UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 7

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

Sistema para la gestión de la información en el estudio neuroinmunológico de proteínas del líquido cefalorraquídeo

Autores: Yosami González Vega
Yasmani Ledesma Valdés

Tutores: Dr. Alberto Juan Dorta Contreras
Ing. Alexander Rodríguez Rabelo
Ing. Leydis Hidalgo López

La Habana, junio de 2013

"Año 55 de la Revolución"

"Si los hombres de ciencia pudieran encontrar hoy día el tiempo y el valor necesario para consolidar honestamente y objetivamente su situación y las tareas que tiene por delante, y si actuaran en consecuencia, acrecentaría considerablemente las posibilidades de dar con una solución a la peligrosa situación internacional presente".

Albert Einstein

Datos de Contacto

Dr. Alberto Juan Dorta Contreras. Doctor en Ciencias de la Salud, Licenciado en Bioquímica, Profesor Titular, Investigador Titular. Facultad de Ciencias Médicas "Dr. Miguel Enríquez", Instituto Superior de Ciencias Médicas de La Habana (ISCMH). Actualmente trabaja en el Laboratorio Central de Líquido Cefalorraquídeo (LABCEL).
Correo electrónico: adorta@infomed.sld.cu

Ing. Alexander Rodríguez Rabelo: Graduado de ingeniero en Ciencias informáticas en el año 2007, profesor Asistente, Se ha desempeñado como jefe de proyecto, jefe de departamento y vicedecano de investigación y postgrado. Ha impartido las asignaturas de Matemática, Práctica Profesional, Debate Histórico Contemporáneo y Metodología de la Investigación Científica. Ha participado en los proyectos de desarrollo de *software* alas BQO y alas HIS donde se ha desempeñado como implementador, analista, administrador de la configuración y jefe de proyecto. Actualmente es el director del centro CESIM.
Correo electrónico: arodriguezra@uci.cu.

Ing. Leydis Hidalgo López: Graduada de ingeniera en Ciencias Informáticas en el año 2010. Trabajó un año de Especialista en el Centro de Desarrollo de *Software* Holguín, donde se desempeñó como líder, implementadora, analista y planificadora del proyecto Sistema de Gestión de Almacenes. Actualmente, Especialista General del Centro de Informática Médica (CESIM) en la UCI desempeñándose como analista de *software* en los proyectos de desarrollo alas BQO y alas HIS.
Correo electrónico: lhlopez@uci.cu.

Agradecimientos

En primer lugar a Fidel, por hacerme partícipe de esta gran Revolución "Con todos y para el bien de todos".

A mi familia porque sin su amor y guía no hubiese podido ser el hombre que soy hoy.

A mis profesores que hicieron de mi un estudiante consagrado, perseverante y una mejor persona.

A mis amigos que me han acompañado en las buenas y en las malas y en especial a mi compañera de tesis que hemos sido amigos inseparables desde segundo año.

A mis tutores, por su ayuda paciente y desinteresada.

A todas aquellas personas que ayudaron a la realización de este trabajo.

Por su dedicación y esmero,

¡Muchas Gracias!

Yasmani Ledesma Valdés

AGRADECIMIENTOS

A mi familia por su apoyo incondicional a cada instante.

A mis profesores que contribuyeron en mi preparación para la vida.

A mis amigos desde la niñez que han estado presente en las buenas y en las malas, en especial a mi compañero de tesis y amigo.

A mis tutores y profesores, por su ayuda paciente y desinteresada.

Yosami González Vega

Dedicatoria

A mis abuelos paternos: Emilia y Rolando por guiarme y apoyarme durante toda mi vida.

A Yuniel por darme todo su amor, compresión y paciencia en el tiempo que hemos vivido juntos.

A mi mamá, mi papá y hermanas, por la confianza depositada en mí.

A mis tías: Laura y Leticia, y mis tíos por estar presente en todo momento.

A todas aquellas personas que han contribuido de una forma u otra con mi formación profesional, a mis tutores, mis compañeros, mis profesores y mis amigos de toda la vida.

Yosami González Vega

A Fidel por ser el artífice de esta hermosa obra que es la Revolución y darme la posibilidad de cumplir este gran sueño.

A mis padres y hermano, por la formación, la confianza y el amor de siempre.

A todas aquellas personas que han contribuido de una forma u otra con mi formación profesional, a mis tutores, mis compañeros, mis profesores y mis amigos de toda la vida.

Yasmani Ledesma Valdés

Resumen

El sistema de salud pública cubana cuenta con un conjunto de centros, los cuales se encargan de brindar una atención integral a la población. Uno de estos es el Laboratorio Central de Líquido Cefalorraquídeo (LABCEL) entidad que tiene como función esencial: la asistencia médica como centro de referencia y excelencia para el estudio del Líquido Cefalorraquídeo (LCR). La presente investigación tiene como objetivo desarrollar una aplicación para informatizar el proceso de gestión de la información en el estudio neuroinmunológico de proteínas del LCR en el LABCEL.

Atendiendo a las necesidades del cliente para el desarrollo de la solución se definieron un conjunto de tecnologías y herramientas a emplear. Como Metodología de desarrollo se utiliza Rational Unified Process (RUP). Para el modelado el Lenguaje de Modelado Unificado (UML). Como herramienta de modelado Visual Paradigm. Business Process Modeling Notation (BPMN) estándar para modelado de procesos del negocio. Como gestor de bases de datos se tiene SQLite. Se usa C++ como lenguaje de programación y Qt Creator como entorno de desarrollo integrado. Además se implementa el patrón de arquitectura Modelo-Vista.

El sistema desarrollado facilitará el trabajo de los especialistas del LABCEL, dado que se dispondrá de mayor rapidez en los procesos desarrollados en el centro posibilitando con ello una mejor atención a los pacientes; además, el personal médico contará con información referente a otros análisis realizados para efectuar estudios comparativos y poder emitir diagnósticos cada vez más precisos.

Palabras clave: *barrera sangre LCR, líquido cefalorraquídeo, neuroinmunológico*.

Tabla de Contenidos

TABLA DE CONTENIDOS

Introducción

En la actualidad el desarrollo de la ciencia y las Tecnologías de la Información y las Comunicaciones (TIC's) imponen nuevos retos a la sociedad. La superación profesional, la rapidez, eficiencia y optimización de los servicios constituyen los principales objetivos a lograr en los diferentes sectores. Las TIC's con su creciente evolución permiten implementar nuevas vías, desarrollar aplicaciones y herramientas interactivas cada vez más especializadas y menos complejas, capaces de brindar una mejor respuesta a los problemas existentes.

Una de las aplicaciones sociales más frecuentes e importantes de las TIC's desde hace varias décadas ha sido en la rama de la medicina. Esto ha permitido al sector de la salud incluir métodos novedosos, sencillos y efectivos de gestión administrativa en consultas, hospitales y centros de investigación biomédica, brindando de este modo mejores servicios a los pacientes; así como, facilitando el trabajo del personal médico y administrativo.

Desde los primeros momentos del triunfo de la revolución cubana el desarrollo de la salud pública constituye un elemento que distingue el proceso revolucionario. Innumerables han sido los logros alcanzados por la medicina cubana en estos años de Revolución y los esfuerzos realizados por el estado para mantener una atención sanitaria a la altura de países desarrollados. Aunque el país no posee la infraestructura tecnológica suficiente para responder a las demandas actuales en este sector; la informatización permite lograr la utilización futura de estos servicios y la formación de recursos humanos capaces de utilizar la informática de forma eficiente en función del desarrollo socio-económico de la nación.

En la actualidad el Ministerio de Salud Pública (MINSAP) se encamina hacia la informatización de sus servicios. De vital importancia para ello ha sido la colaboración de la Universidad de las Ciencias Informáticas (UCI), la cual posee en su infraestructura diversos centros de desarrollo de *software*, entre ellos el Centro de Informática Médica (CESIM) encargado del desarrollo de sistemas de gestión para el sector.

El Sistema Nacional de Salud (SNS) cuenta con centros dirigidos al estudio científico, que utilizan tecnologías de punta para servir de apoyo a la asistencia médica. Uno de estos centros es el Laboratorio Central de Líquido Cefalorraquídeo (LABCEL), entidad que desempeña como funciones esenciales: la investigación científica, la docencia universitaria de pre y postgrado, la asistencia médica como centro de

referencia y excelencia para el estudio del Líquido Cefalorraquídeo (LCR) y la barrera sangre LCR lo que ha contribuido a colocar a sus investigadores en un sitio importante en el quehacer de los neurocientíficos cubanos y del Continente Americano. [1]

El estudio del LCR, adquirido de realizar la punción lumbar a pacientes, tiene el objetivo de detectar las más disímiles afecciones o enfermedades que con otros análisis serían difíciles de diagnosticar. El análisis del LCR es un método de bajo costo, seguro y brinda información extensa y exacta de gran ayuda diagnóstica cuando se encuentran valores alterados. Por tanto, pese al desarrollo e impacto de la imagenología en el diagnóstico de las enfermedades neurológicas, estas no son capaces hasta el momento de ofrecer la información que logra el estudio del LCR. [2]

Los avances en el estudio del LCR en los últimos años han permitido incorporar a la práctica clínica la cuantificación de proteínas y otros metabolitos, incluso de manera automatizada y sobre la base de sistemas especializados. Como elemento esencial en el análisis del LCR en las enfermedades neurológicas está el uso del diagrama de las razones de Reiber o reibergrama. Estos constituyen una herramienta fundamental para el análisis del líquido cefalorraquídeo (LCR). Son diagramas donde básicamente se analiza de forma integrada la funcionalidad de la barrera sangre LCR y la síntesis intratecal de inmunoglobulinas que, de forma aislada, no lograrían tener un mayor impacto en el diagnóstico de algunas enfermedades asociadas a determinados patrones. [2]

Actualmente en LABCEL trabaja un grupo de médicos y especialistas cubanos en el diagnóstico de las diferentes patologías relacionadas con el estudio del LCR y la barrera sangre LCR. Dichos profesionales para realizar el estudio neuroinmunólogico del LCR, utilizan el *software CSF Laboratory* en su versión 3.0. Este les permite realizar reibergramas, a partir de la cuantificación de los analitos de albúmina, IgA, IgM e IgG tanto en el LCR como en el suero para un paciente determinado.

CSF Laboratory posee ciertas limitantes, entre ellas se destacan que la forma en que los datos son gestionados resulta compleja para el especialista, al tener que conocer de aspectos propios del *software*. Por ejemplo: para adicionar una muestra el usuario debe conocer tanto el identificador del paciente como de la institución remitente y para realizar la búsqueda de un paciente el especialista debe ir buscando uno por uno, trayendo consigo que el funcionamiento de la aplicación no sea eficiente y el tiempo de respuesta no sea el más adecuado. Por otro lado, en la actualidad se continúan descubriendo como utilizar reibergramas en estudios de la síntesis intratecal de IgE, MBL, C3c y C4, los que no pueden ser cuantificados en dicho sistema. Por tanto, los valores clínicos obtenidos en el análisis de estas proteínas

solo existen en formato duro; provocando que pueda deteriorarse o perderse. Además, los procesos estadísticos relacionados con estos nuevos descubrimientos son lentos y realizados de forma manual.

Debido al entorno de desarrollo en que fue implementado, requiere de recursos en cuanto a *software* para trabajar en una PC como la máquina virtual de java; por lo que la portabilidad de dicha aplicación se hace más difícil. Al poseer normas del sistema de salud de Alemania, país en que fue implementado; presenta aspectos que no son necesarios para los especialistas cubanos u otros que no posee y son fundamentales para gestionar los datos del paciente analizado. Al tener *CSF Laboratory* la característica de ser un *software* de código cerrado, no se puede acceder a él para realizarle modificaciones o agregar nuevas funcionalidades con el objetivo de resolver dichas limitantes.

Basado en lo antes expuesto se tiene como **problema a resolver**: ¿Cómo facilitar la gestión de la información de los procesos desarrollados en el estudio neuroinmunológico del líquido cefalorraquídeo a partir del análisis de los diagramas de Reiber?

En correspondencia con el problema, el **objeto de estudio** lo constituye el proceso de gestión de la información en el estudio neuroinmunológico del líquido cefalorraquídeo.

El **campo de acción** está centrado en el proceso de gestión de la información en el estudio neuroinmunológico del líquido cefalorraquídeo a partir del análisis de los diagramas de Reiber.

Para dar solución al problema antes mencionado se define como **objetivo general**: Desarrollar un sistema informático que permita la gestión de la información de los procesos desarrollados en el estudio neuroinmunológico del líquido cefalorraquídeo.

Para dar cumplimiento al objetivo general se han derivado las siguientes **tareas:**

1. Valorar las tendencias actuales de los sistemas de gestión de la información en el estudio neuroinmunológico del líquido cefalorraquídeo para apoyar el desarrollo de la investigación.
2. Definir la arquitectura de *software* adecuada para la implementación del sistema.
3. Desarrollar los artefactos necesarios según la metodología de *software* definida correspondientes a las fases Negocio, Requisitos, Análisis y Diseño e Implementación.
4. Implementar una solución que permita la gestión de la información de los procesos desarrollados en el estudio neuroinmunológico del líquido cefalorraquídeo.

Con el desarrollo de la investigación se esperan los siguientes beneficios:

1. Perfeccionar los procesos que se llevan a cabo en el estudio neuroinmunológico del líquido cefalorraquídeo en el Laboratorio Central de Líquido Cefalorraquídeo.
2. Disponer de una herramienta que satisfaga las necesidades reales y actuales de los profesionales de la salud en el Laboratorio Central de Líquido Cefalorraquídeo, mejorando las condiciones de trabajo de los especialistas y la atención a los pacientes.
3. Contar con información referente a otros análisis realizados a pacientes para efectuar estudios comparativos y poder emitir diagnósticos cada vez más precisos.
4. Mejorar el control y seguimiento de las enfermedades neuroinmunológicas en los pacientes al contar con la información disponible y correctamente estructurada.

