadores de sistemas generar diseños que

capturen sus ideas en una forma fácil de comprender y comunicar a otras personas (Booch, Rumbaugh, and Jacobson 1999; Schmuller 2000).

La elección de **Lenguaje Unificado de Modelado** *(Unified Modeling Language* UML) se sustenta en su facilidad para especificar, documentar, visualizar y comprender los artefactos generados en las diferentes fases del desarrollo de software. Así como sus ventajas para la comunicación entre el cliente y el equipo de desarrollo.

1.5.3 Ingeniería de software asistida por computadora

Para (Sommerville 2007), ingeniería de software por computadora (Computer Aided Software Engineering, CASE) es el nombre con el que se identifica la herramienta utilizada para apoyar las actividades del proceso de software. Estas herramientas facilitan la interacción entre los miembros de un equipo al hacer que la diagramación sea un proceso iterativo y dinámico. Los analistas de sistemas se apoyan en estas herramientas desde el principio hasta el fin del ciclo de vida, para incrementar la productividad, comunicarse de manera eficiente con los usuarios y desarrolladores, e integrar el trabajo que desempeñan en el sistema (Kendall and Kendall 2005).

Visual Paradigm es una herramienta CASE multiplataforma que contribuye al desarrollo de sistemas de software fiables, mediante un enfoque orientado a objetos. Su preferencia está determinada por las oportunidades que ofrece para la construcción de aplicaciones de calidad, con mayor rapidez y menor costo. Soporta el ciclo completo de desarrollo de software y permite su documentación en diferentes formatos, empleando UML como lenguaje de modelado. Posibilita la generación de código en una amplia gama de lenguajes de programación y se integra con los principales Entornos de Desarrollo Integrado (IDE) presentes en el mercado, especialmente el lenguaje Java y su IDE NetBeans (Paradigm 2005).

1.5.4 Lenguaje de programación

De acuerdo con las valoraciones de (Trejos Buritica 1999) un lenguaje de programación es un conjunto de instrucciones entendibles y ejecutables por una computadora. Al profundizar sobre este aspecto, se considera que el lenguaje de programación deberá facilitar a los desarrolladores la confección de programas de manera concisa y rápida. En su evaluación, se debe tener en cuenta el soporte a cada parte del ciclo de vida del software y debido a la necesidad de mejora y mantenimiento en el período de explotación, debe ayudar a otros programadores a comprender el funcionamiento de la solución (Mitchell 2002).

La elección del lenguaje de programación está determinada por la necesidad de una herramienta que se adapte a las condiciones de la red hospitalaria cubana. La solución deberá funcionar de

forma centralizada, en un entorno que garantice la disponibilidad de la información generada en cada uno de los hospitales, al tiempo que estará disponible de forma local ante la ausencia de conexión, manteniendo las capacidades de despliegue instantáneo y conectividad, desvinculadas del acceso a la aplicación.

Las afirmaciones de (ORACLE 2020) apuntan que **JavaFX** es una tecnología que ofrece un modelo de desarrollo y despliegue unificado para la creación de Aplicaciones de Internet Enriquecidas (Rich Internet Applications, RIA), esto es, aplicaciones web con las características y capacidades de aplicaciones de escritorio.

Las aplicaciones JavaFX son completamente desarrolladas en Java, una de las tecnologías más ampliamente desplegadas con una de las más grandes comunidades de desarrolladores en el mundo. Estas se valen de beneficios de la plataforma tales como la programación orientada a objetos, herencia, polimorfismo y manejo de excepciones; al tiempo que aprovecha la potencia de las prácticas de programación basadas en patrones de diseño. JavaFX cuenta con su propio lenguaje declarativo JavaFX Script, al igual que un conjunto de herramientas de diseño y desarrollo, lo cual permite crear contenido expresivo, dinámico y funcional, combinando las mejores capacidades de la plataforma Java con funcionalidades de multimedia interactivas (texto, imagen, audio y video). Es una plataforma de código abierto, de libre distribución y fácil acceso, integrada en las versiones más recientes del popular IDE NetBeans (ORACLE 2020; PremKumar and Mohan 2010).

1.5.5 Entorno de desarrollo integrado

Las observaciones de (Bell and Parr 2003), (Matellán Olivera 2004) y (Rouse 2007) señalan que un entorno de desarrollo integrado es un programa compuesto por un conjunto de herramientas que proveen facilidades a los programadores para agilizar el proceso de desarrollo de software. Consta de un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

De acuerdo con (ORACLE 2013), NetBeans IDE es un proyecto exitoso de código abierto con una gran cantidad de usuarios y una comunidad en constante desarrollo. Proporciona un amplio soporte para las últimas tecnologías de Java, incluyendo JavaFX. Soporta integración con diferentes herramientas para el control de versiones y Sistemas Gestores de Bases de Datos (SGBD). Es un producto libre, gratuito y sin restricciones de uso. Su empleo está determinado por la experiencia del equipo de desarrollo en su utilización, en virtud de minimizar el tiempo necesario para la familiarización con nuevos lenguajes y herramientas de desarrollo.

1.5.6 Control de versiones

Git es un software de control de versiones diseñado por Linus Tolvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadoras y coordinar el trabajo que varias personas realizan sobre los archivos compartidos (Cumare 2020).

Posee control de versiones distribuidos que se encargan de manejar numerosos repositorios remotos con los cuales pueden trabajar, de tal forma que puedes colaborar simultáneamente con diferentes grupos de personas en distintas maneras dentro del mismo proyecto. Esto permite establecer varios flujos de trabajo que no son posibles en sistemas centralizados, como pueden ser los modelos jerárquicos (Git 2020).

1.5.7 Prototipos e interfaces de usuario

JavaFX Scene Builder es una herramienta de diseño visual que permite a los usuarios diseñar rápidamente interfaces de usuario de aplicaciones JavaFX, sin codificación. Los usuarios pueden arrastrar y soltar componentes de la interfaz de usuario a un área de trabajo, modificar sus propiedades, aplicar hojas de estilo y el código FXML para el diseño que se está creando se genera automáticamente en segundo plano. El resultado es un archivo FXML que luego se puede combinar con un proyecto Java al vincular la interfaz de usuario a la lógica de la aplicación (Oracle 2019).

1.5.8 Herramientas para la generación de reportes

Tener reportes personalizados es de gran ayuda ya que permite al usuario validar la entrega exitosa de servicios existentes ante el cliente e identificar oportunidades para mejorar la disponibilidad y rendimiento de: dispositivos sistemas operativos, aplicaciones de software y servicios existentes. Estos hallazgos pueden llevar a la necesidad del reemplazo o la reconfiguración de los sistemas por parte del proveedor de servicios subcontratado.

JasperReports es una biblioteca de creación de informes que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML. Está escrito completamente en java y puede ser usado en gran variedad de aplicaciones de java incluyendo J2EE o aplicaciones web, para generar contenido dinámico. Se ha desarrollado un subproyecto que es un servidor integrado para informes: JasperR (Heffelfinger 2006).

1.5.9 Sistema gestor de base de datos

PgAdmin III es un sistema integral de diseño y gestión de bases de datos PostgreSQL para sistemas Unix y Windows. Está disponible gratuitamente bajo los términos de The PostgreSQL License y se puede redistribuir siempre que se cumplan los términos de la licencia. El proyecto está gestionado por el equipo de desarrollo de PgAdmin.

Este software fue escrito como un sucesor de los productos originales pgAdmin y pgAdmin II, que, aunque populares, tenían limitaciones en el diseño. PgAdmin III está escrito en C y utiliza el kit de herramientas multiplataforma wxWidgets (anteriormente wxWindows). La conexión a PostgreSQL se realiza utilizando la biblioteca libpq nativa.

PGadmin es la plataforma de administración y desarrollo de código abierto más popular y con más funciones para PostgreSQL, la base de datos de código abierto más avanzada del mundo. La aplicación se puede usar en plataformas Linux, FreeBSD, Solaris, macOS y Windows para administrar PostgreSQL 8.4 a 9.5 que se ejecuta en cualquier plataforma, así como versiones comerciales y derivadas de PostgreSQL como EDB Postgres Advanced Server (Pgadmin 2019).

1.5.10 Mapeo objeto-relacional

Object-Relational mapping (ORM), o lo que es lo mismo, mapeo de objeto-relacional, es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador.

Hibernate es una herramienta de ORM para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. En otras palabras, Hibernate es un Framework que agiliza la relación entre la aplicación y la base de datos. Para poder aprender a utilizarlo es necesario contar con los conocimientos básicos de base de datos y SQL así como manejar el lenguaje Java (Ottinger 2016).

1.6 Patrones para el desarrollo del software

Los patrones de software son la estrategia perfecta para la resolución de distintos problemas en la ingeniería de software debido a que un patrón logra expresar una relación entre un contexto, un problema y una solución de tal forma que esa solución pueda ser usada nuevamente. Esto indica que un patrón de software es un punto de partida para resolver varios inconvenientes al momento de desarrollar software, lo cual indica que son útiles al momento de generar software de alta calidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Según (Alexander 1979) un patrón es algo que describe un problema que ocurre una y otra vez en el entorno propio, así como la solución a ese problema, de tal modo se puede aplicar esta solución nuevamente, sin hacer lo mismo dos veces. En (Erich Gamma and Wesley 1994) se argumenta que, aunque el reconocido arquitecto Alexander se refería a patrones de edificios y ciudades, este concepto también se aplica para el desarrollo de software.

Teniendo en cuenta lo expuesto anteriormente, se puede decir que los patrones de software promueven altamente la reutilización de código, diseño y de la arquitectura utilizada. Estos reúnen la experiencia que han tenido ingenieros de software experimentados que han tenido éxito en sus planteamientos, recolectando así en los patrones las buenas prácticas de diseño. Las características de un buen patrón son:

1. **Resuelve un problema**: Los patrones capturan soluciones, no principios o estrategias abstractas.
2. **Es un concepto probado**: Capturan soluciones, no teorías o especulaciones. En el caso del “Design Patterns”, el criterio para decidir si algo era un patrón o no, era que éste debía tener al menos 3 implementaciones reales.
3. **La solución no es obvia:** Muchas técnicas de resolución de problemas (como los paradigmas o métodos de diseño de software) intentan derivar soluciones desde principios básicos. Los mejores patrones generan una solución a un problema indirectamente (un enfoque necesario para los problemas de diseño más difíciles).
4. **Describe una relación**: Los patrones no describen módulos sino estructuras y mecanismos.
5. **Tiene un componente humano significante**: El software sirve a las personas. Los mejores patrones aplican a la estética y a las utilidades.

Cada patrón de software tiene una estructura propia que lo define, y lo ubica en la complejidad que maneja, cada uno con una finalidad distinta. El desarrollador de software si quiere hacer uso de los patrones de software debe escoger cual se adapta más a sus necesidades y cual proporciona mejores soluciones y en menor costo.

Los patrones usados para el desarrollo del módulo a implementar son los siguientes:

- **Patrones de arquitectura**: Los patrones arquitectónicos son un concepto clave en el campo de la arquitectura de software: ofrecen soluciones bien establecidas a problemas arquitectónicos, ayudan a documentar las decisiones de diseño arquitectónico, facilitan la

comunicación entre las partes interesadas a través de un vocabulario común y describen los atributos de calidad de un software. (Avgeriou 2005) .

- **Patrones de diseño:** este resulta ser una solución a un problema de diseño, pero 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 and Wesley 1994).

El reciente interés del mundo del software por los patrones tiene su origen a partir de 1995, tras la aparición y el éxito del libro **Design Patters: Elements of Reusable Object-Oriented Software** de la pandilla de los cuatro, Gang of Four, **GoF** (Erich Gamma and Wesley 1994), donde recopilan un conjunto de patrones aplicados por expertos diseñadores de software orientado a objetos. La importancia de los patrones de software radica en que aportan una solución probada a problemas específicos y recurrentes en el diseño de software, que se pueden aplicar con éxito y permiten desarrollar software basado en la experiencia colectiva de ingenieros de software experimentados.

1.7 Estrategia de pruebas

Debido a que los procesos más importantes de desarrollo de software en la mayoría de los casos tienden a ser caóticos, es necesario involucrar procesos de aseguramiento de la calidad, para que se puedan cumplir de manera correcta los requerimientos que el cliente necesita. Por otro lado, el costo que implica reparar un defecto que es descubierto en etapas avanzadas del desarrollo de software, tal como la implementación, es muy alto, hablando en términos del presupuesto del proyecto, como también en el cronograma. Por tal razón, se implementan las pruebas de software desde los comienzos del desarrollo.