Los **métodos teóricos** utilizados para cumplir con las tareas a desarrollar son:

- Análisis histórico-lógico: fue de gran importancia para elaborar la fundamentación teórica de la investigación, permitiendo estudiar lo más relevante en el plano teórico acerca del comportamiento de la barrera sangre-LCR, las metodologías, herramientas y tecnologías que se utilizan para el desarrollo de la aplicación y para el estudio del de software existentes en el mundo y en Cuba.
- Analítico-sintético: este método fue utilizado en todo el proceso investigativo permitiendo descomponer todo el problema en varias partes que posibilitaron una mejor comprensión del mismo, Se usa para analizar la bibliografía encontrada referente al tema en cuestión y se sintetizan los aspectos más importantes para la investigación.
- Inductivo-Deductivo: se utilizó para el planteamiento del objetivo y la extracción de las ideas fundamentales para la elaboración y fundamentación del trabajo de diploma.

Los **métodos empíricos** utilizados para obtener información sobre el objeto de estudio son:

- Entrevista: se usa a través de la realización de entrevistas a los trabajadores del LABCEL con el objetivo de llegar a un acuerdo sobre lo que el cliente quiere que tenga el producto y recopilar toda la información necesaria para el desarrollo del trabajo.
- Observación: se realizaron visitas al LABCEL, para hacer un registro visual de los procesos que se llevan a cabo en la actualidad en el centro.

A continuación se presenta un breve resumen de los capítulos en que fue distribuido el desarrollo de la investigación:

Capítulo 1: Fundamentación teórica del estudio neuroinmunológico de proteínas del líquido cefalorraquídeo. Este capítulo expone los conceptos asociados al dominio del problema planteado que ayudarán a familiarizarse con el entorno en el que se manifiesta la investigación. Se realiza una valoración de los principales sistemas informáticos desarrollados que gestionan información sobre el LCR. Además, se describen las principales herramientas y tecnologías definidas para el desarrollo de software a utilizar.

Capítulo 2: Características del sistema de gestión de la información en el estudio neuroinmunológico de proteínas del líquido cefalorraquídeo a desarrollar. Este capítulo describe el flujo actual de los procesos que se desarrollan en el LABCEL. Se realiza el modelado de los procesos que se llevan a cabo en el estudio, la cual es acompañada de su descripción para un mayor entendimiento. Es presentada la propuesta del sistema, compuesta por los requisitos funcionales y no funcionales definidos a partir de las necesidades del cliente.

Capítulo 3: Diseño e Implementación del sistema de gestión de la información en el estudio neuroinmunológico de proteínas del líquido cefalorraquídeo. Este capítulo explica los aspectos referentes a la arquitectura de software, definiéndose el patrón arquitectónico a utilizar. El sistema es modelado haciendo uso de los Diagramas de Clases de Análisis y Diagrama de Clases del Diseño de los casos de uso del sistema, se muestra el modelo de datos y se modelan además los diagramas de interacción. En el mismo, se presenta el diagrama de componentes y el diagrama de despliegue. Además, se define el estándar de codificación a adoptar.

Capítulo 1 Fundamentación teórica del estudio neuroinmunológico de proteínas del líquido cefalorraquídeo

Este capítulo es el resultado de la búsqueda y análisis de la información relacionada con el objeto de estudio, con el propósito de profundizar en los conceptos que brindan soporte a la investigación. Se realiza el análisis del estado del arte del tema tratado y se exponen las características de las diferentes tecnologías y herramientas propuestas para dar solución al problema.

1.1 Conceptos básicos asociados al dominio del problema

Para un mejor entendimiento del problema se han definido una serie de conceptos relacionados con el mismo. Los que serán explicados a continuación:

1.1.1 Neuroinmunología

La Neuroinmunología es una ciencia que se deriva de sus ciencias matrices: la neurología y la inmunología y puede marcarse sus inicios en la década de los setenta en el pasado siglo, cuando fue posible ampliamente la cuantificación de las inmunoglobulinas mayores en el líquido cefalorraquídeo. Para comprender las bases moleculares de estas ciencias es necesario conocer los conceptos actuales de barrera hematoencefálica y síntesis intratecal de inmunoglobulinas. La barrera hematoencefálica es un equilibrio físico-químico de transporte bidireccional restringido, donde las moléculas se difunden de acuerdo con su masa molecular. Existen proteínas que se difunden de la sangre al líquido cefalorraquídeo y otras se sintetizan en éste. Las inmunoglobulinas se difunden hacia el líquido cefalorraquídeo y se sintetizan en él. [3]

1.1.2 Líquido cefalorraquídeo

El líquido cefalorraquídeo, conocido como LCR, es un líquido de color transparente, que baña el encéfalo y la médula espinal. Circula por el espacio subaracnoideo, los ventrículos cerebrales y el canal medular central. [4]

El líquido cefalorraquídeo puede enturbiarse por la presencia de leucocitos o la presencia de pigmentos biliares o aumento de su contenido proteico. Numerosas enfermedades alteran su composición, su estudio es importante y con frecuencia determinante en las infecciones meníngeas, en carcinomas y hemorragias. También es útil en el estudio de las enfermedades desmielinizantes del sistema nervioso central o periférico. [4]

El líquido cefalorraquídeo tiene tres funciones vitales muy importantes [5]:

- Mantener flotante el encéfalo, actuando como colchón o amortiguador, dentro de la sólida bóveda craneal. Por lo tanto, un golpe en la cabeza moviliza en forma simultánea todo el encéfalo, lo que hace que ninguna porción de éste sea contorsionada momentáneamente por el golpe.
- Sirve de vehículo para transportar los nutrientes al cerebro y eliminar los desechos.
- Fluir entre el cráneo y la médula espinal para compensar los cambios en el volumen de sangre intracraneal (la cantidad de sangre dentro del cerebro), manteniendo una presión constante.

1.1.3 Reibergrama

Reibergrama o diagrama de las razones de Reiber constituye una herramienta fundamental para el análisis del líquido cefalorraquídeo (LCR). Se basan en la relación hiperbólica entre la razón albúmina (QAlbúmina) y las razones de inmunoglobulinas (QIg).

Los analitos que deben cuantificarse para confeccionar un reibergrama son albúmina como proteína marcadora y otras proteínas como IgA, IgM, IgG, IgE, MBL, C3c y C4 tanto en LCR como en el suero. La razón LCR/suero (Q) es el cociente obtenido de la división entre la concentración del analito en LCR y en suero. [6]

El reibergrama consta de las siguientes partes: [7]

- La curva superior hiperbólica más fuerte representa la línea de discriminación entre la fracción de inmunoglobulinas proveniente de la sangre y la fracción proveniente del cerebro.
- Valores por encima de esa línea indica que hay síntesis intratecal de IgG, IgA o IgM. Las líneas discontinuas indican en qué medida ha ocurrido la síntesis intratecal como fracción intratecal (FI IgG, FIIgA o FI IgM) expresada en % con 20, 40, 60 y 80% del total de inmunoglobulina medida en el LCR., donde la línea de discriminación es la referencia del 0% de síntesis intratecal.
- También el reibergrama brinda información sobre la barrera sangre LCR; la primera barra vertical (Qalb= 5) indica el límite superior normal hasta los 15 años, la segunda (Qalb= 6,5) hasta los 40 años y la tercera (Qalb= 8) hasta los 60 años. Valores superiores a éstos, de acuerdo con la edad, indican una disfunción de la barrera sangre LCR.

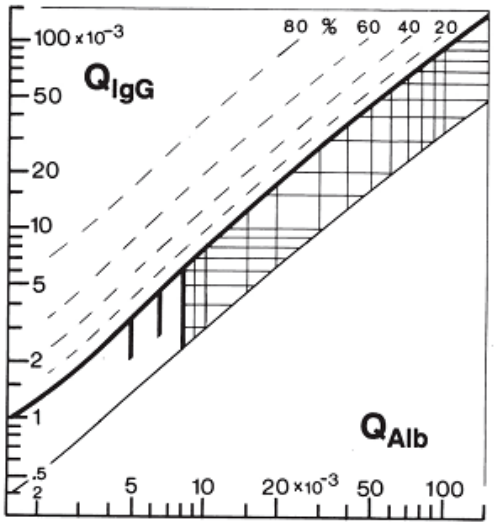


Figura 1 Reibergrama [7]

La función hiperbólica general:[2]

$Qlg = a/b\sqrt{(Qalb^2 + b^2)}$ –c

Donde a/b, b^2 y c se han reportado para Q (lim), Qmedia y Qbajo del rango de referencia para la IgG, IgA e IgM derivada de la sangre en LCR.

Los valores superiores límites Q (lim) de los rangos de referencia del reibergrama se describen como:

1. $Qlim(IgG) = 0.93\sqrt{(Qalb^2 + 6x10^{-6}) - 1.7x10^3}$
2. $Qlim(IgA) = 0.77\sqrt{(Qalb^2 + 23x10^{-6}) - 3.1x10^3}$
3. $Qlim(IgM) = 0.67\sqrt{(Qalb^2 + 120x10^{-6}) - 7.1x10^3}$

Ventajas que ofrece el Reibergrama [2]

- El reibergrama permite conocer las condiciones de la barrera sangre LCR y si hay síntesis de inmunoglobulinas.
- Para el laboratorio ofrece la ventaja de poder comprobar parte del aseguramiento de la calidad del análisis y puede sugerir el neurólogo o el clínico la posibilidad de un análisis adicional para llegar a un diagnóstico.
- Al especialista clínico si se le ofrecen otras informaciones adicionales como los antecedentes y síntomas del enfermo, puede tener la posibilidad de descartar un grupo de enfermedades, saber

que se está en presencia de una infección oportunista, una tuberculosis cerebral o una neuroborreliosis.

- La utilización del reibergrama también podría contribuir a conocer la causa de la enfermedad en el caso de que esté producida por un microorganismo (con otras pruebas adicionales como el índice de anticuerpo) que provoca el proceso inflamatorio.
- Además, se podrá detectar la presencia de un tumor metastásico intratecal, monitorizar la eficacia de terapias y el curso de una enfermedad, así como saber el origen orgánico cerebral de síntomas psiquiátricos, la detección temprana de una infección posquirúrgica o el pronóstico clínico resultado de una hipoxia o infarto cerebral a partir de pistas que se abren cuando se realiza el reibergrama.

1.2 Aplicaciones y tendencias actuales

La evolución en el mundo del software ha llevado a una gran competencia en cuanto a los sistemas informatizados. Los sistemas para el estudio del LCR no están libres de esta competitividad y se han desarrollado de maneras muy diversas, lo que ha permitido la creación de productos de muy buena calidad y un alto grado de compromiso. A continuación se hace una exposición centrada en los sistemas que apoyan el estudio del LCR.

1.2.1 Ámbito internacional

***CSF Laboratory* 3.7 (LCR Laboratorio 3.7)**

La herramienta *CSF Laboratory* [8] fue desarrollada por el grupo empresarial *TeamRoom Software* basado en el conocimiento creado en colaboración con el Prof. Hansotto Reiber de Göttingen, Alemania, el experto líder a nivel internacional en el análisis del LCR. *CSF Laboratory* está disponible en los idiomas alemán, inglés, francés y español.

Características principales del programa:

- Aplicación de escritorio para apoyar la evaluación y la interpretación del líquido cefalorraquídeo (LCR) relacionando los parámetros de laboratorio.
- Para archivar y conservar los datos, el software se suministra con una base de datos SQL.
- No realiza el análisis gráfico de múltiples muestras en una misma interfaz, haciendo más tedioso el trabajo a la hora de establecer comparaciones.
- Evalúa e interpreta los datos disponibles.

- El usuario puede cambiar cualquiera de los resultados del sistema y puede entrar sus propias evaluaciones como texto libre.
- Puede dar formato a los resultados en un documento imprimible o un archivo PDF y activar la impresión.
- Los datos de proteínas se visualizan con Reibergramas.
- La aparición de una disfunción de la barrera sangre LCR puede ser inmediatamente reconocida con la inspección de un Reibergrama.

Esta solución informática fue rechazada como posible vía para darle solución al problema planteado en el presente trabajo debido a las siguientes razones:

- Es un software distribuido bajo licencia privativa por lo que impide el acceso al código fuente para su modificación.
- Permite el análisis de un solo paciente.
- No permite cuantificar las nuevas proteínas IgE, MBL, C3c y C4.
- No se corresponde con las políticas de independencia tecnológica que lleva a cabo Cuba, a través del uso de software libre para el desarrollo de aplicaciones.
- No es multiplataforma, esto trae consigo que el usuario no pueda ejecutarlo utilizando cualquier sistema operativo. Sólo funciona en la plataforma Windows hasta su versión XP.
- Presenta aspectos, para la recopilación de datos de un análisis especial, que no son necesarios para los especialistas cubanos.

CSF Research Tool – Reibergrams 4.0

La herramienta *CSF Research Tool* [9] fue desarrollada por Werner Albaum junto a Hansotto Reiber, profesor de la Universidad de Göttingen (Alemania). Esta herramienta permite la evaluación numérica y gráfica de grupos de pacientes para su uso en los estudios clínicos y control de terapia.

Características principales del programa:

- Realiza el análisis gráfico de múltiples pacientes en una misma vista.
- Cálculo de funciones matemáticas para el análisis de LCR de un solo paciente.
- El diagnóstico y las interpretaciones estadísticas de los datos en Reibergramas.
- El cálculo de estadísticas para síntesis intratecal de inmunoglobulinas en los grupos de pacientes.

- La caracterización de niveles de importancia para trastornos de la barrera sangre LCR patológico o síntesis intratecal de proteínas.

Los datos procesados en esta herramienta permiten:

- Crear y seleccionar los juegos de datos para la evaluación gráfica en los Diagramas de LCR / suero.
- Cálculo estadístico para los trastornos de la barrera sangre LCR.
- Cálculo del nivel de importancia por la comparación con el grupo de control (t-Test).

Esta solución informática fue rechazada como posible vía para darle solución al problema planteado en el presente trabajo debido a las siguientes razones:

- Es un *software* distribuido bajo licencia privativa, por lo que impide el acceso al código fuente para su modificación.
- Permite el análisis de varios pacientes.
- No se corresponde con las políticas de independencia tecnológica que lleva a cabo Cuba, a través del uso de *software* libre para el desarrollo de aplicaciones.
- No permite cuantificar las nuevas proteínas IgE, MBL, C3c y C4.
- No es multiplataforma, esto trae consigo que el usuario no pueda ejecutarlo utilizando cualquier sistema operativo. Sólo funciona en la plataforma Windows hasta su versión XP.
- No posee una base de datos que permita persistir los datos para futuros estudios.

1.2.2 Ámbito nacional

Neuroinmunolab

La herramienta Neuroinmunolab [10] es un sistema de menús tipo “pull-down” que interrelaciona, de modo fácil y amigable para el usuario, aspectos de tratamientos de base de datos, procesamiento aritmético, procesamiento de gráficos y toma de decisiones, con sistemas de ayuda “on-line”.

El sistema está hecho en Turbo Pascal, versión 5.5 y cuenta con un manual de usuario disponible en inglés y español, con bibliografía actualizada y más de setenta referencias sobre el tema. Posee los siguientes requerimientos técnicos:

- IBM AT compatible
- 640 KB de memoria interna
- Un disco duro

- Tarjeta gráfica VGA
- Sistema operativo MSDOS versión 3.3 o superior

Opciones fundamentales del sistema:

- Manejo de base de datos: el usuario tiene la posibilidad de crear una base de datos nueva, abrir una ya creada, listar los elementos, actualizar o adicionar un paciente. Es compatible con el sistema de bases de datos DBASE PLUS lo que permite que una base de datos creada para otra aplicación pueda ser utilizada en la nuestra o viceversa.
- Resolución de fórmulas: realiza los cálculos de más de veinte fórmulas, entre los que se encuentran los índices, razones, además de las fórmulas Tibbling y Link, Tourtellotte, Schuller, Reiber Felgenhauer y Oehman, entre otras.
- Gráficos: salida gráfica de los resultados por pantalla o por impresor.

Elaboración de un reporte por paciente y por fórmula, que muestra los resultados y en base a ello da un diagnóstico o sugiere la evaluación de otras fórmulas para obtener más información al respecto, y en algunos casos ofrece recomendaciones terapéuticas.

Esta aplicación fue desarrollada en el año 1993 por el Instituto Central de Investigaciones Digitales (ICID) en coordinación con el Hospital Pediátrico de San Miguel del Padrón. A pesar de la labor dedicada para su fabricación actualmente no se cuenta con esta, sólo existen publicaciones científicos-técnicas que exponen sus características y funcionalidades fundamentales. La pérdida tuvo lugar cuando se realizaron las migraciones en Cuba al sistema operativo Windows, que no se tuvieron en cuenta los procedimientos de seguridad para salvar la información referente al mismo. Posteriormente dejó de funcionar y aplicarse en instituciones hospitalarias.

1.3 Herramientas y tecnologías a utilizar

La creciente informatización de los procesos productivos y sociales, ha traído consigo que las organizaciones y empresas requieran cada vez más de *software* confiable y de alta calidad. Por esta razón se debe hacer un profundo análisis para poder seleccionar correctamente todas las herramientas que se van a utilizar en el desarrollo de la solución, siempre pensando en las necesidades del cliente y tratando de utilizar las tecnologías más novedosas.

1.3.1 Metodología de desarrollo

Una metodología es un proceso que engloba procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de *software*. Se va indicando paso a paso lo que se debe hacer para lograr un producto informático. Además se debe especificar las personas que van a participar en el proceso así como el papel que van jugar en él.

Existen dos tipos de metodologías: las Robustas y las Ágiles, entre los tipos de metodologías Robustas se encuentran: *Rational Unified Process* (RUP), *Microsoft Solutions Framework* (MSF) y Métrica 3.0; dentro de las metodologías Ágiles se encuentran: *Extreme Programming* (XP), *Scrum y Feature Driven Development* [11].

A continuación se detallan las características de la metodología más utilizada por cada tipo, para de ellas seleccionar la más adecuada para que guie el desarrollo de la aplicación.

Proceso unificado de desarrollo

Proceso unificado de desarrollo (RUP por sus siglas en inglés) [12] es una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo) en una organización o equipo de desarrollo de *software*. Junto con el Lenguaje Unificado de Modelado (UML por sus siglas en inglés), constituye una de las metodologías más utilizadas para el análisis, implementación y documentación de sistemas orientados a objetos.

Es un modelo de *software* que permite el desarrollo de *software* a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Este proceso de desarrollo de software lo conforman el conjunto de actividades necesarias para trasformar los requisitos funcionales de un usuario en un producto de *software*.

Los que sustentan el proceso de desarrollo de *software* son: el proyecto, las personas, el producto y el proceso, existe una estrecha relación entre ellas. Es conocido como las cuatro P en el desarrollo del *software*. RUP define para cada etapa: el flujo de trabajo, los trabajadores que intervienen, las actividades que realizan y los artefactos que se necesitan o producen. Su meta es asegurar la producción de *software* con la más alta calidad, que cumpla con las necesidades de los usuarios dentro del cronograma planeado y la inversión prevista.

Características principales de RUP:

- Dirigido por casos de uso

- Centrado en una arquitectura
- Iterativo e incremental

Ventajas que aporta RUP:

- Desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del *software* mediante una gestión sistemática de los riesgos.
- Producción de *software* que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible.
- Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para las actividades críticas.
- Admite una definición acertada del sistema en un inicio para hacer innecesarias las reconstrucciones parciales posteriores.

La metodología RUP divide en cuatro fases el proceso de desarrollo de *software*:

- Inicio: el objetivo o hito de esta fase es determinar la visión del proyecto identificando y priorizando los riesgos más importantes y, estimando el proyecto de manera aproximada.
- Elaboración: en esta fase el hito es determinar la línea base de arquitectura. Se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema.
- Construcción: en esta fase el hito es llegar a obtener la capacidad operacional inicial. Al final de la misma, el producto contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión, aunque puede que no esté completamente libre de defectos
- Transición: el hito de esta fase es llegar a obtener una primera versión del proyecto. La fase de transición conlleva actividades como la fabricación, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren tras la entrega.

Programación extrema

La metodología (XP por sus siglas en inglés) [11] es una de las más populares actualmente. Sus principales objetivos son muy sencillos de entender, primeramente se trata de dar al cliente el *software* que él necesita y cuando lo necesita, por tanto se debe responder muy rápido a las necesidades del cliente, por tanto se tiene mucho coraje para enfrentar los cambios, incluso cuando sean a finales del

ciclo de programación. Como segundo se tiende a fomentar grandemente el trabajo en equipo, tanto los jefes de proyectos, los clientes y los desarrolladores son parte del grupo y están fuertemente involucrados en el desarrollo de *software*.

Características principales de XP:

- Desarrollo iterativo e incremental
- Pruebas unitarias continuas
- Frecuente interacción del equipo de programación con el cliente o usuario
- Corrección de todos los errores antes de añadir nueva funcionalidad
- Simplicidad en el código

Ventajas que aporta XP:

- Apropiado para entornos volátiles
- Planificación más transparente para los clientes, conocen las fechas de entrega de funcionalidades
- Permite definir en cada iteración cuales son los objetivos de la siguiente
- Permite tener realimentación de los usuarios muy útil
- La presión está a lo largo de todo el proyecto y no en una entrega final

La metodología que más se adapta a las condiciones de la aplicación que se quiere desarrollar es RUP, puesto que recoge un grupo de buenas prácticas de muchas otras metodologías. Es un proceso que define de manera ordenada las tareas a desarrollar. La documentación que se genera es de gran apoyo a las siguientes iteraciones que se van a desarrollar en un futuro. Permitirá lograr un sistema robusto, trabajar con precisión, calidad y perfeccionar el *software* en cualquier momento del desarrollo. Al decidir que el proceso de investigación fuese iterativo e incremental se pueden ir obteniendo versiones de cada iteración consiguiendo que se minimicen los riesgos al momento de implementar la aplicación. Es aplicable tanto a pequeños proyectos (como el del presente trabajo de diploma), así como para grandes proyectos de varios años de duración.

Proceso de negocio

Un proceso de negocio es un conjunto de tareas relacionadas de forma lógica, llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas [13].

Gestión de procesos de negocio

Gestión de Procesos de Negocio (BPM por sus siglas en inglés), consiste en un conjunto de herramientas, tecnologías, técnicas, métodos y disciplinas de gestión para la identificación, modelación, análisis, ejecución, control y mejora de los procesos de negocio. Las mejoras incluyen tanto cambios de mejora continua como cambios radicales [14].

Notación utilizada para modelar los procesos del negocio

Notación para el Modelado de los Procesos del Negocio (BPMN por sus siglas en inglés) es el estándar más reciente para modelado de procesos del negocio, en donde se presentan gráficamente las diferentes etapas del proceso del mismo. La notación ha sido diseñada específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes. Expresa los procesos de negocio en un único Diagrama de Procesos de Negocio (BPD por sus siglas en inglés). BPMN permite hacer un mejor uso de la gestión de procesos del negocio. BPMN ha sido desarrollado para proveer a los usuarios de una notación de uso libre [15].

Lenguaje Unificado de Modelado

UML es el lenguaje de modelado más conocido y utilizado en la actualidad. Es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Las propiedades que han hecho de UML un estándar son: la concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actual y futura. Las estructuras que soporta tienen sus fundamentos en las tecnologías orientadas a objetos, tales como: clases, componentes y nodos. Representa el comportamiento del sistema a través de casos de uso, diagramas de secuencia y de colaboración. [16]

Uno de los objetivos de este modelado visual es ser independiente del lenguaje de implementación, de tal forma que los diseños realizados se pueden implementar en cualquier lenguaje de programación que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos). Con este método formal de modelado se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa. [17]

1.3.2 Lenguaje de programación

Para posibilitar la comunicación entre las computadoras y los humanos se hace necesario de la existencia de los lenguajes de programación para lograr tal interactividad. Un lenguaje de programación está conformado por una serie de reglas sintácticas y semánticas que serán utilizadas por el programador y a través de las cuales creará un programa o subprograma; las instrucciones que forman dicho programa son conocidas como código fuente. [18]

A continuación se realizará una caracterización de los lenguajes más utilizados actualmente.

C++

C++ [19] es un lenguaje de programación popular, utilizado actualmente para resolver diferentes tipos de problemas desde los más simples hasta los más complejos, su código puede ser copilado en varias plataformas. Incorpora una nueva característica que permite aumentar la eficiencia en cuanto a la llamada de funciones, dicha característica es la posibilidad de escribir funciones en línea (*inline*); *inline* es una petición al compilador para sustituir la llamada a la función directamente por su código, es decir, genera un código en línea.

Algunas de las características de C++ son: [20]

- Versatilidad: C++ es un lenguaje de propósito general, por lo que se puede emplear para resolver cualquier tipo de problema.
- Portabilidad: el lenguaje está estandarizado y un mismo código fuente se puede compilar en diversas plataformas.
- Eficiencia: C++ es uno de los lenguajes más rápidos en cuanto a ejecución.
- Herramientas: existe una gran cantidad de compiladores, depuradores, librerías, etc.

C SHARP (C#)

C# [21] es el lenguaje de programación que Microsoft desarrolló principalmente para la plataforma .Net. Para su creación se usaron conceptos de C, C++,Smalltalk, Modula 2 y Java. El código de C# se compila como código administrado, lo cual significa que utiliza los beneficios de los servicios de *Common Language Runtime*, estos servicios incluyen el idioma interoperabilidad, la mejora de la seguridad, el apoyo y la mejora de versiones. C# es un lenguaje de programación orientado a objetos que coge las mejores características de los lenguajes preexistentes como Visual Basic, Java o C++ y las combina como uno solo.

Algunas de las características de C Sharp son:

- Moderno: mejora la productividad en el desarrollo de *software*, incorpora características del estado del arte de los lenguajes actuales.
- Simple: permite una sintaxis sencilla y elegante, evita la utilización de punteros, la gestión de memoria, la validación de límites de arreglos.
- Poderoso: permite el desarrollo de código seguro y no seguro.
- De propósito general: puede ser utilizado para la construcción de aplicaciones Web, aplicaciones de escritorio, servicios Web, aplicaciones para celulares y componentes.
- Totalmente orientado a objetos.

Ventajas:

- Incorpora las características de un lenguaje de última generación y está en continuo desarrollo.
- Concepto formalizado de los métodos get y set, con lo que se consigue código mucho más legible.
- Gestión de eventos es mucho más limpia.

Java

Java [22] es un lenguaje de programación desarrollado por Sun Microsystems a principios de los años 90; soporta las tres características del paradigma de programación orientado a objetos: herencia, encapsulamiento y polimorfismo. Trabaja con sus datos como objetos y con interfaces a esos objetos. La tecnología java está constituida por dos componentes básicos: el lenguaje java y su plataforma, la máquina virtual de java (Java Virtual Machine), fue diseñado para construir *software* altamente seguro, proporciona muchas comprobaciones en compilación y en tiempo de ejecución.

Algunas de las características de Java son:

- Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable
- Proporciona un conjunto de clases potente y flexible
- Facilita la utilización de aplicaciones que se pueden incluir directamente en páginas Web(applets)
- Es gratis, universal y de fácil distribución

Para la selección del lenguaje de programación C++ se tuvo en cuenta las características de los lenguajes antes mencionados y alguna de las restricciones del cliente como la portabilidad, facilidad de instalación, rapidez, consumo mínimo de recursos, entre otras. Con el uso de este lenguaje se garantiza la portabilidad, pues es un lenguaje estandarizado y le facilita al usuario que los ejecutables obtenidos funcionan en cualquier máquina (utilizando las bibliotecas y funciones estándar). Permitiendo la eficiencia, uniformidad, comprensión, flexibilidad y reusabilidad del código a la hora de trabajar. El resultado de una compilación en C++ es eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel.

1.3.3 Entorno de desarrollo integrado

Los Entornos de Desarrollo Integrado (IDE por sus siglas en inglés) [23] generalmente están compuestos por un compilador, un depurador, un editor de código y un constructor de interfaz de usuario, son creados para el desarrollo de aplicaciones en un solo lenguaje de programación; sin embargo, existen algunos, en los que se puede desarrollar en más de un lenguaje. Un IDE no es más que un conjunto de herramientas de desarrollo de *software*. Cuando el lenguaje es orientado a objetos, los IDEs incluyen navegador de clases, inspector de objetos y un diagrama de jerarquías de clases. En los siguientes acápites se dará una breve descripción de algunos de los IDEs que permiten el desarrollo de aplicaciones en C++.