Principalmente, las ventajas que trae la realización de pruebas, en un desarrollo de software son las siguientes (Lozančić 2016):

- Reducen la posibilidad de agregar defectos al software. Si hay que realizar una adición de características requeridas por el cliente y se ve que ya no funcionan bien algunas de las cosas que anteriormente servían, se puede inferir que la nueva funcionalidad es la que contiene defectos, por lo que no hay necesidad de realizar modificaciones a los componentes realizados anteriormente.
- Reducen la posibilidad de encontrar defectos en funcionalidades ya implementadas.

- Las pruebas son buena documentación. Es preferible ver unas pequeñas líneas de "Código de prueba", que son concisas y realmente son fáciles de entender, a revisar línea por línea de código para poder entender que hace determinado componente de software.
- Reducen el costo del cambio, ya que evitan que se descubran los defectos hasta el final del desarrollo de software.
- Permiten realizar reimplementación. Se puede llegar a necesitar reimplementar determinada funcionalidad en un sistema, debido a fallos de seguridad, rendimiento, o simplemente porque no reunía lo que el cliente esperaba de éste, entonces dada tal situación, las pruebas a realizar a dicha funcionalidad van a permitir que se vuelva a desarrollar de manera segura, ya que la prueba encierra el criterio de aceptación sobre la funcionalidad, permitiendo de esta forma que se vuelva a desarrollar algo acorde con la especificación.
- Restringe las características a implementar. Muchas veces los programadores pierden tiempo en detalles que la especificación no pedía. Con las pruebas el programador sabe que tiene que programar en dicha funcionalidad y también probarla, por lo que se restringe a lo que los diseñadores de las pruebas hayan realizado.
- Hacen que el desarrollo sea más rápido ya que a medida que se agregan características al software, las funcionalidades anteriores pueden fallar, pero si se han hecho las pruebas pertinentes a las funcionalidades anteriores, se puede descartar inmediatamente que existan defectos en éstas, por lo que se puede concentrar tranquilamente en la funcionalidad nueva.

Reconocidos autores como (Pressman 2005) y (Sommerville 2007) tienden a coincidir en la definición de los siguientes niveles de pruebas, adaptables a las condiciones del desarrollo en cuestión:

- Pruebas unitarias: se concentran en el diseño y comportamiento de cada componente del software, desde la perspectiva de su implementación.
- Pruebas de integración: prueban la correcta relación entre los componentes del software.
- Pruebas de aceptación: evalúan el cumplimiento de los requisitos pactados con el cliente.
- Pruebas de sistema: se ejecutan en un ambiente similar al ambiente operacional real, probando el software como un todo de conjunto con el resto de los elementos del sistema.

Uno de los pilares de la eXtreme Programming es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la

aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida demandadas por el cliente final.

Las principales ventajas de la utilización de pruebas unitarias automáticas en el desarrollo son:

1. **Fomentan el cambio**: Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
2. **Simplifica la integración**: Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
3. **Documenta el código**: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
4. **Los errores están más acotados y son más fáciles de localizar**: dado que tenemos pruebas unitarias que pueden desenmascararlos.

Las pruebas de aceptación se hacen en dos niveles: pruebas alfa y pruebas beta:

- **Pruebas Alfas:** Es aquella en que se le entrega a un usuario final todo el producto final, junto a su documentación correspondiente para que este, en presencia del desarrollador y en entornos previamente preparados para el proceso de dichas pruebas, vaya informando de todo lo que vea que no está bien, que no se cumple (Ponce 2010). Estas se llevan a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Todo esto se realiza en un ambiente controlado (Pressman 2005).
- **Pruebas Betas:** Es la que se les proporciona a usuarios finales, situados en lugares concretos de los puestos de trabajo donde finalmente será el software implantado, para que sea de nuevo el usuario y sin estar nadie presente del resto de grupos de trabajo, el que de nuevo vuelva a emitir unos informes de resultados e impresiones de la aplicación o sistema software desarrollado. Independiente del proceso seguido, también tendríamos que prestar

especial atención a la documentación entregada al cliente para su posterior evaluación, ya que la entrega de aplicación y documentación, debería tener la misma calidad (Ponce 2010).También es una aplicación “en vivo” del software en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas reales o imaginarios que se encuentran y los reporta al desarrollador periódicamente. Como resultado de los problemas reportados, es posible hacer modificaciones y luego preparar la liberación del producto de software al cliente (Pressman 2005).

1.8 Conclusiones parciales

Como resultado de este capítulo se llegaron a las siguientes conclusiones:

- El estudio de la bibliografía en el área de Anatomía Patológica, suscitó una elevada comprensión del estado del arte en el área del conocimiento en cuestión, aportando los elementos teóricos que fundamentan el presente Trabajo de Diploma.
- La práctica de la autopsia perinatal genera múltiples beneficios, su máximo aprovechamiento se sustenta en una eficiente gestión de la información generada, motivando la aparición de variados sistemas de información en todo el mundo.
- Las soluciones a nivel nacional e internacional estudiadas resultan inapropiadas para resolver el problema planteado demostrando la necesidad de desarrollar una solución que se adapte a la red hospitalaria cubana.
- Las herramientas y tecnologías seleccionadas permiten mantener la línea base de desarrollo del sistema SIRCAP al cual debe incorporarse el módulo a desarrollar.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

Capítulo 2: Análisis y Diseño

El presente capítulo recoge las principales características de la propuesta de solución. Se define el modelo de dominio de la aplicación, los principales conceptos y sus relaciones. Se analizan las técnicas de captura de requisitos, las historias de usuario y prototipos empleados en la validación de los requisitos. Resultado de la planificación, se presenta el plan de iteraciones para el desarrollo del sistema. Se describe la arquitectura, los modelos de diseño y de datos y los patrones aplicados.

2.1 Modelo conceptual

El modelado conceptual es una técnica de análisis de requisitos y de diseño de bases de datos. Ayuda a identificar problemas en los requisitos antes de comenzar el desarrollo, evitando gastos innecesarios. Permite representar de forma abstracta los conceptos y hechos relevantes del dominio del problema y transformarlos posteriormente en un esquema de una base de datos concreta. El modelo conceptual es el enlace entre los requisitos funcionales de un sistema y el diseño de la base de datos (Miguel Fernando González Pinzón 2013). A continuación, en la Figura 1, se muestra el modelo conceptual del módulo del sistema.

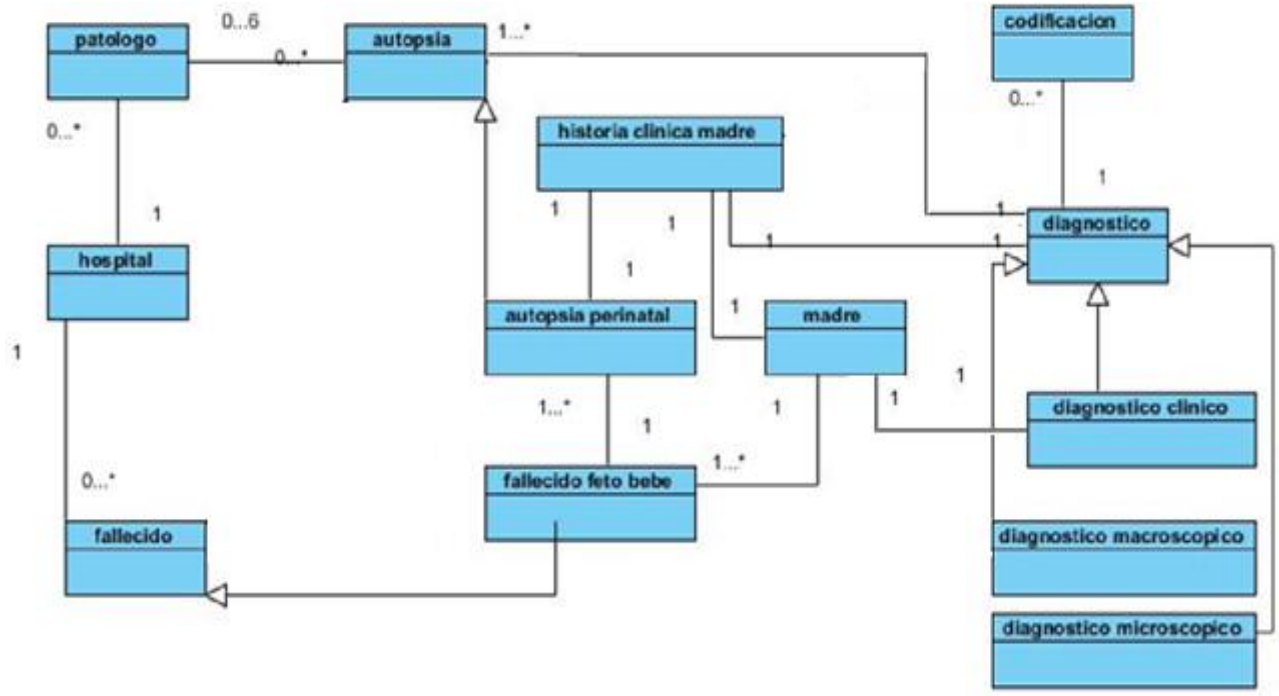

Figura 1 Modelo Conceptual.

Un patólogo, que trabaja en un hospital realiza una autopsia a un fallecido feto o bebe neonatal (hasta 28 días de nacido), el cual tiene una madre que posee una historia clínica que describe, de forma ordenada, todos los acontecimientos que ocurren a un paciente en el curso del embarazo y

las enfermedades que posee. La autopsia está compuesta por un diagnóstico macroscópico, y microscópico. El macroscópico describe el estado de cada uno de los componentes internos de los órganos y sistemas del interior del cadáver. El diagnóstico microscópico es donde se define la causa de muerte y se explican los procesos que la describen, mediante el análisis del diagnóstico macroscópico y la historia clínica de la mamá. La causa de muerte tiene una codificación internacional según las causas de muerte registradas en la norma de codificación de la OMS.

2.2 Ingeniería de requisitos

Un área de conocimiento de gran importancia en el desarrollo de software, es la ingeniería de requerimientos. Esta comprende las actividades de obtención (captura, descubrimiento y adquisición), análisis (y negociación), especificación, y validación de requisitos. Además, establece una actividad de gestión de requerimientos para manejar los cambios, mantenimiento y rastreabilidad de los requerimientos (Guerra 2020).

2.2.1 Identificación de requisitos

El proceso de obtención de requisitos, cuya finalidad es llevar a la luz los requisitos, no solo es un proceso técnico, sino también un proceso social que envuelve a diferentes personas, lo que conlleva dificultades añadidas a su realización. Existe un gran número de técnicas para obtener requisitos, sin embargo, ninguna es suficiente por sí sola, por lo que se recomienda combinarlas para obtener requerimientos completos. A continuación, se describen las que fueron utilizadas en la investigación.

Entrevistas

La entrevista es de gran utilidad para obtener información cualitativa como opiniones o descripciones subjetivas de actividades. Es una técnica muy utilizada, y requiere una mayor preparación y experiencia por parte del analista. Se puede definir como un “intento sistemático de recoger información de otra persona” a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada. Debe quedar claro que no basta con hacer preguntas para obtener toda la información necesaria. Es muy importante la forma en que se plantea la conversación y la relación que se establece en ella (Guerra 2020). Durante este proceso se entrevistaron a los especialistas:

- Doctor José Hurtado de Mendoza Amat, Profesor titular e Investigador Titular, Doctor en Ciencias Médicas, Especialista en II Grado en Anatomía Patológica y Profesor Consultante. Ha realizado varios artículos y publicaciones sobre la necesidad de un sistema informático que gestione toda la información generada en las diferentes áreas de la anatomía patológica.

- Doctora Sandra Aguilar Isla, Especialista en Anatomía Patológica, Directora de Patología Perinatal de la provincia de la Habana y Especialista en Ginecopatología. Es la patóloga principal responsable de la recogida de información y diagnóstico de enfermedades de autopsias perinatales en el país.