Qt Creator

Qt Creator 2.5.0 es un entorno de desarrollo integrado [24] multiplataforma adaptado a las necesidades de los desarrolladores de Qt para desarrollar aplicaciones en C++ de manera sencilla y rápida. Está creado para el desarrollo de aplicaciones con las bibliotecas Qt, utilizando su versión 2.5.0. Tiene integrado los sistemas de control de versiones más populares tales como Subversion, CVS, Perforace y Mercurial. Provee soporte para la construcción de aplicaciones de escritorio y dispositivos móviles.

Las características principales de Qt *Creator*:

- Desarrollar aplicaciones Qt rápida y fácilmente con asistentes de proyectos, así como acceder rápidamente a los últimos proyectos y sesiones.
- Cuenta con dos editores visuales, Qt *Designer* para diseñar interfaces de usuario a partir de Qt *Widgets*, y Qt Quick *Designer* para el desarrollo de interfaces de usuario animadas con el lenguaje JavaScript.

- Contiene un sofisticado editor de código que proporciona completamiento de código y ayuda de contexto para el lenguaje C++ y JavaScript.
- Generar, ejecutar e implementar proyectos de Qt dirigidos a plataformas móviles y de escritorio, tales como Microsoft Windows, Mac OS X, Linux, Symbian, Android y Maemo.
- Acceder fácilmente a la documentación mediante el uso de la ayuda contextual integrada de Qt
- Este se encuentra bajo licencia libre además de presentar la gran ventaja de ser multiplataforma.

Monodevelop

Monodevelop es un entorno de desarrollo [25] libre y gratuito, diseñado sobre todo para C. Incluye manejo de clases, ayuda incorporada y autocompletado de código y un depurador integrado. Puede ejecutarse en las distintas distribuciones de Linux y Mac.

C++ Builder

C++ Builder es un entorno integrado de desarrollo [26] bastante completo, intuitivo, fácil de manejar y eficiente, que permite el desarrollo rápido de aplicaciones en entornos MS Windows.

Características fundamentales de Borland C++ Builder:

- Utiliza el lenguaje de programación orientado a objetos C++ (como su nombre indica), con todas las ventajas que supone el uso de la OO.
- Permite realizar programación visual y programación dirigida por eventos.
- Características avanzadas de IDE: editor sensible a la sintaxis, ayuda sensible al contexto, inspector de objetos, compilador y depurador integrado, etc.

Microsoft Visual Studio C++

Visual C++ es el entorno integrado de desarrollo [27] de Microsoft proporciona un entorno de desarrollo eficaz y flexible para la creación de aplicaciones basadas en Microsoft Windows y Microsoft .NET. Se puede usar como sistema de desarrollo integrado o como conjunto de herramientas individuales. Visual C++ ofrece las siguientes ventajas para el desarrollador profesional.

- Creación de aplicaciones con bibliotecas galardonadas
- Creación de aplicaciones y componentes conectados a .NET altamente adaptados

- Creación de aplicaciones y componentes no administrados y basados en Windows altamente adaptados
- Traslado del código C++ existente a .NET de forma granular y a un ritmo autodefinido

Se decide utilizar como IDE de desarrollo Qt Creator ya que es una amplia plataforma de desarrollo que contiene clases, librerías y herramientas para la construcción de aplicaciones de interfaz gráfica en C++ que pueden operar en varias plataformas. Con Qt pueden desarrollarse aplicaciones gráficas y que operen mejor que las nativas con el uso de librerías propias y con la introducción de nuevas tecnologías como OpenGL. Dispone de un gran número de herramientas que le facilitan la creación de botones, formularios y ventanas de diálogo con el uso del mouse, esto se logra con la herramienta Qt Designer. Es totalmente gratuito para aplicaciones de código abierto. Es multiplataforma por lo que la aplicación resultante puede ser compilada y utilizada en cualquier sistema operativo sin necesidad de cambiar el código.

1.3.4 Herramienta CASE

La Ingeniería de *Software* Asistida por Computación (CASE) se caracteriza por la aplicación de métodos y técnicas, a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación [28].

Visual Paradigm

Visual Paradigm [29] es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Algunas de las principales características de esta herramienta:

- Soporte de UML
- Ingeniería inversa - Código a modelo, código a diagrama
- Generación de código - Modelo a código, diagrama a código
- Editor de detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.

- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (SGBD) existentes a diagramas de Entidad-Relación.

Rational Rose Enterprise

Rational Rose[30] es una herramienta de diseño orientada a objetos, que da soporte al modelado visual, es decir, que permite representar gráficamente el sistema, permitiendo hacer énfasis en los detalles más importantes, centrándose en los casos de uso y enfocándose hacia un *software* de mayor calidad, empleando un lenguaje estándar común que facilita la comunicación. Proporciona mecanismos para realizar la Ingeniería Inversa, es decir, que a partir del código se pueda obtener información sobre su diseño; adicionalmente permite generar código en diferentes lenguajes a partir de un diseño en UML.

Algunas de las principales características de esta herramienta:

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "*Design Patterns: Elements of Reusable Object-Oriented Software*".
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Capacidad de análisis de calidad de código.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.

Enterprise Architect

Enterprise Architect (EA) [31] es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de *software* desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multiusuario, basada en *Windows*, diseñada para ayudar a construir *software* robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. El manual de usuario está disponible en línea.

Algunas de las principales características de esta herramienta:

- Diseño y construcción de UML
- Extensiones personalizadas para modelado de procesos y más
- Documentación de alta calidad compatible con MS *Word*
- Modelado de Datos, Ingeniería directa de Base de Datos a DDL e ingeniería inversa de Base de Datos desde ODBC
- Multiusuario (solamente para ediciones Profesional y Corporativa)
- Ingeniería de Código Directa e Inversa (ediciones Corporativa y Profesional solamente) - Soporte para ActionScript 2.0, Java, C#, C++, VB.Net, Delphi, Visual Basic, Python y PHP
- Facilidad de Importación/Exportación XMI

Como quiera que todas las herramientas CASE descritas presentan licencias de *software* costosas, y ya la Universidad de las ciencias informáticas, donde se realiza la presente investigación posee la licencia del Visual Paradigm, esto constituye un elemento fundamental a la hora de seleccionar cuál de ellas utilizar. Además es importante destacar que Visual Paradigm es una herramienta multiplataforma y robusta. Permite el control de versiones y genera la documentación en varios formatos lo que favorece la rapidez del trabajo de los desarrolladores. Existe una gran cantidad de información sobre la herramienta, tutoriales y videos que ayudan a su aprendizaje fácilmente.

1.3.5 Sistema gestor de base de datos

De forma sencilla un Sistema de Gestión de Bases de Datos (SGBD) se puede definir como un conjunto de programas dedicados para acceder a una colección de datos muy bien interrelacionadas. “Los SGBD, son aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma”. [32]

MySQL

MySQL es un SGBD [33] relacional, licenciado bajo la GPL (acrónimo en inglés de GNU) de la GNU (*General Public License*). Su diseño multihilo le permite soportar una gran carga de forma muy eficiente.

Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo
- Soporta gran cantidad de tipos de datos para las columnas
- Dispone de API's 6 en gran cantidad de lenguajes (C, C++, Java, PHP, etc.)
- Gran portabilidad entre sistemas

- Soporta hasta 32 índices por tabla
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos

PostgreSQL

PostgreSQL es un SGBD [34] objeto-relacional ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un SGBD puramente orientado a objetos.

Las principales características de este gestor de bases de datos son las siguientes:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios
- Permite la declaración de funciones propias, así como la definición de disparadores
- Soporta el uso de índices, reglas y vistas
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos
- Cuenta con un amplio conjunto de enlaces con lenguajes de programación como por ejemplo C, C++, Java, Python, PHP y otros
- El progreso continuo del gestor de datos de código abierto PostgreSQL brinda a los consumidores la opción de instalar una base de datos no privativa

Oracle

Oracle es un SGBD [35] con características objeto-relacionales, que pertenece al modelo evolutivo de SGBD.

Las principales características de este gestor de bases de datos son las siguientes:

- Entorno cliente/servidor
- Gestión de grandes bases de datos
- Usuarios concurrentes
- Alto rendimiento en transacciones

- Sistemas de alta disponibilidad
- Disponibilidad controlada de los datos de las aplicaciones
- Gestión de la seguridad

SQLite

SQLite [36] es un SGBD de dominio público, totalmente libre. Una de sus diferencia con relación a los motores de bases de datos convencionales es su arquitectura cliente/servidor, pues SQLite es independiente, simplemente se realizan llamadas a subrutinas o funciones de las propias librerías de SQLite, lo cual reduce ampliamente la latencia en cuanto al acceso a las bases de datos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en un solo ordenador. Es un gestor de bases de datos totalmente libre y multiplataforma.

Las principales características de este gestor de bases de datos son las siguientes:

- *Tamaño:* SQLite tiene una pequeña memoria y una única biblioteca es necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- *Rendimiento de base de datos:* SQLite realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- *Portabilidad*: se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- *Interfaces:* cuenta con diferentes interfaces del API, las cuales permiten trabajar con C++, PHP, Perl, Python, Ruby, Tcl, groovy, etc.
- *Costo:* SQLite es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.
- *No posee configuración:* de la forma en que fue creado y diseñado SQLite, no necesita ser instalado. No prender, reiniciar o apagar un servidor, e incluso configurarlo. Esta cualidad permite que no haya un administrador de base de datos para crear las tablas, vistas, asignar permisos. O bien la adopción de medidas de recuperación de servidor por cada caída del sistema.

SQLite elimina las incomodidades que generan al usuario la configuración e instalación de sistemas adicionales como Oracle, MySQL o PostgreSQL. En caso de ser necesario para los especialistas cambiar de ordenador, solo deben copiar un solo archivo creado por la base de datos de SQLite. Con su

utilización el tiempo de respuesta de una petición de la aplicación a la base de datos es menor debido a su característica de un rendimiento eficiente de la base de datos.

1.3.6 Estilos arquitectónicos

Buschmann[1] define estilo arquitectónico como una familia de sistemas de *software* en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. [37]

Entre los estilos arquitectónicos más difundidos y usados se encuentran:

- Estilos de flujo de datos
    - Tubería y filtros: [38] el sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes. Los componentes son programas independientes; el supuesto es que cada paso se ejecuta hasta completarse antes que se inicie el paso siguiente.
- Estilos centrados en datos
    - Arquitecturas de pizarra o repositorio: [39] utiliza una estructura de datos para representar el estado actual y una colección de objetos independientes que operan sobre él como componentes principales. Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de habla, etc.), o en sistemas que involucran acceso compartido a datos con agentes débilmente acoplados. [40]
- Estilos de llamada y retorno
    - Model-View-Controller (MVC): [41] separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes, el modelo que se encarga de administrar el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y responde a instrucciones de cambiar el estado (habitualmente desde el controlador); la vista que maneja la

[1] **Frank Buschmann** Ingeniero principal de sistemas e ingeniería de software y arquitectura de Siemens Corporate Technology en Munich.

visualización de la información y el controlador que interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

- Arquitecturas en capas: definida por Garlan y Shaw [42] como una organización jerárquica, tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las interacciones entre las capas ocurren generalmente por invocación de métodos, por definición los niveles más bajos no deben poder utilizar funcionalidad ofrecida por las capas superiores. Este estilo facilita la modularidad del sistema, la localización de errores y mejora considerablemente el soporte del sistema.

- Arquitecturas orientadas a objetos: [12] los componentes del estilo se basan en principios orientados a objetos como encapsulamiento, herencia y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación; las interfaces están separadas de las implementaciones; en cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases [42]. Entre las cualidades de este estilo, la más básica concierne a que se puede modificar la implementación de un objeto sin afectar a sus clientes. Asimismo es posible descomponer problemas en colecciones de agentes en interacción. Además, por supuesto (y esa es la idea clave), un objeto es ante todo una entidad reutilizable en el entorno de desarrollo.

- Arquitecturas basadas en componentes: [43] los sistemas de *software* basados en componentes se basan en principios definidos por una ingeniería de *software* específica. Los componentes son las unidades de modelado, diseño e implementación, las interfaces están separadas de las implementaciones, y las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico, los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

- Estilos de Código Móvil

- Arquitectura de máquinas virtuales: [44] la arquitectura de máquinas virtuales se ha llamado también intérpretes basados en tablas o sistemas basados en reglas, este estilo incluye un seudo-programa a interpretar y una máquina de interpretación. Ambas variedades abarcan, sin duda, un extenso espectro que va desde los llamados lenguajes de alto nivel hasta los paradigmas declarativos no secuenciales de programación.

- Estilos heterogéneos

  - Sistemas de control de procesos: desde el punto de vista arquitectónico, mientras casi todos los demás estilos se pueden definir en función de componentes y conectores, los sistemas de control de procesos se caracterizan no sólo por los tipos de componentes, sino por las relaciones que mantienen entre ellos. Es uno de los pocos tratamientos arquitectónicos para modelos cibernéticos. La ventaja señalada para este estilo radica en su elasticidad ante perturbaciones externas. [45]

  - Arquitecturas basadas en atributos: la arquitectura basada en atributos o ABAS fue propuesta por Klein y Klazman. La intención de estos autores es asociar a la definición del estilo arquitectónico un framework de razonamiento (ya sea cuantitativo o cualitativo) basado en modelos de atributos específicos. [45]

- Estilos Peer-to-Peer

  - Arquitecturas basadas en eventos: las arquitecturas basadas en eventos se han llamado también de invocación implícita. Las arquitecturas basadas en eventos se vinculan históricamente con sistemas basados publicación-suscripción. Los conectores de estos sistemas incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa un componente puede anunciar mediante difusión uno o más eventos. [46]

  - Arquitecturas orientadas a servicios (SOA): es lo suficientemente flexible, elegante y ágil garantizando las soluciones que las empresas han anhelado siempre. Es una arquitectura de *software* construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces; es una relación entre servicios y consumidores

de servicios, ambos lo suficientemente amplios como para representar una función de negocio completa. [46]

- Arquitecturas basadas en recursos: define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es nada menos que la World Wide Web, donde los URIs identifican los recursos y HTTP es el protocolo de acceso. El argumento central es que HTTP mismo, con su conjunto mínimo de métodos y su semántica simplísima, es suficientemente general para modelar cualquier dominio de aplicación. De esta manera, el modelado tradicional orientado a objetos deviene innecesario y es reemplazado por el modelado de entidades tales como familias jerárquicas de recursos abstractos con una interfaz común y una semántica definida por el propio HTTP. [41]

Para el desarrollo de la presente solución se propone el estilo de llamada y retorno ya que es un estilo que se centra en la interacción de un usuario con el sistema. La arquitectura se puede dividir en dos partes fundamentales, la primera representa la interfaz del usuario con la que este realiza las llamadas al sistema, la segunda contiene la lógica del procesamiento del negocio que se realiza tras la llamada del usuario al sistema. Esta familia de estilos favorece la modificabilidad, escalabilidad y la reusabilidad.

1.4 Conclusiones parciales

Al abordar los elementos relacionados a la fundamentación teórica se logró aumentar el horizonte investigativo, dándole así soporte a la investigación científica con el objetivo de solucionar la problemática presente. El análisis de las soluciones existentes que gestionan información relacionada con el estudio neuroinmunológico del LCR, demostraron que no cumplimentan el objetivo trazado en la presente investigación. Además, al realizar un análisis detallado de las tendencias tecnológicas actuales a nivel mundial, se seleccionaron las herramientas y tecnologías adecuadas para el desarrollo de la aplicación basándose en las características de las mismas.

Capítulo 2 Características del sistema para la gestión de la información en el estudio neuroinmunológico de proteínas del líquido cefalorraquídeo a desarrollar

En este capítulo se aborda todo lo referente a las características del sistema, la situación actual y el problema existente en el LABCEL. Se identifica el negocio en general y se detallan los requisitos funcionales y no funcionales, los que permitirán formar los cimientos de la aplicación de escritorio a construir.

2.1 Modelo de procesos del negocio

El análisis de los modelos de procesos de negocios está enfocado principalmente en qué elementos de la realidad a ser modelada pueden ser representados. El ámbito del modelado considera aspectos inherentes a los procesos de negocios.

En la investigación se utiliza específicamente el proceso de negocio que es un conjunto estructurado de actividades, diseñado para producir una salida determinada o lograr un objetivo. Los procesos describen cómo es realizado el trabajo y se caracterizan por ser observables, medibles, mejorables y repetitivos. Estructuralmente, un proceso de negocio está constituido por un conjunto de actividades como elemento básico. [47]

El modelo de negocio tiene como objetivo describir los procesos existentes u observados en la organización en la cual se va a establecer un sistema, además de comprender sus problemas actuales e identificar las mejoras potenciales.

2.1.1 Descripción de los procesos del negocio del LABCEL

El LABCEL es un laboratorio especializado para análisis avanzado de líquido cefalorraquídeo. Este centro tiene como objetivo prestar servicios científico – técnicos de alto valor agregado para las instituciones hospitalarias del país. Los estudios realizados en el laboratorio se hacen gracias a la información obtenida de las muestras de LCR y suero de un paciente. Dichas muestras son almacenadas en el banco de muestras perteneciente al laboratorio. Las personas que laboran en este centro son especialistas en neuroinmunología, constituyendo su principal tarea el estudio de las muestras. El estudio que se realiza tiene el propósito de obtener información para determinar la afección que pueda presentar el paciente y arribar a conclusiones sobre cómo estas enfermedades se comportan en una determinada población.

Los especialistas se encargan de la gestión de las muestras y de registrar los resultados que se obtienen del análisis de las mismas. Son los facultados de mantener la confidencialidad de la información gestionada y del cuidado y almacenamiento de las condiciones de las muestras para posteriores estudios.

El **proceso de estudio neuroinmunológico de proteínas del LCR** inicia cuando el especialista del LABCEL recibe por parte de una institución hospitalaria una solicitud de estudio, emitida por un médico de dicha institución. La acción se realiza mediante un trabajador mensajero o un familiar del paciente quien entrega la solicitud, la muestra del LCR de una punción (ya sea lumbar, cisternal o ventricular) y de suero del paciente.

El especialista procede a registrar la información de la solicitud de estudio donde se recogen los datos generales del paciente, de la institución hospitalaria, del técnico que realizó la extracción de las muestras y del médico que indicó la realización de las mismas. Las muestras recepcionadas pasan a ser analizadas por un especialista en neuroinmunología con la ayuda de equipos médicos; obteniéndose un informe de resultados bioquímicos con los datos del análisis realizado. A continuación, el especialista registra los resultados del análisis de las muestras.

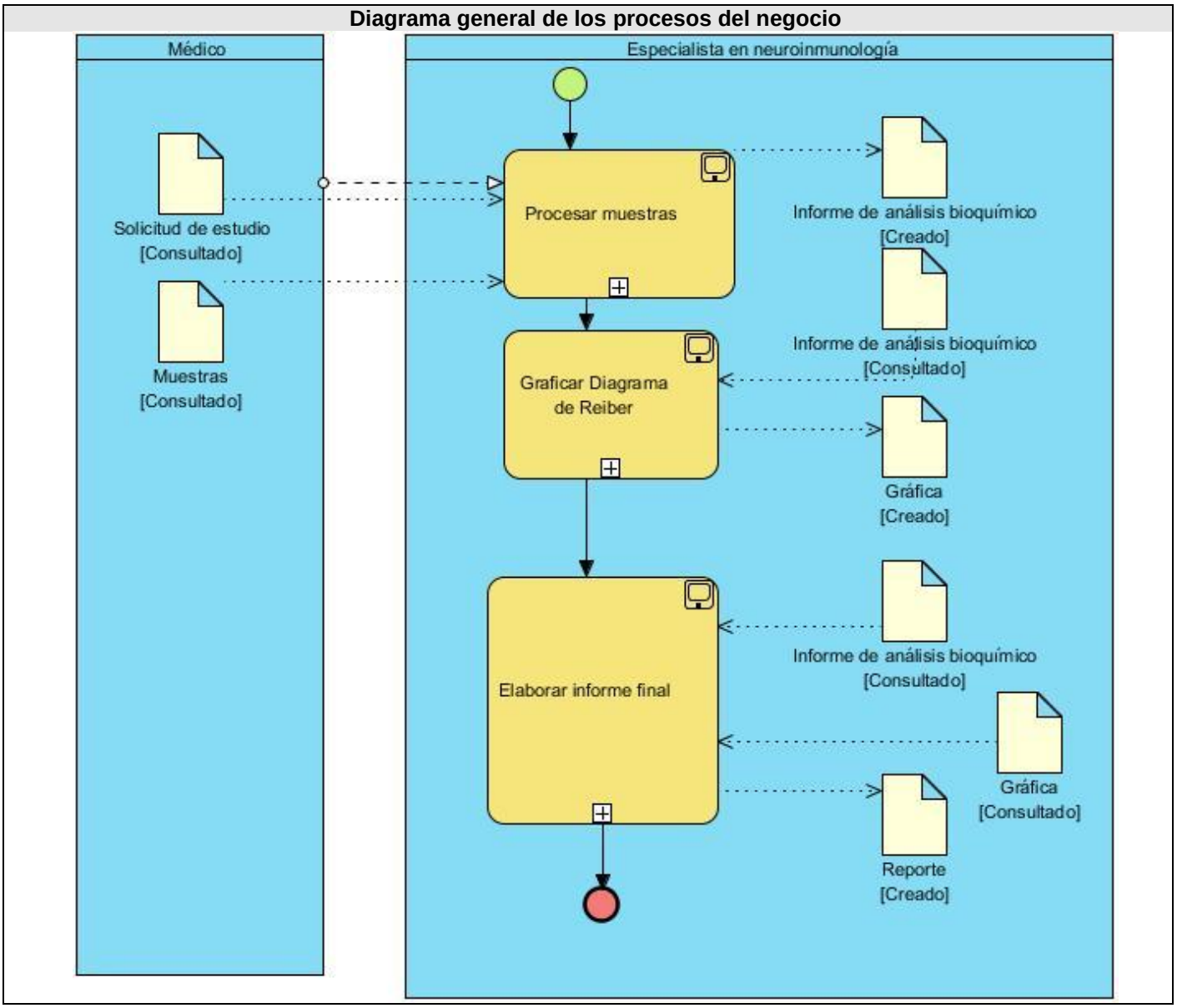
Los datos obtenidos pasan a ser evaluados con fórmulas de cálculo establecidas, que determinan la síntesis intratecal de las inmunoglobulinas específicas que se desean analizar. El especialista procede a graficar los diagramas de Reiber correspondientes a cada clase de inmunoglobulina u otra proteína haciendo uso de los datos derivados mediante el cálculo. Una vez obtenidos y analizados los gráficos se determina un diagnóstico para el paciente. Los resultados son guardados para estudios futuros y se utilizan para elaborar un informe final, que será entregado al mensajero de la institución remitente, acción que finaliza el proceso.

2.1.2 Diagrama de procesos del negocio

Los diagramas de procesos del negocio ayudan a describir en detalle qué es lo se desarrolla dentro del negocio, y para ello se examinaron los roles específicos que juegan las personas (Trabajadores del negocio) y las actividades que realizan. [48]

A continuación se muestra el diagrama de procesos actuales del negocio:

Tabla 1 Diagrama general de los procesos del negocio



Dentro de los procesos del negocio se destacaron los procesos **Procesar muestras** y **Elaborar informe** final, al desarrollarse en ellos las actividades en que se enfoca la presente investigación.

Seguidamente se muestran los diagramas de proceso correspondiente a los procesos antes mencionados:

Tabla 2 Diagrama del proceso: P_1 Procesar muestras

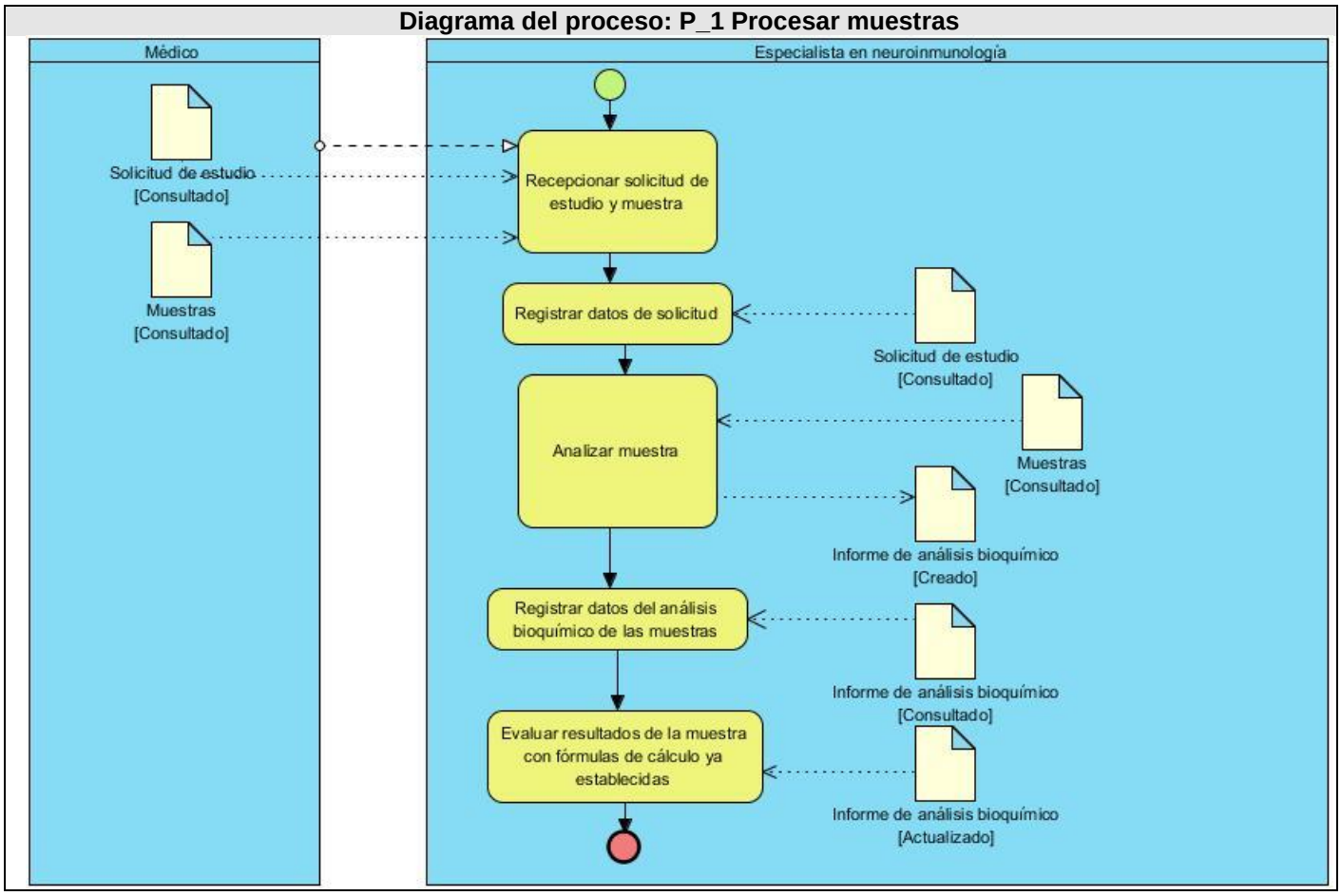
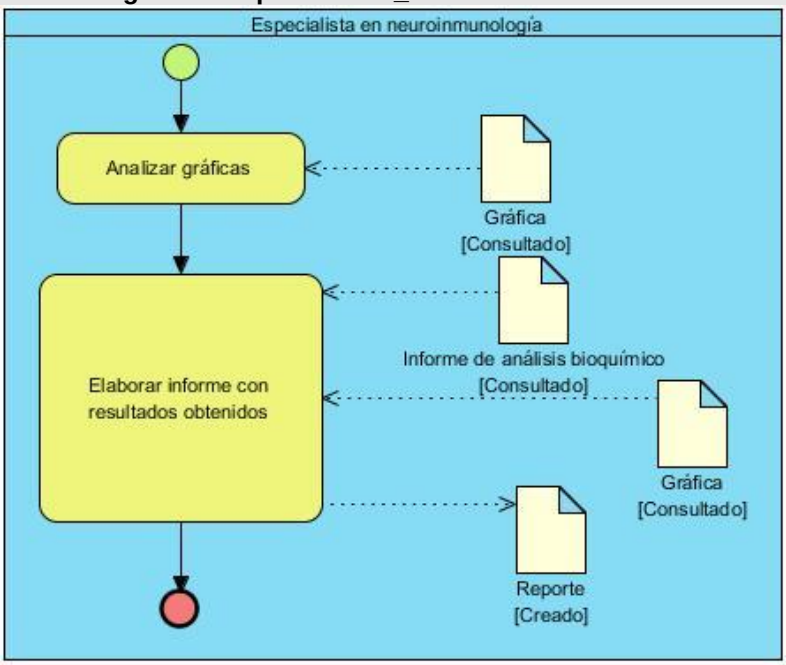


Tabla 3 Diagrama del proceso: P_3 Elaborar informe final

Diagrama del proceso: P_3 Elaborar informe final



2.2 Propuesta del sistema a desarrollar

Para resolver el problema antes planteado se propone como solución, una aplicación de escritorio que gestione los datos que se generan en el LABCEL al realizar un estudio neuroinmunológico a las muestras de LCR y suero de la sangre de un paciente determinado. Las características básicas del sistema estarán centradas en:

- Permitir gestionar los datos de los pacientes atendidos en la institución.
- Permitir gestionar los datos de las instituciones que remiten a los pacientes a realizarse un estudio.
- Permitir gestionar los datos de las muestras del estudio realizado.
- Generar informe final con los resultados del estudio neuroinmunológico realizado.