Ambos especialistas proporcionaron la parte de la información médica necesaria para el entendiendo del negocio y desarrollo del módulo. Mencionaron los datos que se almacenan en cada autopsia, así como la descripción del proceso de recogida de información y la interrelación entre los distintos elementos que componen el proceso. También argumentaron que elementos que eran necesarios informatizar dentro del proceso de gestión de información de las autopsias perinatales y sugirieron bibliografías básicas necesarias para un mayor entendimiento del objeto de estudio.

Estudio de documentación

- La principal documentación estudiada fue el libro ¨Autopsia, garantía de calidad en la medicina¨ del doctor José de Mendoza (Hurtado de Mendoza Amat 2014), bibliografía básica de anatomía patológica en Cuba, donde se describen los principales procesos del desarrollo de la autopsia, así como los datos de protocolo de autopsia perinatal en el país. También se investigaron conceptos básicos relacionados con el objeto de estudio definidos por la organización mundial de la salud y la codificación médica establecida en la CIE10.
- Para el desarrollo de la aplicación se estudió la documentación referida a la metodologías, herramientas y lenguajes a utilizar, para una mayor comprensión de cada uno.

Tormenta de ideas (Brainstorming)

Esta técnica se utilizó para identificar los requisitos, en la cual, se tuvo en cuenta la información brindada por los especialistas de la medicina y se realizó el análisis de las necesidades del módulo. Como resultado del uso de las técnicas descritas anteriormente se identificaron 28 requisitos funcionales y 7 no funcionales los cuales se describen en los siguientes epígrafes.

2.2.2 Requisitos funcionales

A continuación, se muestran los requisitos funcionales identificados:

- **RF-01** Adicionar fallecido perinatal.
- **RF-02** Modificar fallecido perinatal.

- **RF-03** Buscar fallecido perinatal.
- **RF-04** Visualizar fallecido perinatal.
- **RF-05** Adicionar diagnóstico clínico.
- **RF-06** Modificar diagnóstico clínico.
- **RF-07** Buscar diagnóstico clínico.
- **RF-08** Visualizar diagnóstico clínico.
- **RF-09** Adicionar diagnóstico macroscópico.
- **RF-10** Modificar diagnóstico macroscópico.
- **RF-11** Buscar diagnóstico macroscópico.
- **RF-12** Visualizar diagnóstico macroscópico.
- **RF-13** Adicionar diagnóstico microscópico.
- **RF-14** Modificar diagnóstico microscópico.
- **RF-15** Buscar diagnóstico microscópico.
- **RF-16** Visualizar diagnóstico microscópico.
- **RF-17** Adicionar autopsia perinatal.
- **RF-18** Modificar autopsia perinatal.
- **RF-19** Buscar autopsia perinatal.
- **RF-20** Visualizar autopsia perinatal.
- **RF-21** Adicionar reportes dinámicos.
- **RF-22** Modificar reportes dinámicos.
- **RF-23** Buscar reportes dinámicos.
- **RF-24** Visualizar reportes dinámicos.
- **RF-25** Registrar patólogo.
- **RF-26** Modificar patólogo.
- **RF-27** Desactivar patólogo.

- **RF-28** Emitir informe de autopsia.

2.2.3 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Se aplican al sistema en su totalidad y no se refieren directamente a funciones específicas, sino a sus propiedades emergentes; la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento son algunas de ellas. Pueden estar dirigidos incluso al proceso de desarrollo, especificando herramientas o estándares de calidad a emplear en función de las necesidades del usuario (Sommerville 2007). A continuación, se enuncian los requisitos no funcionales identificados en la propuesta de solución:

Requisitos de hardware:

- **RNF-01** Recursos tecnológicos.

  El análisis de la base tecnológica presente en los departamentos de Anatomía Patológica del país, arrojó que el nuevo sistema deberá operar con prestaciones mínimas de hardware:

  1. Procesador: 1.6 GHz. o superior.
  2. Memoria RAM: 512 MB o superior.
  3. Disco duro: 40 GB o superior.

Requisitos de software:

- **RNF-02** Entorno de ejecución.

  Para la ejecución de la aplicación, con independencia del sistema operativo, constituyen premisa indispensable las siguientes instalaciones:

  1. Entorno de ejecución de Java (JRE), versión 1.8 o superior.
  2. Sistema gestor de bases de datos PostgreSQL, versión 9.2 o superior.

Requisitos de interfaz:

- **RNF-03** Interfaz de usuario.

  El sistema debe ofrecer una interfaz amigable, intuitiva y fácil de operar, con un modo de operación lo más cercano posible al del software en explotación. Además, debe mantener la línea de diseño establecida para alcanzar la uniformidad en la solución final.

Requisitos de seguridad:

- **RNF-04** La seguridad estará regida por un mecanismo de control de acceso basado en roles, convenientemente asignados a los usuarios del sistema al momento de su creación.
- **RNF-05** Se establecerá un mecanismo de autenticación para el acceso al sistema.
- **RNF-06** Las contraseñas no podrán ser almacenadas en texto plano, o en espacios que permitan su acceso o modificación por personas no autorizadas.
- **RNF-07** El sistema debe preservar la seguridad de la información, garantizando en todo momento su confidencialidad, integridad y disponibilidad.

2.2.4 Especificación de requisitos funcionales

La especificación de requisitos establece la base para el acuerdo entre usuarios y desarrolladores de software, quedando definido el comportamiento deseado del producto (IEEE 2004).

En el entorno de XP, las historias de usuario constituyen el artefacto utilizado para describir las funcionalidades del sistema (Kent Beck,1996). Las mismas contienen una breve descripción del comportamiento del sistema desde la perspectiva del usuario y representan un medio de comunicación entre el software y el equipo de desarrollo. A continuación, se presentan las historias de usuario correspondientes a tres de los requisitos funcionales identificados (Ver de las Tablas 3 a la 5).

Tabla 3 Historia de Usuario: Adicionar autopsia perinatal.

HISTORIA DE USUARIO	
Número: HU-5	**Usuario:** Todos
Nombre de historia: Adicionar fallecido perinatal.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 8	**Iteración asignada:** 1
Programador responsable: Daymari Martínez Rodríguez.	
Descripción: Permite registrar los datos generales asociados a un fallecido perinatal.	
Observaciones:	

Tabla 4 Historia de usuario: Buscar autopsia perinatal.

HISTORIA DE USUARIO	
Número: HU-8	**Usuario:** Todos

Nombre de historia: Modificar fallecido perinatal.	
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Daymari Martínez Rodríguez.	
Descripción: Permite modificar los datos de una autopsia perinatal.	
Observaciones: Se requiere que el usuario haya insertado los datos del fallecido previamente.	

Tabla 5 Historia de usuario: Buscar fallecido perinatal.

HISTORIA DE USUARIO	
Número: HU-17	**Usuario:** Todos
Nombre de historia: Buscar fallecido perinatal.	
Prioridad en negocio: Media.	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 3
Programador responsable: Daymari Martínez Rodríguez.	
Descripción: Muestra los datos de los fallecidos perinatales registrados en el sistema que cumplen criterios de búsqueda previamente establecidos, permitiendo modificarlos si fuera necesario.	
Observaciones:	

2.2.5 Validación de requisitos funcionales

El objetivo principal de la actividad de validar los requisitos es comprobar que el software descrito por la especificación de requisitos del sistema se corresponde con las necesidades de negocio de clientes y usuarios, obteniendo así su aprobación y permitiendo generar una línea base entre los requisitos y las peticiones de cambio del sistema a desarrollar. Esta es una de las tareas que forman parte del proceso de desarrollar los requisitos de un software que satisfaga las necesidades de negocio dentro del proceso de ingeniería de requisitos.

Para la validación de los requisitos se hace uso de los prototipos de interfaz de usuario ya que permiten a los clientes y usuarios entender más fácilmente el software propuesto y los requisitos en formato textual. Si durante este proceso los clientes y usuarios ponen de manifiesto discrepancias con los requisitos propuestos, estos deben identificarse como problemas que son resueltos en el

proceso de gestión de los requisitos. A continuación, se muestran el prototipo correspondiente a la validación del requisito de la historia de usuario “Insertar diagnóstico macroscópico” (Ver Figura 2).

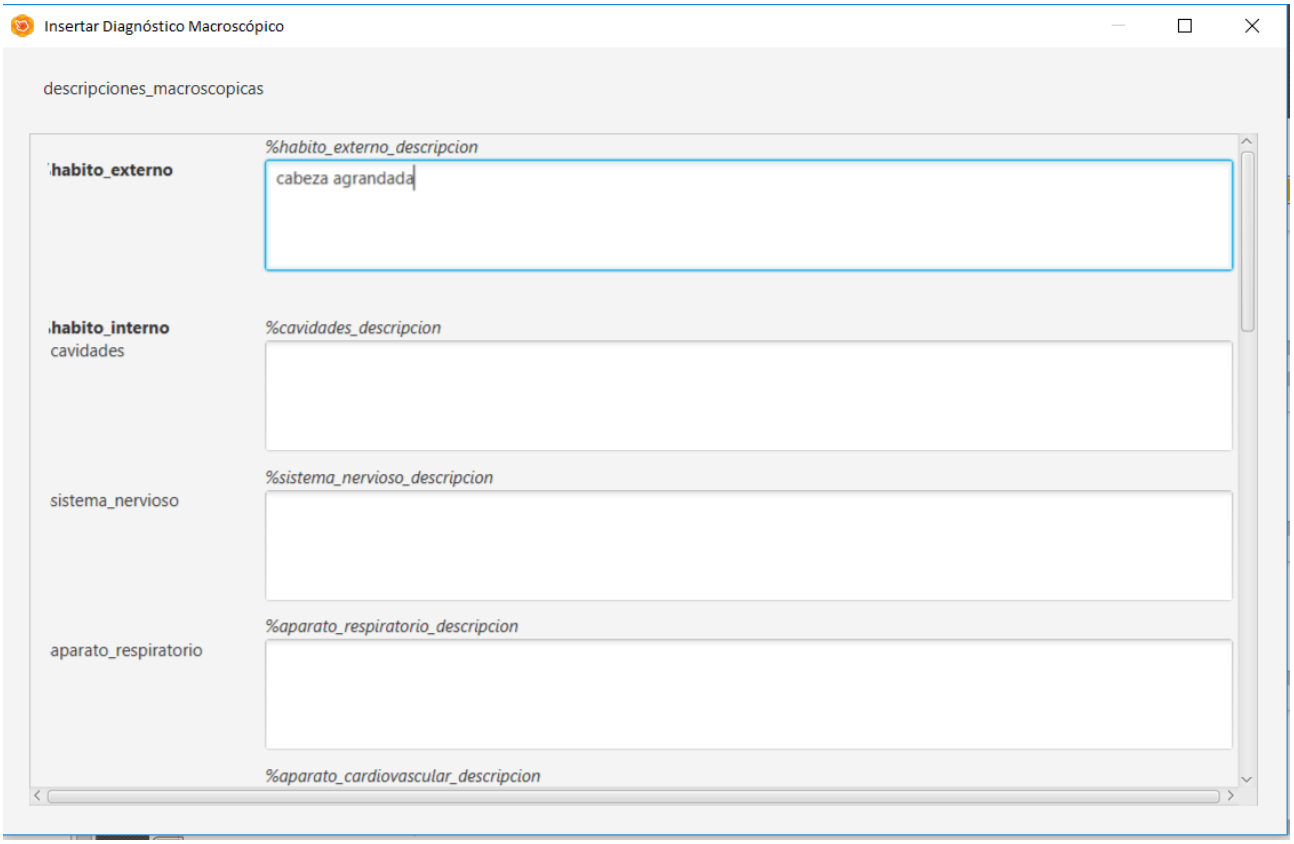

Figura 2 Prototipo de interfaz de usuario: Insertar diagnóstico macroscópico.

2.3 Plan de iteraciones

El ciclo de desarrollo de software guiado por XP se caracteriza por ser iterativo e incremental. Los niveles de prioridad para el negocio y el esfuerzo estimado de implementación para cada una de las historias de usuario especificadas, hicieron posible la planificación de la implementación del sistema, quedando definidas cuáles historias de usuario serían implementadas en cada iteración.

Tabla 6 Plan de iteraciones.