2.3 Especificación de los requisitos del software

La especificación de los requerimientos tiene como propósito principal llegar a un mejor entendimiento entre el cliente y los desarrolladores del sistema, para así poder determinar lo que debe y no hacer el

sistema. Además, una buena elección de los requerimientos hace posible mantener el orden y un seguimiento detallado de la aplicación.

Seguidamente se especifican y describen los requerimientos del sistema con el propósito de un mejor entendimiento interno del mismo.

2.3.1 Requisitos funcionales

Los requerimientos funcionales son capacidades o condiciones con las que debe cumplir el producto a elaborar, no alteran la funcionalidad del *software*, se mantienen invariables sin importar con qué cualidades o propiedades se relacionen. [49]

Seguidamente se especifican y describen los requerimientos del sistema con el propósito de un mejor entendimiento interno del mismo.

Descripción de los Requisitos Funcionales

RF1. Registrar institución: permite introducir los datos del instituto/laboratorio donde se realiza el estudio neuroinmunólogico de las muestras de LCR y suero remitidas por una institución hospitalaria.

RF2. Adicionar paciente: permite introducir los datos de un paciente.

RF3. Buscar paciente: permite buscar a un paciente determinado.

RF4. Listar pacientes: permite listar todos los pacientes a los que ya se le ha hecho un estudio neuroinmunólogico.

RF5. Registrar muestra: permite introducir los datos de las muestras de LCR y suero de un paciente.

RF6. Registrar análisis especiales: permite introducir los datos de un análisis en especial realizado a las muestras de LCR y suero de un paciente.

RF7. Registrar conteo diferencial de células: permite introducir los datos de las células presentes en las muestras de LCR y suero de un paciente.

RF8. Exportar informe de conteo diferencial de células: permite exportar en formato PDF un informe con los datos de las células presentes en las muestras de LCR y suero de un paciente.

RF9. Imprimir informe conteo diferencial de células: permite imprimir un informe con los datos de las células presentes en las muestras de LCR y suero de un paciente.

RF10. Registrar anticuerpos: permite introducir los datos de los anticuerpos presentes en las muestras de LCR y suero de un paciente.

RF11. Registrar institución remitente: permite introducir los datos de la institución remitente/institución hospitalaria que remitió las muestras de LCR y suero de un paciente para su estudio neuroinmunólogico.

RF12. Listar institución remitente: permite listar todas las instituciones remitentes.

RF13. Exportar informe final: permite exportar en formato PDF un informe con los resultados del estudio neuroinmunólogico realizado a las muestras de LCR y suero de un paciente. En él se recoge información del instituto, del paciente, valores clínicos asociados a las muestras y los gráficos en dependencia de los valores de las proteínas que fueron registrados.

RF14. Imprimir informe final: permite imprimir un informe con los resultados del estudio neuroinmunólogico realizado a las muestras de LCR y suero de un paciente.

2.3.2 Requisitos no funcionales

Los requerimientos no funcionales son capacidades o cualidades que el producto debe tener, o sea, características que hagan al producto atractivo, rápido, usable o confiable. Están estrechamente vinculados a los requisitos funcionales, puesto que una vez que está definido lo que el sistema debe hacer, es necesario especificar como ha de hacerlo. Pueden llegar a marcar la diferencia entre un producto bien aceptado por los clientes y usuarios o uno de poca o ninguna calidad y aceptación [50].

1. **Requerimientos de *software*:**
   - Las computadoras en las que se utilizará el *software*, deben tener instalado: en la plataforma Windows en su versión XP o superior o la plataforma GNU-Linux en su versión *Ubuntu12.4 o Debian* 6.0 estable.
   - Se debe disponer de un visor para documentos en formato PDF.
2. **Requerimientos de hardware**
   - Las computadoras que utilizarán el *software* deben tener capacidad de hardware que soporte 128 Mb de memoria RAM, un microprocesador de 2.0 Hz.
   - La estación de trabajo debe disponer de una impresora para la impresión de los reportes emitidos por el sistema.
3. **Requerimientos de apariencia o interfaz**
   - Las interfaces de usuario contendrán los datos claros y bien estructurados para facilitar la interpretación correcta de la información.
   - El usuario debe tener acceso por varias vías a una determinada funcionalidad.

- La aplicación debe utilizar como idioma el español.
- Los componentes visuales deben ser auto descriptibles.

4. Requerimientos de usabilidad

- El sistema podrá ser utilizado por cualquier persona que posea pocos conocimientos informáticos debido a la homogeneidad con el sistema operativo.

5. Requerimientos de confiabilidad

- No se podrá eliminar ni modificar la información en el sistema.
- Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo.

6. Requerimientos de eficiencia

- El sistema adoptará buenas prácticas de programación para incrementar el rendimiento en las operaciones.
- La búsqueda de la información se realizará de manera más rápida con el uso del modelo *QSortFilterProxyModel*, disminuyendo el tiempo de respuesta de la aplicación.

7. Requerimiento de seguridad

- El sistema evitará la inyección SQL validando la entrada de los datos con la utilización de expresiones regulares.

8. Requerimientos de soporte

- El sistema debe tener una arquitectura flexible, capaz de permitir cambios sin que afecten la estructura de la solución.
- Se permitirá realizar modificaciones posteriores para adaptar mejoras al sistema o en caso que cambien las necesidades de los clientes.

9. Requerimientos de portabilidad

- El sistema permitirá ser lo más portable posible y sin dependencia de aplicaciones ajenas necesarias para su funcionamiento.

10. Requerimientos de implementación

- Se usará como entorno de desarrollo integrado (IDE) Qt Creator, lenguaje de programación C++ y como gestor de bases de datos SQLite.

11. Requerimientos políticos-culturales

- El idioma que se empleará en el sistema será el español.
- Contará con logotipos e imágenes que se encuentren en correspondencia con el carácter científico y profesional del tema.

2.4 Actores del sistema

Un actor del sistema representa el rol que juega una o varias personas, un equipo o un sistema automatizado que interactúa con el sistema y que es externo a él. [51]

En la siguiente tabla se muestra la descripción del actor del sistema.

Tabla 4 Actor del sistema

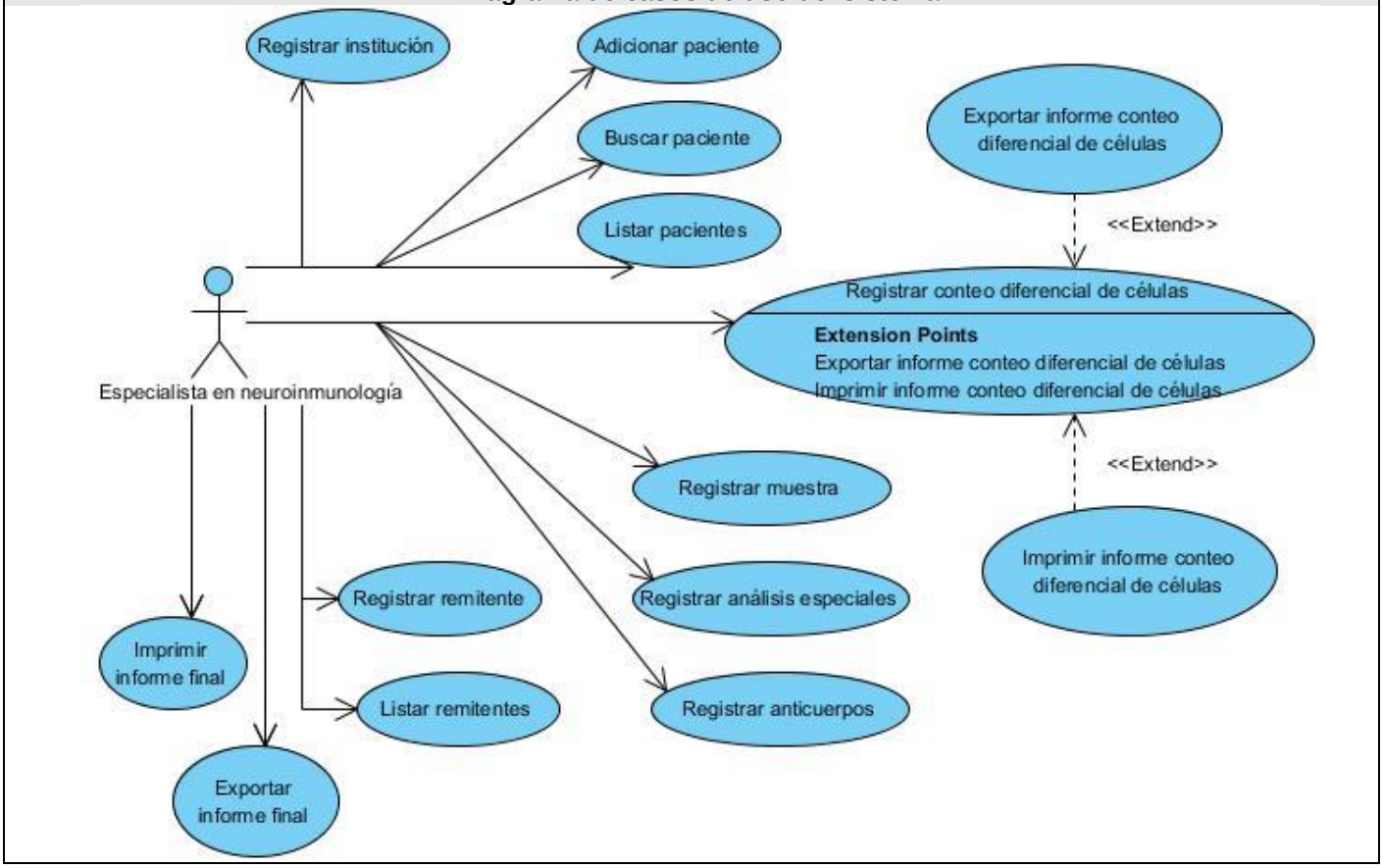
Actor	Función
Especialista en neuroinmunología	Persona encargada del estudio de las muestras de líquido obtenidas de la punción lumbar. Facultado para emitir diagnóstico. Puede gestionar toda la información relacionada con las muestras de LCR registradas en el sistema y los resultados del estudio de las mismas. Puede generar reportes e imprimirlos.

2.5 Diagrama de caso de uso del sistema

Los diagramas de casos de uso muestran las relaciones entre los casos de uso de un sistema y sus actores. Tiene como objeto modelar el comportamiento de un sistema, un subsistema o una clase. [51]

Tabla 5 Diagrama de casos de uso del sistema

Diagrama de casos de uso del sistema



2.6 Casos de uso del sistema

Para conocer en detalle cada caso de uso se muestra a continuación la descripción general asociada a estos:

Tabla 6 Descripción del CU "Adicionar paciente"

Caso de Uso	Adicionar paciente
Resumen	El caso de uso inicia cuando un usuario (Usuario o Especialista) decide insertar un paciente.
Actores	Usuario(Especialista)
Precondiciones	
REFERENCIAS	
Referencias	RF 2
FLUJO NORMAL DE LOS EVENTOS	

Acción del Actor	Respuesta del sistema
El caso de uso inicia cuando el actor accede a la opción “Paciente” y decide realizar la acción de: Adicionar un paciente.	Muestra un formulario con los campos necesarios para el registro de los datos de un paciente: ❖ No. HC ❖ Nombre ❖ Primer Apellido ❖ Segundo Apellido ❖ Sexo ❖ Fecha de nacimiento ❖ Provincia ❖ Municipio ❖ Dirección
Inserta los datos especificados.	Verifica que los datos insertados están completos y son correctos. • Detecta que existen campos vacíos que son obligatorios de llenar. (Ver Flujo Alterno 1). • Detecta errores en los datos registrados. (Ver Flujo Alterno 1). • Detecta que ya existe ese paciente. (Ver Flujo Alterno 2). Se activa el botón para adicionar el paciente al sistema una vez introducido todos los datos. (Botón “Adicionar”)
Decide adicionar un paciente en el sistema. (Botón “Adicionar”)	Se insertan los datos especificados. El sistema visualiza el listado actualizado de los pacientes.
Flujo Alterno 1	
	Se señalan con un asterisco color rojo los campos que tienen datos incorrectos o si se dejó algún campo necesario vacío para indicar el error. Permite volver a registrar los datos.
Flujo Alterno 2	
	Muestra el mensaje “El paciente ya existe en el sistema.”
Decide adicionar una muestra en el sistema. (Botón “Examen”)	Si el usuario selecciona el botón “Examen”. Ver descripción CU ¨Registrar muestra¨

Tabla 7 Descripción del CU "Registrar muestra"

Caso de Uso	Registrar muestra
Resumen	El caso de uso se inicia cuando un usuario desea insertar una determinada muestra.
Actores	Usuario(Especialista)
Precondiciones	Debe estar insertado el paciente para registrarle una nueva muestra.
REFERENCIAS	
Referencias	RF 5
FLUJO NORMAL DE LOS EVENTOS	
Acción del Actor	Respuesta del sistema
El caso de uso inicia cuando el actor accede a la opción “LCR” y decide realizar la acción de: Registrar una muestra.	Muestra un formulario con los campos necesarios para el registro de los datos de una muestra de LCR: • Volumen • Médico • Fecha de punción • Recibo de la muestra • Institución remitente • Impresión diagnóstica
Inserta los datos especificados.	Verifica que los datos insertados están completos y son correctos. • Detecta que existen campos vacíos que son obligatorios de llenar. (Ver Flujo Alterno 1). • Detecta errores en los datos registrados. (Ver Flujo Alterno 1). • Detecta que no existe el Instituto remitente. (Ver Flujo Alterno 2).
Decide adicionar una muestra en el sistema. (Botón “Guardar”)	Se insertan los datos especificados de la muestra. El sistema visualiza el listado actualizado de las muestras.
Flujo Alterno 1	

	Se señalan con un asterisco color rojo los campos que tienen datos incorrectos o si se dejó algún campo necesario vacío para indicar el error. Permite volver a registrar los datos.
Flujo Alterno 2	
Decide adicionar una institución remitente en el sistema. (Botón ).	Si el usuario selecciona el botón ). Ver descripción CU ¨Registrar Institución remitente¨.