No. Iteración	**Historia de usuario**	**Prioridad**	**Esfuerzo estimado**	
1	Adicionar fallecido perinatal.	Alta	8	40
	Adicionar diagnóstico clínico.	Alta	5	

	Adicionar diagnóstico macroscópico.	Alta	3	
	Adicionar diagnóstico microscópico.	Alta	8	
	Adicionar autopsia perinatal.	Alta	8	
	Registrar patólogo.	Alta	8	
2	Visualizar fallecido perinatal.	Media	3	25
	Modificar fallecido perinatal.	Media	2	
	Visualizar diagnóstico clínico.	Media	3	
	Modificar diagnóstico clínico.	Media	2	
	Visualizar diagnóstico macroscópico.	Media	3	
	Modificar diagnóstico macroscópico.	Media	2	
	Visualizar diagnóstico microscópico.	Media	3	
	Modificar diagnóstico microscópico.	Media	2	
	Visualizar autopsia perinatal.	Media	3	
	Modificar autopsia perinatal.	Media	2	
3	Buscar fallecido perinatal.	Media	4	71
	Buscar autopsia perinatal.	Media	4	
	Buscar diagnostico macroscópico.	Media	4	
	Buscar diagnostico microscópico.	Media	4	
	Buscar diagnóstico clínico.	Media	4	
	Adicionar reportes dinámicos.	Alta	21	
	Visualizar reportes dinámicos.	Media	13	
	Modificar reportes dinámicos.	Media	4	
	Buscar reportes dinámicos.	Media	3	
	Modificar patólogo.	Baja	4	
	Desactivar patólogo.	Baja	3	
	Exportar informe de autopsias	Baja	3	

CAPÍTULO 2: ANÁLISIS Y DISEÑO

2.4 Arquitectura de la solución

Para el diseño arquitectónico del sistema se hará uso del patrón Modelo Vista Controlador (Ver Figura 3), el cual garantiza la organización del código fuente de la aplicación, dividiéndola en tres componentes fundamentales: el modelado del dominio, la presentación y las clases controladoras.

Figura 3 Modelo vista controlado. Fuente: creación propia.

- El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tantas 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 (Buschmann et al. 2001).
- 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' (Buschmann et al. 2001).

- 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 (Buschmann et al. 2001).

2.5 Modelo de diseño

Uno de los principios fundamentales de la metodología XP es la simplicidad, por tanto, en la etapa de diseño se propone el empleo de tarjetas Clase Responsabilidad Colaboración (CRC), en lugar de diagramas de clases para la descripción del sistema en notación UML.

La característica más sobresaliente de las tarjetas CRC es su sencillez y adaptabilidad (Casas and Reinaga 2012). Las tarjetas CRC son fichas que representan cada una de las clases del sistema, en ellas se describen brevemente las responsabilidades de la clase, mostrando un listado de las clases con las que colabora para ejecutar dichas responsabilidades. Su utilización en el diseño potencia el uso de patrones de asignación de responsabilidades (GRASP) (Larman 1999). A continuación, se muestran las tarjetas CRC correspondientes a tres las clases controladoras presentes en la propuesta de solución (Ver Tablas 7, 8 y 9).

Tabla 7 Tarjeta CRC: CodificadorController.java.

TARJETA CRC	
Clase: CodificadorController.java	
Súper Clase: AnchorPane.java	
Responsabilidades:	**Colaboraciones:**
Realizar codificación de Causa de Muerte. Modificar codificación de Causa de Muerte.	Codificacion.java DiagnosticoClinicoController.java DiagnosticosMacroscopicosController.java DiagnosticosAnatomopatologicosController.java

Tabla 8 Tarjeta CRC: ProtocoloController.java.

TARJETA CRC
Clase: ProtocoloController.java
Súper Clase: AnchorPane.java

Responsabilidades:	**Colaboraciones:**
Registrar los datos relacionados con la autopsia perianal. Modificar los datos relacionados con la autopsia perianal.	AutopsiaPedianal.java DiagnosticoClinicoController.java DiagnosticosMacroscopicosController.java DiagnosticosAnatomopatologicosController.java

Tabla 9 Tarjeta CRC: BuscarAutopsiaPerinatalController.java.

TARJETA CRC	
Clase: BuscarAutopsiaPeriantalController.java	
Súper Clase: AnchorPane.java	
Responsabilidades:	**Colaboraciones:**
Recuperar las autopsias perinatales que cumplan los criterios de búsqueda establecidos.	AutopsiaPerinatal.java FallecidoPerinatal.java ProtocoloController.java

2.6 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular (Erich Gamma and Wesley 1994).

Relacionados con el diseño de los objetos y frameworks de pequeña y mediana escala. Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño estructurado, común, que lo hace útil para la creación de diseños orientados a objetos reutilizables (slideshare. 2020).

2.6.1 Patrones GOF

Los patrones GOF a usar en la solución propuesta son los siguientes

- **Factory Method (Método de Fabricación)**: Define una interfaz para crear un objeto, pero deja que las subclases decidan qué clase se instancia. Permite que una clase delegue en sus subclases la creación de objetos (Gamma et al. 2004). Se empleó como complemento al patrón DAO, centralizando en una clase *(DAOHibernateFactory.java)* la responsabilidad de instanciar las implementaciones DAO particulares del ORM Hibernate, evitando la dependencia del mismo en las clases controladoras.
- **Singleton (Único)**: Garantiza que sólo exista una instancia de la Clase y proporciona un punto de acceso global a esa instancia(Gamma et al. 2004). Se utilizó para garantizar la

existencia de una única instancia de la clase *SessionFactory.java*, canal único de comunicación entre la base de datos y la aplicación, garantizando un mecanismo de acceso global a dicha instancia.

- **Facade (Fachada)**: Proporciona una interfaz unificada para un conjunto de interfaces. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar *)* (Gamma et al. 2004). Empleado en la interfaz principal *(SARCAPController.java*.
- **Visitor (Visitante)**: Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera (Gamma et al. 2004). Se empleó para separar los algoritmos del control interno de calidad de la estructura de las entidades persistentes involucradas, unificándolos en una sola clase creada a tales efectos.

2.6.2 Patrones GRASP

Los patrones GRASP fueron utilizados en la propuesta de solución de la siguiente forma:

- **Experto:** con el fin de asignar las responsabilidades a la clase experta en la información (Larman 1999). El patrón resulta de utilidad en las clases del modelo, las cuales contienen toda la información relacionada con los objetos persistentes que representan.
- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de los objetos (Larman 1999). Aplicable a la clase *SARCAPController.java*, la cual instancia las clases encargadas de cada una de las funcionalidades accesibles desde la interfaz principal.
- **Bajo acoplamiento:** garantiza la asignación de responsabilidades de manera que el acoplamiento entre las clases permanezca bajo (Larman 1999**)**. El uso de este patrón se evidencia en todo el diseño del módulo, donde solo existen las relaciones necesarias entre las clases para que puedan cumplir con las responsabilidades asignadas, minimizándose el número de relaciones entre ellas.
- **Alta cohesión:** con el propósito de asignar las responsabilidades a las clases de manera que permanezcan lo más cohesionadas posible, garantizando que ninguna realice un trabajo excesivo gracias a la colaboración con otras clases (Usaola 2020). Ejemplo de esto es la clase AutopsiaPerinalNeonatal la cual al recibir un cambio en sus atributos requiere una restructuración de los métodos para su correcto funcionamiento.
- **Controlador:** propone asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase (Larman 1999). En la arquitectura definida este patrón se evidencia

en las clases controladoras, responsables de implementar todas las funcionalidades pertenecientes a una interfaz determinada.

La utilización de estos patrones resultó en un diseño sencillo, pensado para la creación de un sistema robusto, fácil de entender, mantener y ampliar; aumentando la capacidad de reutilización de sus componentes y conservando el encapsulamiento de la información.

2.6.3 Patrón Objeto de Acceso a Datos

El patrón Objeto de Acceso a Datos (DAO, por sus siglas en inglés) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo**.** Propone separar por completo la lógica de negocio de la lógica para acceder a los datos, de esta forma, el DAO proporcionará los métodos necesarios para insertar, actualizar, borrar y consultar la información; por otra parte, la capa de negocio solo se preocupa por lógica de negocio y utiliza el DAO para interactuar con la fuente de datos (Blancarte 2018). Se utilizó el patrón DAO para: abstraer y encapsular los accesos, gestionar la conexión a la fuente de datos obtener los datos almacenados.

2.7 Modelo de datos

El **modelo de datos** es un modelo abstracto que organiza elementos de datos y estandariza cómo se relacionan entre sí y con las propiedades de las entidades del mundo real (ORACLE 2020).

El Modelo Entidad-Relación (MER) asociado a la propuesta de solución está compuesto por 65 tablas, de ellas 32 nomencladoras y el resto destinadas al almacenamiento de la información correspondiente al dominio del problema (Ver Anexo 7). En él se ven las tablas del sistema principal junto con las del módulo a desarrollar. A continuación, se muestra algunas de las tablas más importantes del modelo de datos de todo el sistema (Ver Figura 4 y 5).

CAPÍTULO 2: ANÁLISIS Y DISEÑO

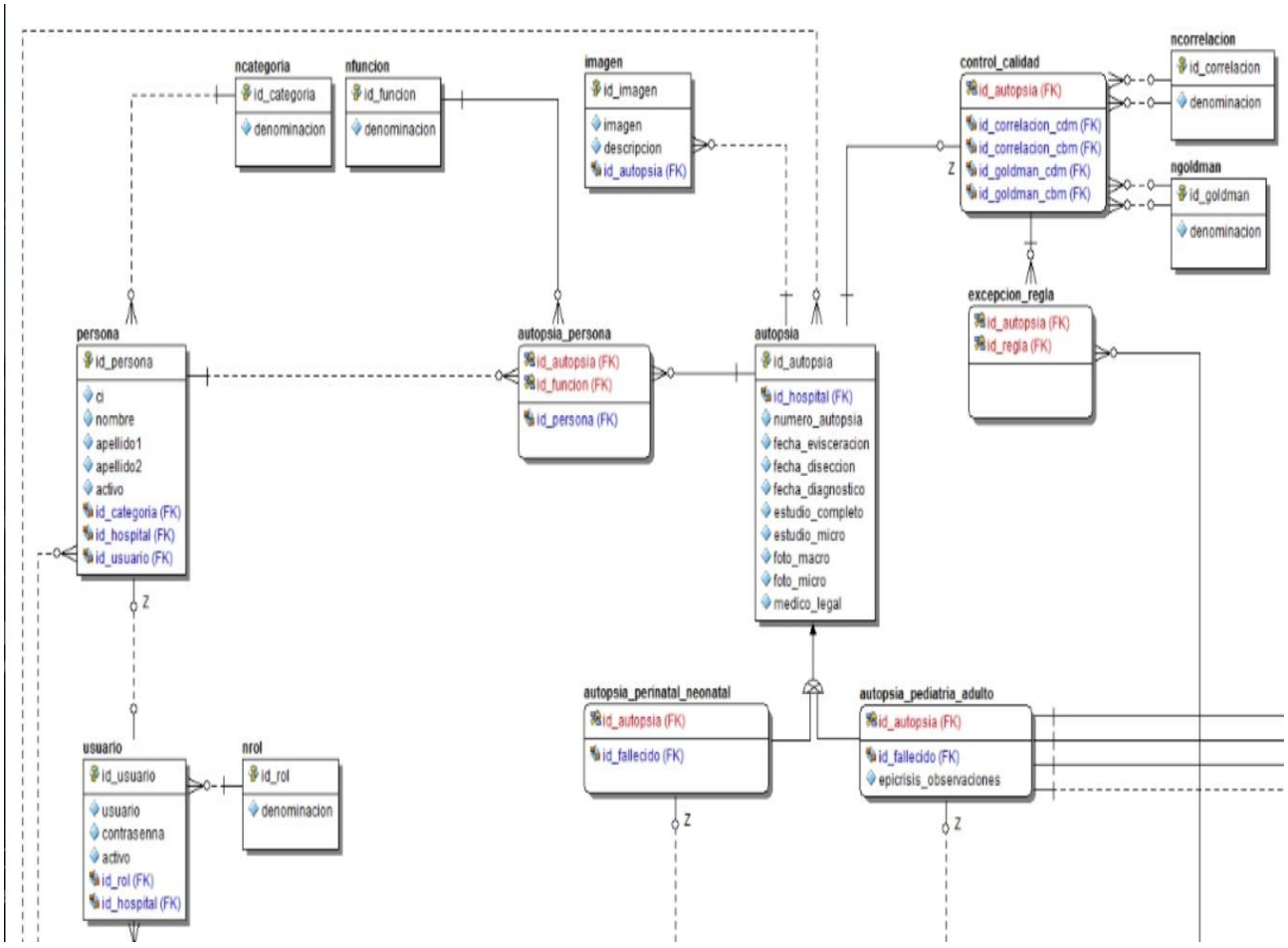

Figura 4 Tablas significativas (14) del modelo de datos.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

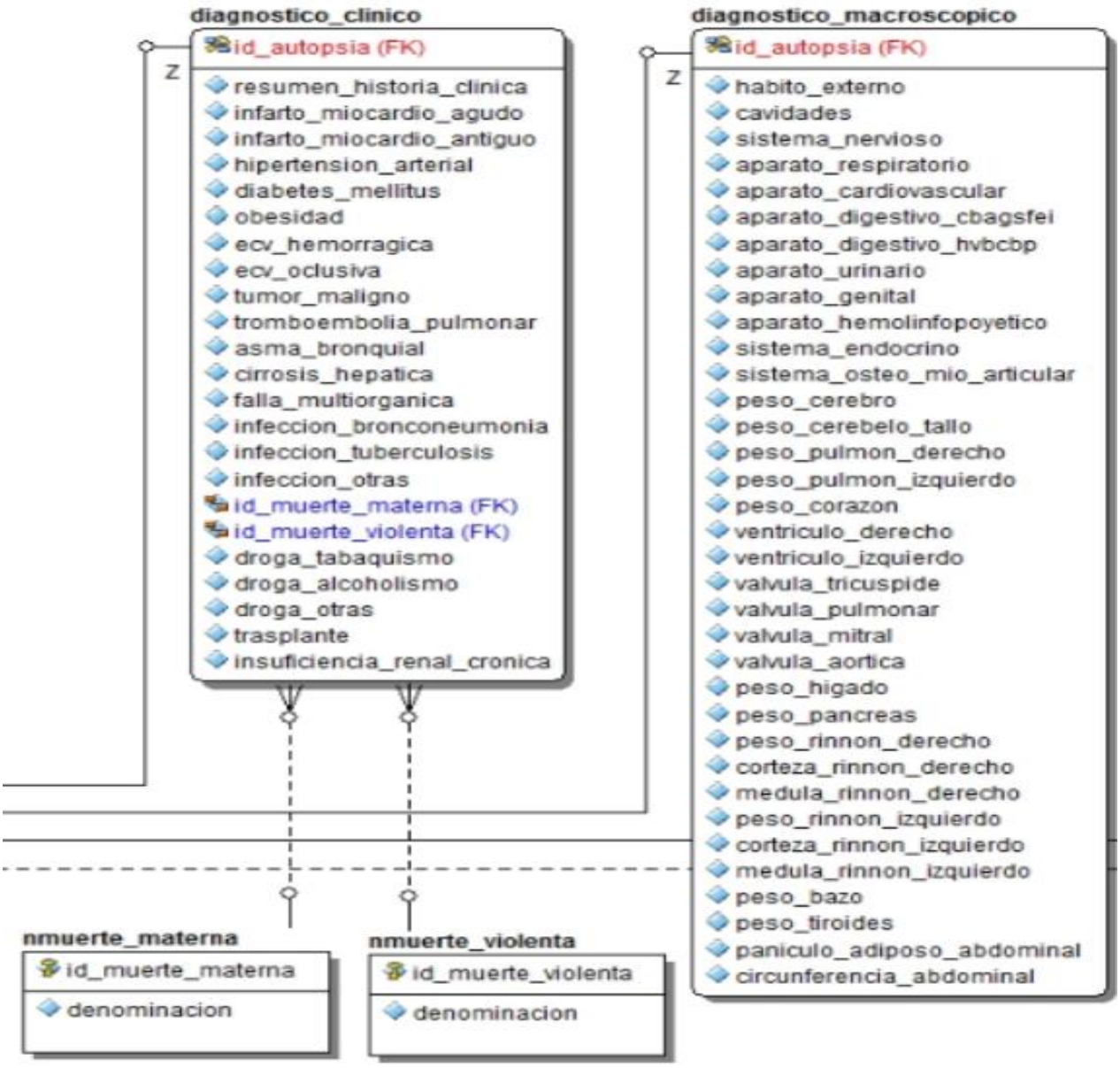

Figura 5 Tablas significativas (4) del modelo de datos.

2.8 Conclusiones parciales

Con este capítulo, quedaron definidos los elementos esenciales para la construcción de la propuesta de solución y como resultado del análisis y diseño se concluyó que:

- La creación de un modelo conceptual permitió una representación visual del dominio del problema.

- La especificación y validación de los requisitos del sistema contribuyó a la planificación de la implementación en función del esfuerzo necesario y su prioridad para el negocio.
- El uso de patrones de diseño contribuyó a lograr un diseño sencillo, fácil de mantener y ampliar, proporcionando una estructura para el módulo a desarrollar y posibilitando el empleo de buenas prácticas de programación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Capítulo 3: Implementación y Prueba

En el presente capítulo se describe el modelo de implementación de la solución propuesta y los estándares de codificación utilizados. Se explica la estrategia de pruebas a seguir en el desarrollo de la solución para validar la misma, y finalmente se exponen los resultados obtenidos en la aplicación de las pruebas.

3.1 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El uso de estándares de codificación permite lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo (Alonso 2012).

Nomenclatura

El idioma por defecto a la hora de dar sentido funcional al nombre de clases, variables y constantes. Este será una mezcla entre la nomenclatura tradicional en español y la nomenclatura funcional adoptada.

Cabecera de archivo.

Todos los archivos deben iniciar con una cabecera que especifique el autor de los últimos cambios, adicionalmente se podrán incluir comentarios u otros datos de interés (Ver Figura 6).

Figura 6 Cabecera de archivo.

Paquetes

Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales (Ver Figura 7).

.

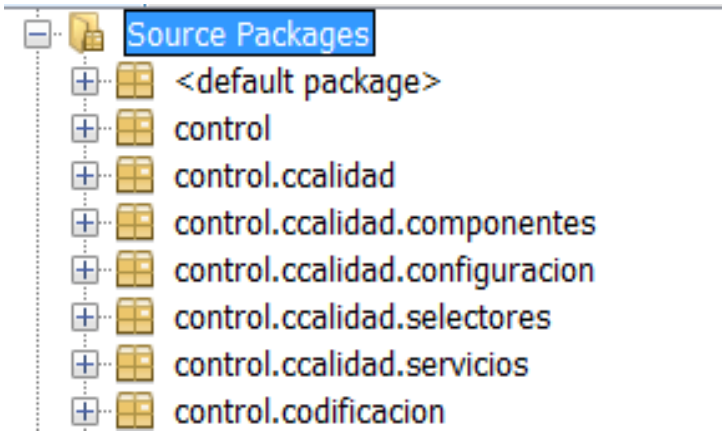

Figura 7 Paquetes de clases.

Nombres de clases

Los nombres de las clases deberán ser en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas siguiendo el estilo de codificación Lower Camel Case. El nombre ha de ser lo suficientemente descriptivo, no importando a priori la longitud del mismo y no se permiten caracteres especiales (Ver Figura 8).

Figura 8 Definición de clases.

Métodos

Para la definición de los métodos se va a usar el estilo de codificación Lower Camel Case. Los nombres deberán ser verbos en infinito, iniciándose en minúscula con la primera letra

de cada palabra interna en mayúscula. No se permiten caracteres especiales, siendo el nombre lo suficientemente descriptivo sin importar su longitud (Ver Figura 9).

```
34      public void getEvisceracionPrincipal() {
35          //instrucciones
36      }
```

Figura 9 Definición de métodos.

Variables y Constantes

Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos. Se escribirán en su totalidad en minúscula, usando la mayúscula para la separación de palabras y cada palabra interna en mayúsculas (Ver Figura 10).

```
11      private String idAutopsia;
12      private Date fechaEvisceracion;
13      private Set<Imagen> imagenes;
```

Figura 10 Definición de variables y constantes.

Comentarios

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se utilizan los caracteres /* para especificar el inicio del comentario y */ para su fin (en caso de que sea mayor de una línea de código) y se utiliza // para un comentario de una sola línea de código (Ver Figura 11).

```
/**
 * The main() method is ignored in correctly deployed JavaFX application.
 * main() serves only as fallback in case the application can not be
 * launched through deployment artifacts, e.g., in IDEs with limited FX
 * support. NetBeans ignores main().
 *
 * @param args the command line arguments
 */
```

Figura 11 Definición de comentarios.

Uso y posición de llaves en bloques de instrucciones.

Todo bloque de instrucciones debe ir entre llaves, aun cuando conste de una sola línea. Las llaves de apertura se colocarán al final de la sentencia que delimita el bloque de instrucciones, las de cierre se alinean con el inicio de la sentencia en una nueva línea (Ver Figura 12).

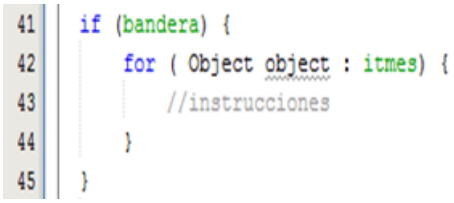

Figura 12 Uso y posición de llaves en bloques de instrucciones.

Espacios entre operadores.

Para mejor legibilidad del código, se colocarán espacios a ambos lados de los operadores (Ver Figura 13).

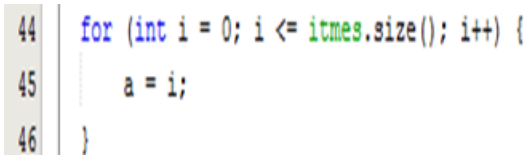

Figura 13 Espacios entre operadores.

3.2 Pruebas de unidad

La prueba de unidad se realiza cuando: la interfaz de un método no es clara, la implementación es complicada, para testear entradas y condiciones inusuales o luego de modificar algo. Éstas deben contemplar cada módulo del sistema que pueda generar fallas. Para poder integrar el código realizado al ya existente, el mismo debe aprobar satisfactoriamente todos los casos de prueba definidos (Malfará et al. 2006) .

En XP se deben escribir las pruebas unitarias para cada módulo antes de escribir el código. No es necesario escribir casos de prueba para todos los módulos, sólo para aquellos en que exista la posibilidad de que puedan fallar. Luego de escribir el código, se ejecutan las pruebas, las cuales deben resultar 100% efectivas para que el código pueda integrarse al sistema. En caso contrario hay que solucionar los errores y ejecutar nuevamente los casos de prueba hasta lograr que ninguno de ellos posea errores (Malfará et al. 2006).

Para realizar las pruebas unitarias del módulo propuesto se va a utilizar la herramienta JUnit, la cual constituye un marco de trabajo de código abierto para la creación de pruebas

automatizadas y una instancia de la arquitectura Unit para marcos de trabajo enfocados en pruebas de unidad, de forma tal de poder soportar un testing continuo y mantener organizados los casos de pruebas.

JUnit es el framework xUnit para la plataforma Java, que permite definir los casos de prueba unitarios para las clases importantes del sistema. Permite ejecutar las clases de forma controlada para poder evaluar si estas se comportan de la manera esperada. JUnit ejecutará las pruebas sobre la clase y comparará contra los resultados esperados. Si los resultados son correctos indicará que las pruebas fueron exitosas, en caso que alguno de los resultados no sea el esperado Junit indicará que se produjo un fallo, señalando el o los métodos que no pasaron sus evaluaciones.(JUnit 2020). Es importante destacar que Junit seguirá con las pruebas, aunque ya se haya detectado una falla.

En JUnit, los casos de prueba son clases que derivan de la clase TestCase, e implementan métodos sin parámetros de nombre testXXX, donde XXX es una descripción de lo que está probando ese método (Gamma et al. 2004). Las pruebas implementadas que se extenderán de la clase TestCase tienen la posibilidad de sobrescribir los métodos setUp() y tearDown(), los cuales son invocados antes y después de cada método de test. Esto permite inicializar y liberar recursos entre la ejecución de los distintos métodos en la clase, permitiendo asegurarse de que no hay efectos colaterales entre la ejecución de los distintos test. El propio framework incluye formas de ver los resultados (runners) que pueden ser en modo texto.