2.7 Conclusiones parciales

Al profundizar en el funcionamiento del negocio del laboratorio LABCEL, se identificaron dificultades tales como: el sistema que utilizan actualmente no permite cuantificar la nuevas proteínas IgE, MBL, C3c y C4, la gestión de la información se hace engorrosa; lo que sienta las bases para la etapa de análisis de la presente investigación. Con el uso de la entrevista se identificaron los requisitos el sistema lo que permite el fácil mantenimiento y gestión de los mismos.

Capítulo 3 Diseño e Implementación del sistema para la gestión de la información en el estudio neuroinmunológico de proteínas del líquido cefalorraquídeo

En el presente capítulo se profundiza en los casos de usos detallándolos de manera que permitan reflejar una vista interna del sistema, descrito con el lenguaje de los desarrolladores. Trazando como objetivos esenciales: transformar los requerimientos definidos a una propuesta de diseño que constituirá una guía para la implementación del sistema. Se muestra el diagrama de componentes; así como, la disposición física de los distintos nodos por medio del diagrama de despliegue. Además, se define el estándar de codificación a emplear en la solución a desarrollar.

3.1 Descripción de la arquitectura propuesta

Los patrones de arquitectura definen la estructura de un sistema *software*, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño.

En la realización de este sistema se define a utilizar la arquitectura modelo/vista una especialización del Modelo Vista Controlador (MVC).

Si en la arquitectura MVC se combinan las vistas y los objetos del controlador, el resultado es la arquitectura de modelo/vista. Esto todavía separa la forma en que los datos se almacenan de la forma en que se presenta para el usuario, pero proporciona un marco sencillo basado en los mismos principios. Esta separación hace posible la visualización de los mismos datos en varias vistas diferentes, y para poner en práctica nuevas formas de vistas, sin necesidad de cambiar las estructuras de datos subyacentes.

La arquitectura de modelo / vista

En la arquitectura modelo / vista [52] el modelo se comunica con una fuente de datos, que proporciona una interfaz para los demás componentes en la arquitectura. La naturaleza de la comunicación depende del tipo de fuente de datos, y la forma en que el modelo se implementa.

La vista obtiene índices del modelo, que son referencias a los elementos de los datos. Mediante el suministro de los índices de modelo, la vista puede recuperar los elementos de los datos de la fuente de datos.

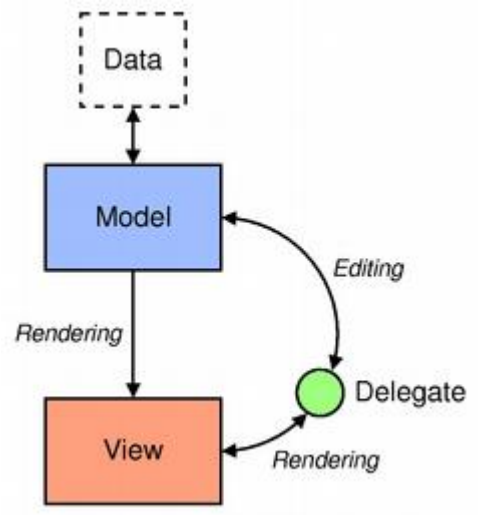


Figura 2 Arquitectura Modelo-Vista [52]

En las vistas estándar, un delegado crea los elementos de datos de la misma. Cuando un elemento es editado, el delegado se comunica con el modelo directamente utilizando el índice del modelo.

Modelos, vistas y delegados se comunican entre sí mediante señales y ranuras*:*

- Las señales del modelo le informa a la vista de los cambios de los datos en poder de la fuente de datos.
- Las señales de la vista proporcionan información acerca de la interacción del usuario con los elementos que se muestran.
- Las señales del delegado se utilizan durante la edición para decirle al modelo y la vista acerca del estado del editor.

Modelos

Los modelos para acceder a los diferentes tipos de fuentes de datos, derivan de *QAbstramItemModel*, y entre los que proporciona están: *QStringListModel*, *QStandardItemModel*, *QFileSystemModel* y *QSqlRelationalTableModel*.

- *QStringListModel*, es usado para guardar una lista de QStrings.
- *QStandardItemModel*, se usa para el manejo de estructuras más complicadas en forma de árbol, que pueden guardar cualquier tipo de datos.

- *QFileSystemModel*, provee de la información que obtiene del sistema de ficheros, como QDirModel, pero más avanzado.
- *QSqlRelationalTableModel*, se usa para tratar datos provenientes de una base de datos relacional. De ella deriva *QSqlTableModel* que se usa para guardar los datos de una tabla concreta de una base de datos. De ella deriva *QSqlQueryModel* que guarda los datos de la respuesta a una consulta a una base de datos.

Vistas

Las vistas son todas derivadas de la clase *QAbstractItemView*. Las vistas que de ella se derivan son tres tipos principalmente: *QListView*, *QTableView* y *QTreeView*.

- *QListView*, proporciona una vista unidimensional, donde hay sólo una columna (*column*) y muchas filas (*row*). Dispone de una serie de métodos para cambiar la presentación de los ítems de la misma.
- *QTableView*, es la más usada, sobre todo en la presentación de tablas de bases de datos. Tiene forma bidimensional con varias columnas y varias filas, además posee cabecera tanto para las columnas como para las filas, para poner sus títulos en ellas.
- *QTreeView*, es usada sobre todo en acceso a directorios y dispone la información en forma de árbol. Además también dispone de cabecera horizontal (*header*), para poner títulos a las propiedades de los elementos.

Delegados

Los delegados son todos los derivados de la clase *QAbstractItemDelegate*. Los delegados que de ella se derivan son: *QStyledItemDelegate* y *QItemDelegate*.

- *QStyledItemDelegate* implementa el delegado por defecto.
- *QItemDelegate* proporciona la visualización y funciones de edición de elementos de los datos de un modelo

3.2 Modelo de Análisis

El análisis consiste en obtener una visión un poco más detallada del sistema, de modo que se centra en los requisitos funcionales, para refinarlos y estructurarlos. Con el objetivo de conseguir una comprensión

más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura. [53]

En la construcción del Modelo de Análisis se identifican las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de Clases del Análisis. Los Diagramas de Clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones. [54]

Tabla 8 Diagrama de clases de análisis. CU Adicionar paciente

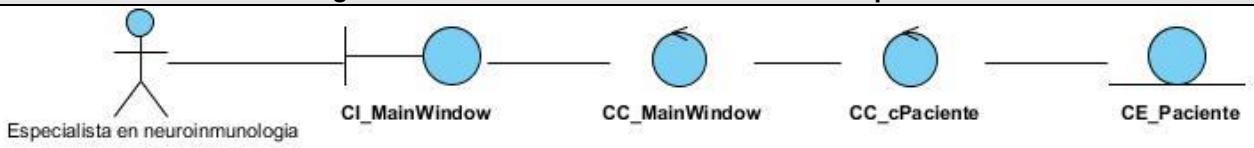
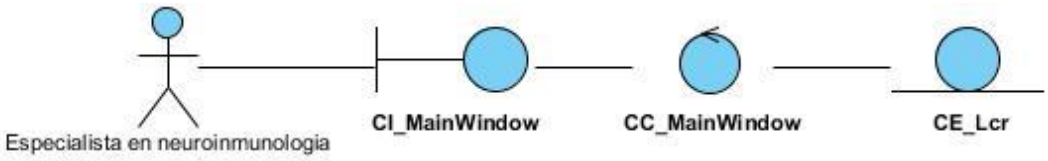
Diagrama de clases de análisis. CU Adicionar paciente


Tabla 9 Diagrama de clases de análisis. CU Registrar muestra

Diagrama de clases de análisis. CU Registrar muestra


3.3 Modelo del Diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación. [55]

3.3.1 Patrones de diseño

Actualmente están disponibles un elevado número de patrones que abarcan varios dominios de aplicaciones y lenguajes. La noción de un patrón, como un concepto reutilizable, ha sido desarrollada en varias áreas además del diseño de *software*. El uso de patrones es una forma efectiva de reutilización. [56]

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos, indicando la manera de cómo utilizarlo. Una de las

situaciones más complicadas en orientación a objeto consiste en elegir las clases adecuadas y decidir cómo estas clases deben interactuar. Los Patrones Generales de *Software* para la Asignación de Responsabilidades (GRASP) se encargan de realizar una descripción de los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Estos no compiten con los patrones de diseño, más bien ofrecen una guía para encontrar los patrones de diseño. Algunos de estos tipos de patrones son: [57]

- Bajo acoplamiento: este patrón se ve reflejado en las clases *dbConexion.h y CustomItemDelegate.h* al aplicar en su implementación este tipo de patrón. En ellas existen pocas dependencias respecto a las demás clases. Este aspecto es necesario, pues si todas las clases dependen de todas se violaría el principal principio del empleo de patrones que es el concepto de reutilización, además existiría poco código utilizado de modo independiente lo cual sería imposible de reutilizar en otro proyecto.
- Alta cohesión: este patrón se ve reflejado en las clases *printwidget.h, cpaciente.h y cinstitucionRemitente.h*. La primera realiza una labor única dentro del sistema, se encargan de crear y generar los reportes del sistema en formato PDF. La segunda y tercera gestionan todo lo relacionado con los pacientes y las instituciones remitentes respectivamente. Dichas gestiones en el sistema es desempeñada solamente por esas clases. En las mismas se ve presente la utilización del patrón Alta Cohesión al realizar solo funciones específicas y no demasiadas gestiones.
- Experto: las clases este patrón se ve reflejado en *cpaciente.h y cinstitucionRemitente.h* son las únicas dentro del sistema que gestionan información relacionada con los pacientes y las instituciones remitentes, ya que son las únicas clases que contienen los datos o atributos de dichas información por cada clase. En dichas clases se refleja la utilización del patrón experto pues contienen toda la información necesaria para realizar la labor que tiene encomendada y así la cumplen.
- Creador: este patrón se ve reflejado en la clase mainwindow.h. Ella es la encargada de crear y guiar la asignación de responsabilidades relacionadas con la creación de los objetos de tipo paciente, institución, impresión e institución remitente. En ella se ve reflejada la utilización del patrón creador, pues es la mejor candidata en el sistema para asignar la responsabilidad de crear estos tipos de objetos, al contener toda la información necesaria para realizar esta operación.

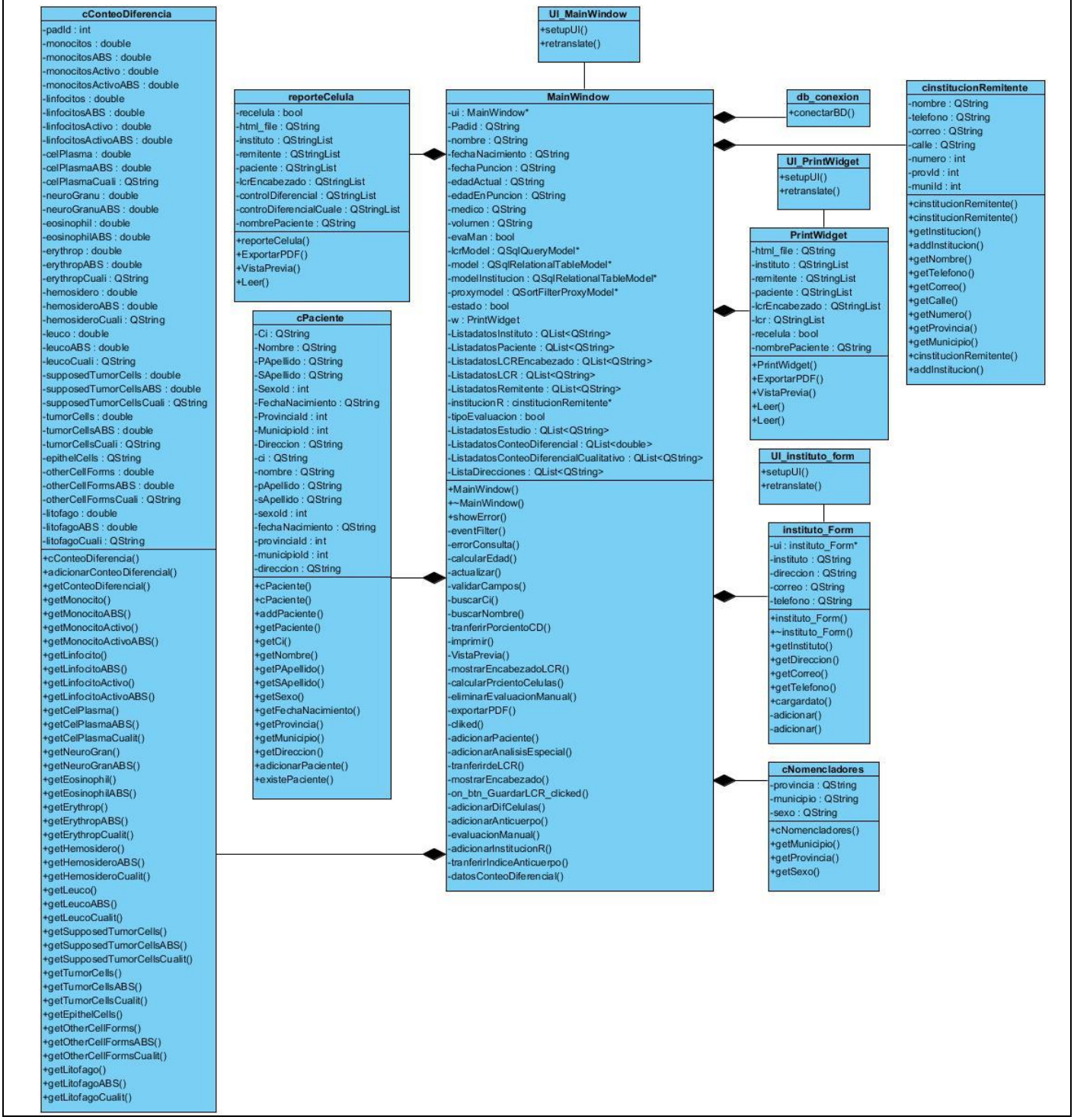
- Controlador: este patrón se ve reflejado en la clase mainwindow.h que es la clase que controla todo el flujo de eventos en el sistema.