El IDE NetBeans, que fue usado para la implementación, cuenta con plug-ins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, lo que facilitó el hecho de enfocarse en la prueba y el resultado esperado, y dejó a la herramienta la creación de las clases que permiten coordinar las pruebas.

Para escribir un test, se siguieron estos pasos:

1. Definición de una subclase de TestCase.
2. Sobrescribir el método setUp() para inicializar el objeto(s) a probar.
3. Sobrescribir el método tearDown() para liberar el objeto(s) a probar.
4. Definir uno o más métodos testXXX() públicos que prueben el objeto(s) y aserten los resultados esperados.
5. Definir un método factoría suite () estático que cree un TestSuite que contenga todos los métodos testXXX() del TestCase.
6. Definir un método main() que ejecute el TestCase en modo por lotes.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

A continuación, se describe como se realizaron las pruebas para la clase AutopsiaPerinatalNeonatal. La misma consta de 11 atributos y 3 métodos (Ver Figura 14).

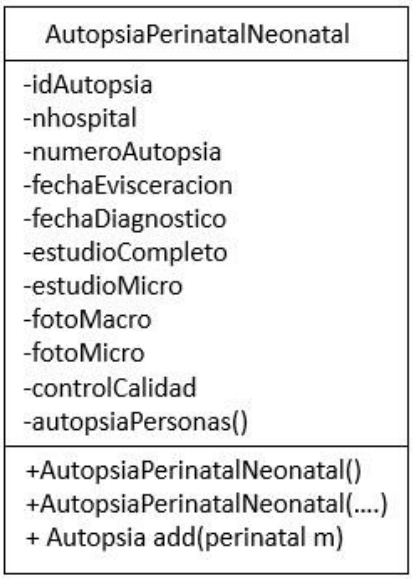

Figura 14 Clase AutopsiaPerinatalNeonatal.

La Tabla 10 muestra una breve descripción de los métodos de la clase AutopiaPerinatalNeonatal.

Tabla 10 Métodos de una clase que se le realizan pruebas.

Método	**Descripción**
public AutopsiaPerinatalNeonatal ()	Constructor de la clase de la clase padre Autopsia
public AutopsiaPerinatalNeonatal(Nhospital nhospital, String autopsia, Date fechaEvisceracion, Date fechaDiseccion, Date fechaDiagnostico, Boolean estudioCompleto, Boolean estudioMicro, Boolean fotoMacro, Boolean fotoMicro, Boolean medicoLegal)	Contructor de la clase AutopsiaPerinatalNeonata
public Autopsia add(perinatal m)	Devuelve un objeto Autopsia si se le pasan los datos de un bebe perinatal.

Los pasos que se siguieron, para construir un caso de prueba para la clase AutopsiaPerinatalNeonatal, fueron los siguientes:

1. Definir la clase de prueba que derive de la clase TestCase (Ver Figura 15).

```
import junit.framework.TestCase;
import junit.framework.TestSuite;
import junit.framework.Test;

   public class PerinatalTest extends TestCase {
   //…
   private AutopsiaPerinatalNeonatal12;
   private AutopsiaPerinatalNeonatal14;
   //…
}
```

Figura 15 Clase TestCase.

2. Sobreescribir el método setUp() para inicializar el/los objetos implicados en la prueba (Ver Figura 16).

```
protected void setUp() {
   mobj12 = new Perinatal(12, datos);
   mobjf14 = new Perinatal(14, datos);
}
```

Figura 16 Método setUp().

3. Sobreescribir el método tearDown() para liberar el/los objetos implicados en la prueba (en este caso no fue necesario realizar el paso porque no hacía falta liberar objetos ya que no iban a ser utilizados nuevamente).
4. Definir uno o más métodos testXXX() que prueban las diferentes funcionalidades y casos de la clase (Ver Figura 17).

```
public void testAdd() {
   Autopsia expected = new AutopiaPerinatalNeonatal(26, datos);
   Autopsia result =AutopiaPerinatalNeonatal.add (12, datos);
   assert(expected.equals(result));
}
public void testEquals() {
   assert(!result.equals(null));
   assert(mobj12, mobj12datos);
   assertEquals(result, new Autopsia(12, datos));
   assert(!.equals(mobjf14));
}
```

Figura 17 Métodos Test.

5. Definir un método de clase suite() que crea una TestSuite a la que se añaden todos los métodos testXXX() del TestCase (Ver Figura 18).

```
public static TestSuite () {
   TestSuite suite = new TestSuite();
   suite.addTest(new PeriantalTest("testEquals"));
   suite.addTest(new PeriantalTest("testAdd"));
   return suite;
}
```

Figura 18 Método de clase Suite().

6. Definir un método main() que ejecuta el TestCase (o invocar el Test Runner en modo gráfico y especificar el caso de prueba o la suite(Ver Figura 19).

```
public static void main(String args[]) {
   test.textui.TestRunner.run(suite());
}
```

Figura 19 Método main().

La interfaz de usuario en modo gráfico presenta una barra de progreso la cual toma color verde si todas las pruebas ejecutadas son 100% satisfactorias, o color rojo si falló alguna. En este último caso se despliega en una lista la descripción de las fallas o errores generados (Ver Figura 20).

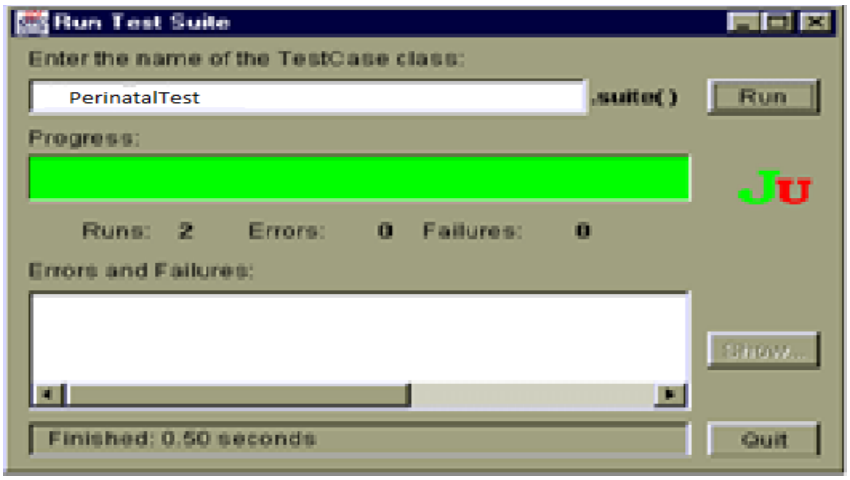

Figura 20 Interfaz de usuario de Junit.

Haciendo uso de la herramienta JUnit fueron realizadas las pruebas unitarias a las clases más relevantes para la solución, probándose la ejecución controlada de sus funcionalidades lo cual permitió evaluarlas acorde a su comportamiento esperado. Los errores fueron corregidos a medida que fueron detectados. La automatización de las pruebas facilitó su repetición luego de cada cambio importante en el código fuente, garantizando la efectividad de la integración entre los componentes.

3.3 Pruebas de integración

La prueba de integración es la prueba que se aplica cuando todos los módulos se combinan para formar un programa de trabajo. La prueba se realiza a nivel de módulo, en lugar de a nivel de declaración como en las pruebas unitarias. Estas enfatizan las interacciones entre módulos y sus interfaces. Las mismas permiten el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente, con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes (Leung 1990).

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Objetivos de las pruebas de integración

El objetivo de cada fase de prueba es detectar errores que es probable que no hayan sido detectados en fases de prueba anteriores. Las pruebas de integración, por tanto, deben apuntar a detectar errores que no fueron descubiertos durante las pruebas unitarias.

Teniendo en cuenta lo anterior se definen tres objetivos prácticos de prueba cuando se está integrando el sistema A con su llamado módulo B (Leung 1990):

1. Asegurarse que A no utilizará funcionalidades que no se puedan proporcionar por B (es decir, comprobar si falta un error de función).
2. Asegurase de que A no utilizará otras funcionalidades de B si B suministra más funcionalidades que las requeridas por A (es decir, compruebe error de función adicional).
3. Asegurarse de que los estándares de interfaz entre A y B sean preservados (es decir, comprobar si hay error de interfaz).

Los tipos fundamentales de estrategias de integración son:

Integración incremental: Es donde se combinan el siguiente componente (módulo) con el conjunto de componentes que ya están probados y se va incrementando progresivamente el número de componentes a probar (Leung 1990).

Integración no incremental: Se prueba cada componente por separado y posteriormente se integran de una vez realizando todas las pruebas pertinentes(Leung 1990).

Plan de Pruebas de Integración definido por la IEEE 829 (Leung 1990)**:**

1. Identificación: Nombre de planes concretos para reconocerlos.
2. Elementos a probar: elementos del módulo a probar.
3. Enfoque: estrategia a seguir.

En el módulo de perinatales se comprobó la compatibilidad y funcionalidad de las interfaces del sistema principal con el módulo incorporado, así como las creación, lectura, escritura y consultas a la información contenida en la base de datos del sistema, la cual se carga y procesa en el módulo en cuestión. Siguiendo la estrategia de prueba se definieron los aspectos mostrados en la Tabla 11.

Tabla 11 Estrategia de Prueba de integración

Identificación	PlanInte001
Elementos a probar	Datos de las clases de AutopsiaPerinatalNeonatal, FallecidoFetoBebe, que se muestran en vistas que se cargan en el módulo, así como su interacción con la base de datos de autopsias.
Enfoque	Integración no incremental

A continuación se muestran los resultados de las pruebas de integración realizadas al módulo (Ver Figuras 21 y 21).

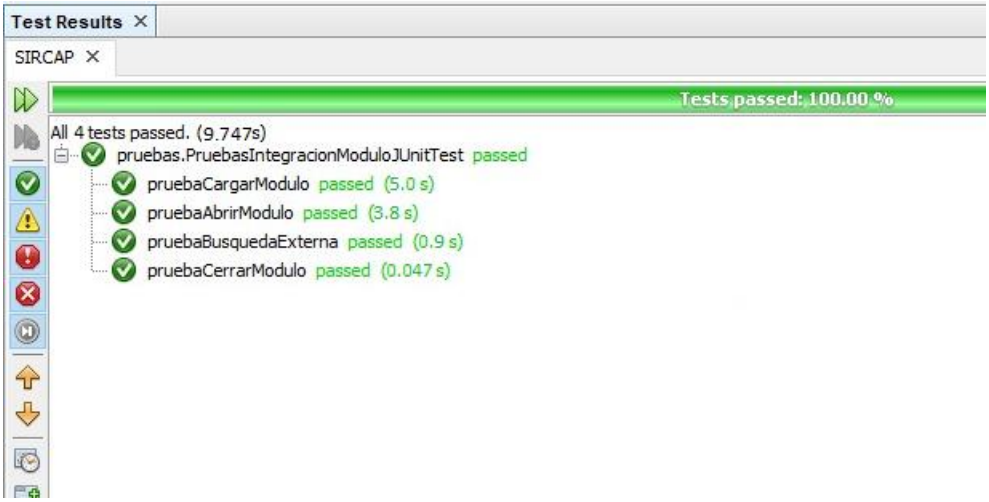

Figura 21 Pruebas de integración con interfaces del módulo.

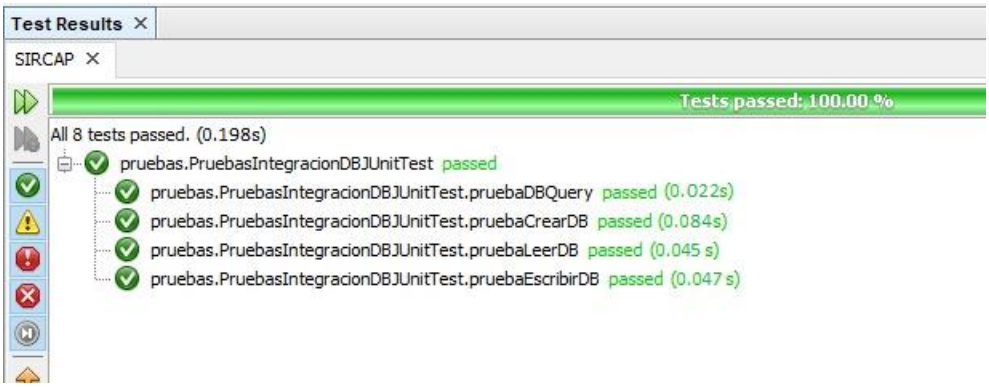

Figura 22 Pruebas de integración del módulo con la base de datos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.4 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra definidas para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención (Cillero 2020). La mismas permiten al cliente saber cuándo el sistema funciona, y que los programadores conozcan que es lo que resta por hacer" (Jeffries 2000). Estas tienen dos tipos de procedimiento para realizarlas, las llamadas pruebas alfa y beta (Ponce 2010).

Las pruebas alfa se realizaron al final de cada iteración planificada, en un entorno controlado por el equipo de desarrollo. Para ello se definieron casos de prueba por cada una de las historias de usuario planificadas en la iteración. Su utilización permitió validar las funcionalidades implementadas, registrándose los errores y problemas detectados como no conformidades, y dándosele solución antes de pasar a la próxima iteración. En las sucesivas iteraciones, además de las pruebas a las nuevas funcionalidades, se repitieron las pruebas anteriores aumentando su criticidad.

En las Tablas 12 y 13 se muestra el caso de prueba de aceptación para la historia de usuario "Adicionar fallecido perinatal", y el juego de datos para realizar la prueba.

Tabla 12 Caso de Prueba de aceptación de la HU Adicionar fallecido perinatal.

CASO DE PRUEBA DE ACEPTACIÓN			
Historia de Usuario:		HU-15: Adicionar fallecido perinatal.	
Descripción:		Permite validar el registro de los datos generales de un fallecido perinatal.	
Condiciones de ejecución:		El usuario se autentica en el sistema. En la interfaz principal, se selecciona el menú **Inicio,** luego en el menú desplegable **Módulo de Perinatal** y seguidamente **Introducir fallecido perinatal**, o alternativamente la combinación de teclas de acceso directo **CONTROL + I.**	
Escenarios de prueba:		**Flujo del escenario:**	**Resultados esperados:**
EP1	Adicionar fallecido perinatal introduciendo datos válidos.	Se introducen datos válidos para un fallecido perinatal.	Se registran los datos. El protocolo pasa de página.

		Se intenta pasar a la siguiente página del protocolo.	
EP2	Adicionar fallecido perinatal introduciendo datos no válidos.	Se introducen datos no válidos para un fallecido perinatal. Se intenta pasar a la siguiente página del protocolo.	El sistema muestra una alerta sobre los datos no válidos. Se resaltan los campos no válidos. El protocolo no pasa de página.
EP3	Adicionar fallecido perinatal dejando campos obligatorios vacíos.	Se introducen datos dejando campos obligatorios vacíos. Se intenta pasar a la siguiente página del protocolo.	El sistema muestra una alerta sobre los campos obligatorios que permanecen vacíos. Se resaltan los campos vacíos. El protocolo no pasa de página.
EP4	Adicionar fallecido perinatal previamente registrado en el sistema.	Se introducen los datos de un fallecido ya adicionado. Se intenta pasar a la siguiente página del protocolo.	El sistema muestra una alerta sobre los datos del fallecido previamente adicionado. Se resalta el número de fallecido perinatal. El protocolo no pasa de página.
EP5	Cancelar adicionar fallecido perinatal.	Se introducen algunos datos del fallecido perinatal. Se presiona el botón **Cancelar**.	El sistema muestra una alerta, solicitando confirmación para continuar. Se cierra el protocolo del fallecido perinatal.

Tabla 13 Juego de datos de prueba: HU Adicionar datos de fallecido perinatal (V: válido, I: inválido).

Escenarios de prueba	**Fallecido perinatal**	**Perinatal**	**Semanas del fallecido**	**Nombre de la madre**	**Fecha de disección**	**Ejecutado por**	**Fecha de diagnostico**	**Ejecutado por**
EP1	**FD20200001**	**Si**	**10/05/2020**	**Maria**	**10/05/2014**	**Juan**	**13/10/200**	**Miguel**
	(V)	(V)	(V)	(V)	(V)	(V)	(V)	(V)
	A20200002		**11/05/2020**					
	(V)	(V)	(V)	(V)	(V)	(V)	(V)	(V)
EP2	**EB2184**		**13/05/2020**					
	(I)	(V)	(V)	(V)	(V)	(V)	(V)	(V)

EP3			**17/05/2020**					
	(I)	(V)	(V)	(V)	(V)	(V)	(V)	(V)
	FP20200007							
	(V)	(V)	(I)	(V)	(V)	(V)	(V)	(V)
EP4	**FP20140001**	**Si**	**1/05/2020**	**Laura**	**2/06/2020**	**Alejandro**	**2/6/2020**	**Joaquin**
	(V)	(V)	(V)	(V)	(V)	(V)	(V)	(V)
EP5	**FP20200004**		**6/05/2020**					
	(V)	(V)	(V)	(V)	(V)	(V)	(V)	(V)

3.5 Beneficios del desarrollo del módulo

El análisis de los resultados de la autopsia perinatal genera múltiples beneficios mayormente orientados a mejorar la calidad del trabajo médico. Su máximo aprovechamiento se sustenta en la gestión eficiente de la información generada. El proceso de recogida de información antes de la creación del módulo de perinatales se realizaba manual por lo que no existía un máximo aprovechamiento de los datos recopilados, y el acceso y modificación de la información se realizaba de forma tediosa.

Las características del módulo propuesto pueden resumirse en los siguientes puntos:

- El procesamiento local de los datos con este módulo fortalece la capacidad de autoevaluación de la atención perinatal mediante el análisis de datos en la propia Institución asistencial.
- La solución no solo gestiona y centraliza la información referente a las autopsias perinatales, además permite mostrar la información de reportes que ayudan en la toma de decisiones en este sentido.
- Mejora la calidad de los datos asociados a la autopsia perinatal. Al estar validados en el sistema los campos de entrada de la información se garantiza la corrección de los datos a en el momento de entrar al sistema, lográndose mayor agilidad en este proceso, así como que la información esté centralizada y disponible desde la aplicación.
- Facilita la comunicación intra y extra institucional, favorece el cumplimiento de las normas, registra datos de interés legal y facilita la auditoria.

- Sirve de base para la planificación de la atención, al proveer la información necesaria para identificar la población, evaluar la asistencia, categorizar los problemas y realizar investigaciones operativas.

3.6 Conclusiones parciales

Como resultado del trabajo de implementación y las pruebas se llegaron a las siguientes conclusiones parciales:

- El uso de estándares de codificación aseguró una homogeneidad en la nomenclatura de las clases y métodos facilitando su comprensión a otros desarrolladores.
- Las pruebas de software desarrolladas demostraron la calidad y buen funcionamiento del sistema corrigiéndose la totalidad de no conformidades detectadas durante el proceso.
- Mediante el uso del módulo desarrollado se mejora la gestión de las autopsias perinatales, propiciando a la red hospitalaria cubana contar con una herramienta de apoyo en el análisis de la información generada en este proceso.

Conclusiones

- El estudio de bibliografías, la investigación científica y entrevistas a expertos en el área demostró la necesidad de un módulo de autopsias perinatales para el Sistema Informático de Registro y Control en Anatomía para la Red Hospitalaria de Cuba.
- La realización del análisis y diseño propuesto por la metodología XP, así como la incorporación de patrones de diseño, permitió sentar las bases para el desarrollo del módulo propuesto.
- La realización de pruebas de unidad y de aceptación permitieron identificar y corregir los errores presentes en el módulo desarrollado garantizando la calidad del mismo.
- Mediante el uso del módulo desarrollado se mejora la gestión de las autopsias perinatales, propiciando a la red hospitalaria cubana contar con una herramienta de apoyo en el análisis de la información generada en este proceso.

Referencias Biográficas

1. Alexander, Christopher. 1979. *The Timeless Way of Building*.
2. Alonso, Daminan Perez. 2012. *Normas y Estandares de Codificación*.
3. Avgeriou, Paris. 2005. "Architectural Patterns Revisited - A Pattern Language." In.
4. Beck, Kent, and Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change, Second Edition* (Addison Wesley Professional).
5. Bell, Douglas , and Mike Parr. 2003. *Java para estudiantes* (Pearson Editorial).
6. Blancarte, Oscar. 2018. "Data Access Object (DAO) Pattern." In.
7. Booch, Grady, James Rumbaugh, and Ivar Jacobson. 1999. *El Lenguaje Unificado de Modelado/The Unified Software Development Process* (Pearson Education).
8. Buschmann, Frank , Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 2001. *Pattern-Oriented Software Architecture, A System of Patterns* (Wiley).
9. Cap. 2019. "Anatomic pathology computer systems." In *CAP TODAY*, 55-64.
10. Casas, Sandra, and Héctor Reinaga. 2012. 'Aspectos Tempranos: un enfoque basado en Tarjetas CRC', *Sociedad Colombiana de Computación*.
11. Charvat, Jason. 2003. *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects* (John Wiley & Sons, Inc: New Jersey).
12. Cillero, Manuel. 2020. 'Mi cirscustancia digital: pruebas de aceptacion'. https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/aceptacion/.
13. Cumare, Jesús. 2020. 'Patrones en la Ingeniería de Software Category: Ingenieria de software '. https://ingsoftwarei2014.wordpress.com/category/patrones-en-la-ingenieria-de-software/.
14. Erich Gamma, Richar Helm, Ralph Johnson y John Vlissides,, and Addison Wesley. 1994. *Design Patters: Elements of Reusable Object-Oriented Software*.
15. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 2004. *Design Patterns Elements of Reusable Object-Oriented Software*.
16. Git. 2020. 'Git --distributed-even-if-your-workflow-isnt', Accessed 12-2. https://git-scm.com/.
17. Guerra, César Arturo. 2020. 'Técnicas Para la Obtención de Requerimientos', *SG Software Guru*.

18. Heffelfinger, David R. 2006. *JasperReports for Java Developers* (Packt Publishing).
19. Hurtado de Mendoza Amat, José. 2009. *Autopsia. Garantía de calidad en la medicina* (Editorial Ciencias Médicas: La Habana).
20. Hurtado de Mendoza Amat. 2014. " Autopsia. Garantía de calidad en la Medicina." In, 239. La Habana: Editorial Ciencias Médicas.
21. Hurtado de Mendoza Amat, José, and Reynaldo Álvarez Santana. 2008. "Situación de la autopsia en Cuba y el mundo. La necesidad de su mejor empleo." In *Revista Latinoamericana de Patología*, 3-8. La Habana.
22. Hurtado de Mendoza Amat, José, Reynaldo Álvarez Santana, and Israel Borrajero Martínez. 2008. 'Evaluación de la calidad de los diagnósticos premortem de causas de muerte según autopsias. Cuba, 1994-2003. Primera parte', *Revista Latinoamericana de Patología*, 46: 85-95.
23. Hurtado de Mendoza Amat, José, Teresita de J. Montero González, and Ignacio Ygualada Correa. 2013. "Situación actual y perspectiva de la autopsia en Cuba." In *Revista Cubana de Salud Pública*, 135-47. La Habana.
24. IEEE. 2004. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*.
25. Isabel, Guerra. 2019. *Libro Blanco 2019 de la Anatomía Patológica en España* (España).
26. Jacobson, Ivar, Grady Booch, and James Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software* (Pearson Educación. S.A.: Madrid).
27. Jardines Méndez, José B. 1995. 'Cuba: El reto de la atención primaria y la eficiencia en salud %J Educación Médica Superior', 9: 1-2.
28. Jeffries, Ron. 2000. "Extreme Testing. ." In.: 2004.
29. JUnit. 2020. 'JUnit 4', Accessed 02/06/2014. http://junit.org/.
30. Kendall, Kenneth E , and Julie E Kendall. 2005. *Análisis y diseño de sistemas* (Pearson Education).
31. Larman, Craig. 1999. *UML y Patrones. Introduccion al analisis y diseño orientado a objetos* (Prentice Hall: México).
32. Letelier, Patricio, and María Carmen Penadé. 2003. 'Métodologías Ágiles en el Desarrollo de Software: eXtreme Programming (XP)', Universidad Politécnica de Valencia.
33. Leung, Hareton K. N. 1990. 'A Study of Integration Testing and Software Regression at the Integration Level', *IEEE Software*: 290-301.
34. Lozančić, Ana. 2016. "Benefits of Software Testing." In.

35. Malfará, Dayvis, Diego Cukerman, Fernando Cócaro, and Juan Pablo Cassinelli. 2006. 'Testing en eXtreme Programming'.
36. Matellán Olivera, Vicente. 2004. *Compilación de ensayos sobre software libre* (Dykinson: Madrid).
37. Medell Gago, Mariana. 2004. 'AvanPat 1.1. Sistema de Información para Anatomía Patológica', Accessed 14/11/2013. http://conganat.uninet.edu/6CVHAP/autores/trabajos/T067/index.html.
38. Mendoza, Jose Hurtado de. 2019a. "Concepto de Perinatal." In, edited by Daymari Martinez Rodriguez.
39. Mendoza, Jose Hurtado de. 2019b. "que es la patologia." In, edited by Daymari Martinez Rodriguez.
40. Miguel Fernando González Pinzón, Juan Sebastián González Sanabria. 2013. 'Aplicación del estándar ISO/IEC 9126-3 en el modelo de datos conceptual entidad-relación.', Unidad Pedagogica y Tecnologica de Colombia.
41. Mitchell, John C. 2002. *Concepts in Programming Languages* (Cambridge University Press).
42. ORACLE. 2013. 'Welcome to NetBeans', Accessed 20/11/2013. https://netbeans.org/index_es.html.
43. JavaFX. 2019. 'JavaFX Scene Builder', Accessed 12-2. https://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html.
44. JavaFX. 2020. 'JavaFX Developer Home', Accessed 20/11/2013. http://www.oracle.com/technetwork/java/javafx/overview/index.html?ssSourceSiteId=ocomen.
45. Ottinger, J.B, Linwood,y Minter. 2016. 'Beginning Hibernate: For Hibernate 5'. https://books.google.com.cu/books?id=kOt6DQAAQBAJ. .
46. Ovalle S., Alfredo, Elena Kakarieka W., Ángel Correa P., María Teresa Vial P., and Carlos Aspillaga M. 2005. 'ESTUDIO ANÁTOMO-CLÍNICO DE LAS CAUSAS DE MUERTE FETAL %J Revista chilena de obstetricia y ginecología', 70: 303-12.
47. Paradigm, Visual. 2005. "Visual Paradigm Suite User´s Guide." In.
48. Park, Seung, Anil V Parwani, Raymond D Aller, Lech Banach, Michael J Becich, Stephan Borkenfeld, Alexis B Carter, Bruce A Friedman, Marcial Garcia Rojo, Andrew Georgiou, Gian Kayser, Klaus Kayser, Michael Legg, Christopher Naugler, Takashi Takashi Sawai, Hal Weiner, Dennis Winsten, and Liron Pantanowitz. 2013.

'The history of pathology informatics: A global perspective', *Journal of Pathology Informatics*, 4.

49. Pedro Arango, Leonardo Gabrera 2014a. 'Sistema Informatico de Registro y Control de Anatomia Patologica para la Red Hospitalaria de Cuba'.
50. Pedro Arango, Leonardo Gabrera , José Hurtado de Mendoza Amat. 2014b. 'Sistema informático para la gestión del conocimiento en los departamentos de Anatomía Patológica'. http://www.revinformatica.sld.cu/index.php/rcim/article/view/326.
51. Pgadmin. 2019. 'pgAdmin - PostgreSQL Tools.', Accessed 12-2. https://www.pgadmin.org.
52. Ponce, J., Domínguez-Mayo, F.J., Escalona, M.J., Mejías, M., Pérez, D., Aragón, G., Ramos, I. 2010. ' Pruebas de Aceptación en Sistemas Navegables.', *Revista Española de Innovación, Calidad e Ingeniería del Software [en linea]*, 3: 50.
53. PremKumar, LLawrence, and Praveen Mohan. 2010. *Beginning JavaFX* (Apress).
54. Pressman, Roger. 2005. *Ingeniería de Software. Un enfoque prático* (Mc Graw Hill).
55. Rouse, Margaret. 2007. 'What is integrate development environment (IDE)', Accessed 07/11/2013. http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment.
56. Salud, Organizacion Mundial de la. 2020a. "Clasificación de autopsias." In.
57. Salud, Organizacion Mundial de la. 2020b. 'Objetivos de desarrollo del milenio'. https://www.who.int/topics/millennium_development_goals/about/es/#.
58. Schmuller, Josep. 2000. *Aprendiendo UML en 24 horas* (Pearson Educación: Naucalpan de Juárez).
59. slideshare. 2020. 'Patrones de diseño'. https://pt.slideshare.net/deyanireth/diapositivas-sobre-patrones-de-diseo.
60. Sommerville, Ian. 2007. *Software Engineering* (Pearson Education).
61. Ticona Rendón, Manuel, and Diana Huanco Apaza. 2011. 'Factores de riesgo de la mortalidad perinatal en hospitales del Ministerio de Salud del Perú %J Revista Cubana de Obstetricia y Ginecología', 37: 432-43.
62. Trejos Buritica, Omar Ivan. 1999. *La esencia de la Lógica de Programación*.
63. Usaola, Macario Polo. 2020. "Patrones GRASP." In, http://www.inf-cr.uclm.es/www/mpolo/asig/0304/102/patronesgrasp.pdf. Departamento de

Tecnologías y Sistemas de Información.Escuela Superior de Informática, Universidad de Castilla–La Mancha.

Anexos

Anexo 1 Protocolo de autopsia. Sistema nervioso y aparato respiratorio. Fuente: (Hurtado de Mendoza Amat 2009).

Protocolo de autopsia perinatal

Autopsia No. _____
Hábito ext.: Cadáver de feto_____ Neonato___ Lactante___
VT __ CC__ CT__ CA __ Pie __
Peso: _____ Sexo: _____
Piel: Color_____Petequias _____Equimosis _____Punturas _____
Maceración: Desprendimiento a la presión oblicua __ Flictenas __
Color de la base:_____
Ojos_____Nariz_____Boca_____

Sistema nervioso:
Edema de cuero cabelludo: Sí __ No__ Caput sucedaneum. Sí ___ No___
Cefalohematoma: Sí_____No_____
Localización_____ Fontanelas_____
Huesos de la calota craneal. Cabalgamiento: Sí __ No __
Fracturas: Sí __ No ___
Encéfalo: Peso _____
Duramadre. Desgarros: Sí ___ No ___
Senos venosos. Trombosis: Sí ___ No ___
Leptomeninges. Transparentes____Opacas_____Hemorragia:_____
Localización:_____
Surcos y circunvoluciones _____
Cortes verticotransversales _____
Hemorragias: Sí __ No __
Malformación Sí _____ No _____ Descripción:_____
_____
Cavidades: Situs visceral. Normal ___ Situs inverso total ___
Situs inverso parcial _____
Contenido cav. peritoneal _____
Cav. pleural _____
Cav. pericárdica_____
Diafragma. Curvatura _____ Hernia_____
_____

Aparato respiratorio:
Laringe _____ Tráquea _____
Bronquios _____
Pulmones. Peso combinado _____ Normal _____
Descripción:

Anexo 2 Protocolo de autopsia. Aparato circulatorio. Fuente: (Hurtado de Mendoza Amat 2009).

Aparato circulatorio:
Corazón. Peso ____Normal ____ Foramen oval. Permeable ____
No permeable _____
Ductus arterioso. Permeable _____ No permeable _____
Drenaje venoso_____
Conexiones aurículo-ventriculares _____
Conexiones ventrículo-arteriales _____
Medidas. Perímetros valvulares:
Aorta ____ Pulmonar____Tricúspide____Mitral ____
Miocardio. VI_____VD _____ Aorta_____
Vasos umbilicales: Arterias. Sí ___ No ___Trombosis ___ Ruptura ___
Vena. Trombosis. Sí ___ No ___
Placenta Simple ____ Múltiple ____ Peso _____
Medidas ___ x ___ x ___
Inserción del cordón_____
Inserción de membranas_____
Cara fetal _____Cara materna _____
Cortes seriados. Infartos blancos _____ Infartos rojos _____
Hematoma _____
Trombosis intervellosa ____ Hemangioma ____
Longitud del cordón ____ Diámetro _____ Nudos. Sí ___ No ____
Hematoma _____
Aparato digestivo:
Faringe permeable: Sí___No___ Esófago permeable: Sí___No___
Fístula Sí___No ____
Contenido_____
Mucosa _____
Estómago contenido _____
Mucosa _____
Intestino delgado. Permeable: Sí _____ No _____
Contenido _____
Intestino grueso. Permeable: Sí _____ No _____
Contenido _____
Hígado. _____
Peso_____Color_____Consistencia_____
Hematoma_____Absceso_____
Vesícula. Permeable. Sí ___ No ___ Resto vías biliares.
Permeables: Sí___ No___
Páncreas. Peso _____ Descripción _____

PROHIBIDA SU REPRODUCCIÓN

Anexo 3 Protocolo de autopsia. Aparato urinario. Fuente: (Hurtado de Mendoza Amat 2009).

Aparato urinario:

Riñones. Peso combinado: _____Color: _____

Superficie _____

Sección: Corteza y médula _____

Cálices pelvis y uréteres. Dilatados: Sí __No __

Descripción: _____

Vejiga. Mucosa _____Contenido _____

Dilatada: Sí ___ No _

Uretra. Valva de uretra posterior. Sí ___ No ___

Anexo 4 Protocolo de autopsia. Aparato genital. Fuente: (Hurtado de Mendoza Amat 2009).

Aparato genital:

Genitales externos: Vulva _____ Pene _____

Escrotos _____

Genitales internos:

Vagina. Atresia. Sí ____ No ____ Tabicación. Sí ____ No ____

Útero. Normal _____ Malformado _____

Descripción: _____

Trompas y ovarios. Normales _____

Agenesia _____ Hipoplasia _____

Descripción: _____

Testículos. Agenesia _____

Posición _____

Próstata _____

Anexo 5 Protocolo de autopsia. Sistemas hemolinfoyético, endocrino y osteomiocarticular. Fuente: (Hurtado de Mendoza Amat 2009).

Sistema hemolinfoyético:
Bazo. Peso _____ Normal _____ Color _____
Consistencia _____
Ganglios linfáticos _____
Sistema endocrino:
Hipófisis. Peso _____ Tiroides. Peso ____ Timo. Peso _____
Color _____
Adrenales. Peso _____Hemorragia. Sí _____ No _____
Tumor. Sí _____ No _____
Sistema osteomiocarticular:
Tono muscular. Presente _____ Ausente _____
Disminuido _____
Malformaciones musculoesqueléticas _____
Cultivo: _____
Fotografía: _____
Rayos X:_____
Fecha:_____
Nombre del prosector: _____

Anexo 6 Protocolo de autopsia. Datos de la madre, fallecido, diagnóstico clínico, y anatomopatológico. Fuente: (Hurtado de Mendoza Amat 2009).

Departamento de Anatomía Patológica. Informe de autopsia perinatal

Hospital _____ Autopsia_____

Datos de la madre

Nombre:
Historia clínica:
Edad:___ años
Color piel:_____
Municipio:
Ocupación:T__E__AC__J__D__
Fecha ingreso:
Fecha egreso:
Especialidad egreso:
Paridad: G___ P___ A___
Edad gestacional:____ Semanas:____
Tipo de parto: Transpelviano___Cesárea___Aborto___
Embarazo ectópico___

Datos del fallecido

Fecha ingreso:_____
Fecha egreso:_____
Especialidad egreso:_____
Muerte: AP___IP___
Extra H___Intra H___
Edad:
Sexo:
Peso:
Apgar:

Diagnósticos clínicos:

a) Principal enfermedad o condición del feto o R/N: _____
b) Otras enfermedades o condiciones del feto o R/N:_____
c) Principal enfermedad de la madre que afecto al feto o R/N: _____
d) Circunstancias relevantes (aclare si es de la madre o del feto o R/N cuando sea necesario):

Diagnósticos anatomopatológicos:

a) Principal enfermedad o condicion en el feto o R/N:
b) Otras enfermedades o condiciones en el feto o R/N:
c) Principal enfermedad de la madre que afecto al feto o R/N:
d) Circunstancias relevantes (aclare si es de la madre o del feto o r/n cuando sea necesario):

Causa de muerte (Quebec modificado):

Observaciones:_____
_____

Firma del patólogo:_____

Anexo 7 Modelo de datos.