3.3.2 Diagrama de las clases del diseño

Los diagramas de clases del diseño constituyen diagramas de estructura estática que presentan las clases del sistema y las relaciones entre ellas; expresan lo que el sistema puede hacer y cómo puede ser construido. Es muy significativo al constituir estos artefactos los que se necesitan modelar para que el desarrollador los implemente y obtener así el producto final con la calidad requerida. [54]

Tabla 10 Diagrama de clases diseño

Diagrama de clases diseño



3.3.3 Descripción de algunas de las clases de diseño

La descripción de las clases del diseño permite saber los atributos que conforman las mismas, así como las operaciones más relevantes con las que cuenta cada una de estas. A continuación se realiza la descripción de algunas de las clases de diseño asociadas al sistema.

Tabla 11 Descripción de la clase cPaciente

cPaciente	
Descripción:	La clase es la encargada de gestionar toda la información relacionada con los pacientes.
Tipo de clase:	**controladora**

Nombre	**Tipo**
ci	QString
nombre	QString
pApellido	QString
sApellido	QString
sexoId	int
fechaNacimiento	QString
direccion	QString
provId	int
muniId	int

Para cada responsabilidad:	
Nombre:	**Descripción:**
cPaciente()	Constructor de la clase.
cPaciente(QString ci,QString nombre,QString pApellido, QString sApellido, int sexoId, QString fechaNacimiento, int provinciaId, int municipioId,QString direccion)	Sobrecarga del constructor de la clase para construir un objeto de la misma con datos.
cPaciente* getPaciente(int id)	Retorna un objeto de la clase.
void adicionarPaciente(QString ci, QString nombre,QString pApellido,QString sApellido, int sexoId, QString, fechaNacimiento, int provinciaId, int municipioId, QString direccion)	Método para adicionar un paciente.
QString getCi()	Retorna el número de historia clínica del paciente.
QString getNombre(	Retorna el nombre del paciente.
QString getPApellido()	Retorna el primer apellido del paciente.
QString getSApellido()	Retorna el segundo apellido paciente.
int getSexo()	Retorna el id del sexo del paciente.
QString getFechaNacimiento()	Retorna la fecha de nacimiento del paciente.
int getProvincia()	Retorna el id de la provincia del paciente.
int getMunicipio()	Retorna el id del municipio del paciente.

QString getDireccion()	Retorna la dirección del paciente.

Tabla 12 Descripción de la clase PrintWidget

PrintWidget	
Descripción:	Clase encargada de crear los reportes, imprimirlos y exportarlos a PDF.
Tipo de clase:	**controladora**
Nombre	**Tipo**
html_file	QString
Para cada responsabilidad:	
Nombre:	**Descripción:**
PrintWidget ()	Constructor de la clase.
void Imprimir()	Método para imprimir el informe generado.
void ExportarPDF()	Método para exportar a PDF el informe generado.
void VistaPrevia()	Método para tener una vista previa del informe generado.
void Leer(QList<QString> Listainstitu, QList<QString> listaRemitente, QList<QString> Listaipaciente, QList<QString> ListaEnLCR, QList<QString> ListaLCR)	Método para crear el informe que se va a exportar a PDFo imprimir.

Tabla 13 Descripción de la clase MainWindow

MainWindow	
Descripción:	Clase controladora principal del sistema.
Tipo de clase:	**controladora**
Nombre	**Tipo**
edadActual	QString
edadEnPuncion	QString
porcientoTCD	float
lcrModel	QSqlQueryModel *
model	QSqlRelationalTableModel *
modelInstitucion	QSqlRelationalTableModel *
proxymodel	QSortFilterProxyModel *
estado	bool
wImprimir	PrintWidget
contInstitucionR	cinstitucionRemitente *
contPaciente	cPaciente *
contNomencladores	cNomencladores *
instituto	instituto_Form *
ListadatosInstituto	QList<QString>
ListadatosPaciente	QList<QString>

ListadatosLCREncabezado	QList<QString>
ListadatosLCR	QList<QString>
ListadatosRemitente	QList<QString>
Para cada responsabilidad:	
Nombre:	**Descripción:**
void showError(const QSqlError &err)	Muestra los errores con la base de datos.
void imprimir()	Llama al método imprimir de la clase PrintWidget.
void exportarPDF()	Llama al método exportar a PDF de la clase PrintWidget.
void VistaPrevia()	Llama al método vista previa de la clase PrintWidget.
void buscarCi(QString criterio)	Busca un paciente por el número de historia clínica pasada por parámetros.
void buscarNombre(QString criterio)	Busca un paciente por el nombre pasado por parámetros.
void mostrarEncabezado(QModelIndex indice)	Actualiza y muestra los datos del paciente en el encabezado del programa.
void mostrarEncabezadoLCR(QModelIndex indice)	Actualiza y muestra los datos del LCR en el encabezado del programa.
void graficarvariasMuestra(QModelIndex)	Grafica todas las muestras del paciente seleccionado.
void cliked(QModelIndex indice)	Se ejecuta cuando se da clic en el botón dentro de la tabla paciente.
void evaluacionManual()	Cambia el estado de evaluación del sistema a manual.
void eliminarEvaluacionManual()	Elimina el estado de evaluación del sistema y lo pone en automático.
void adicionarLCR()	Registra una muestra de LCR en el sistema.
void adicionarAnalisisEspecial()	Registra un análisis especial en el sistema.
void adicionarAnticuerpo()	Registra un análisis de anticuerpo en el sistema.
void adicionarDifCelulas()	Registra un análisis de diferencia de células en el sistema.
void adicionarPaciente()	Llama al método adicionar paciente de la clase cPaciente.
void llenarCombo(int indice)	Llena los combo box con los datos
void QSpecificaIndice()	Calcula la Qespecífica y el índice de anticuerpo en el análisis de anticuerpo.
void tranferirIndiceAnticuerpo()	Transfiere los datos calculados de la vista de análisis de anticuerpo para la vista de la muestra LCR.
void calcularPrcientoCelulas()	Calcula el porciento de células en el análisis de conteo diferencial.
void tranferirPorcientoCD()	Transfiere los valores del porciento calculado de la vista de análisis del conteo diferencial al de la muestra LCR.
void tranferirdeLCR()	Trasfiere los valores necesarios del LCR a los análisis especiales y de conteo diferencial.
void adicionarInstitucionR()	Llama al método adicionar institución remitente de la clase cinstitucionRemitente.
void on_btn_GuardarLCR_clicked()	Al dar clic en el botón Guardar se registran todos los datos de la muestra y los análisis en la base de datos.
void validarCampos()	Llama a todos los métodos de validación de campos.

void desactivarPestannas()	Desactiva las pestañas de la muestra y los análisis.
void activarPestannas()	Activa las pestañas de la muestra y los análisis.
void limpiarComponentesPaciente()	Limpia todos los componentes de la vista de la interfaz paciente.
void limpiarComponentesAnalisisEspecial()	Limpia todos los componentes de la vista de la interfaz análisis especial.
void limpiarComponentesAnticuerpo()	Limpia todos los componentes de la vista de la interfaz anticuerpo.
void limpiarComponentesConteoDiferencial()	Limpia todos los componentes de la vista de la interfaz conteo diferencia de células.
void limpiarComponentesLCR()	Limpia todos los componentes de la vista de la interfaz LCR.
void errorConsulta(QSqlQuery a)	Muestra los errores de una consulta pasada por parámetros.
int calcularEdad(QString stringFecha, QString stringfNacimiento)	Calcula la edad del paciente dado dos fechas.
void crearButoGroup()	Crea los grupos de botones (radio button)
void actualizar(QString idpaciente)	Actualiza los datos del paciente en la vista dado un id

3.3.4 Diagramas de interacción

Entre los artefactos que brinda UML para expresar las iteraciones entre los objetos proporcionándole cumplimiento a los requerimientos del sistema, se encuentran los Diagramas de Interacción. Estos permiten modelar aspectos dinámicos del sistema. Una interacción es un conjunto de objetos y sus relaciones, incluye los mensajes ordenados temporalmente, mediante los cuales pueden establecer comunicación. [58]

Diagramas de secuencia

Los diagramas de secuencia muestran interacciones entre objetos de una aplicación basadas en el tiempo. Conteniendo para ello, los objetos con sus ciclos de vida y los mensajes que se envían entre ellos ordenados secuencialmente. Los diagramas de secuencia son particularmente importantes para los diseñadores, pues clarifican los roles de los objetos en un flujo, facilitando así una entrada básica para determinar las responsabilidades de una clases y las interfaces. [58]

A continuación se muestra una representación de estos diagramas del sistema.

Tabla 14 Diagrama de secuencia. CU Adicionar paciente

Diagrama de secuencia. CU Adicionar paciente

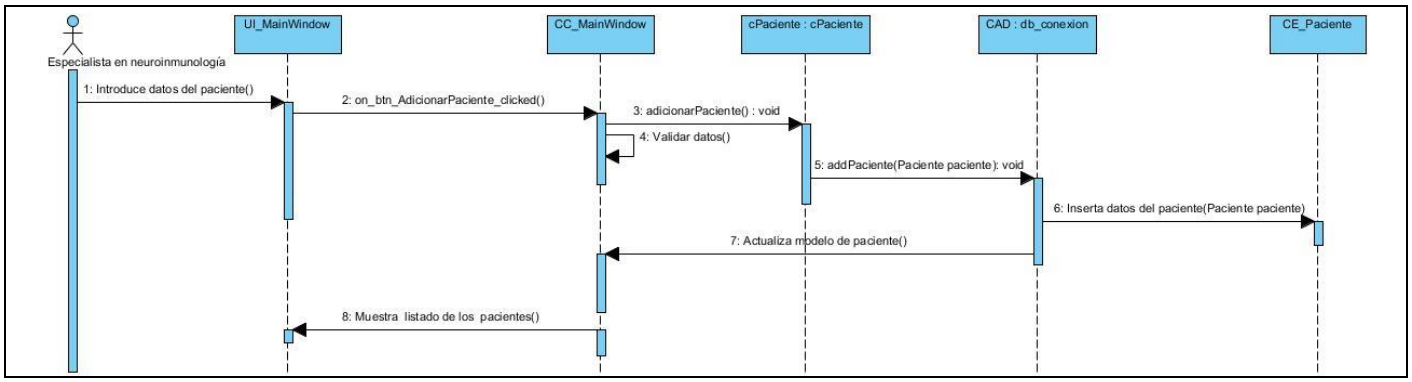
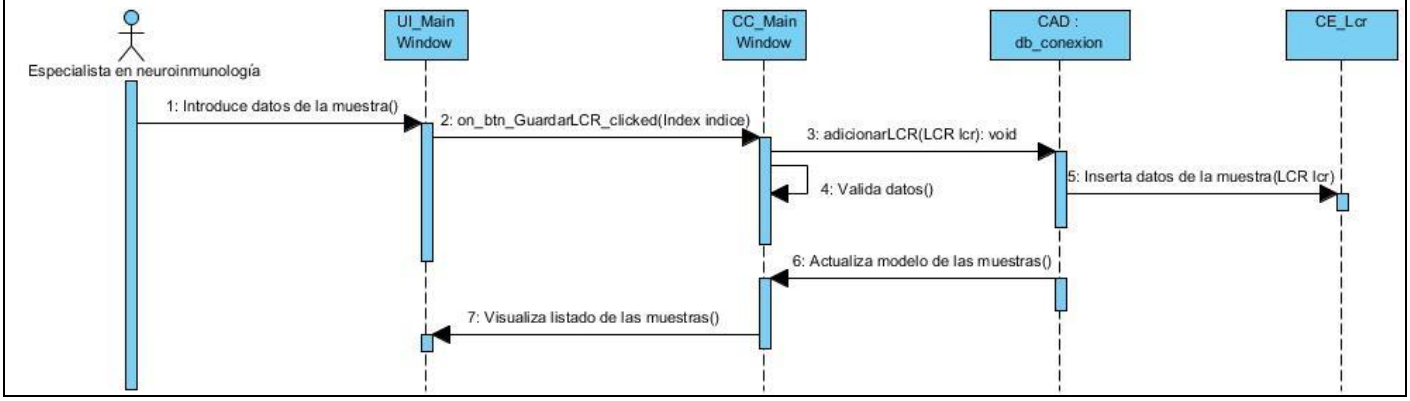


Tabla 15 Diagrama de secuencia. CU Registrar muestra

Diagrama de secuencia. CU Registrar muestra

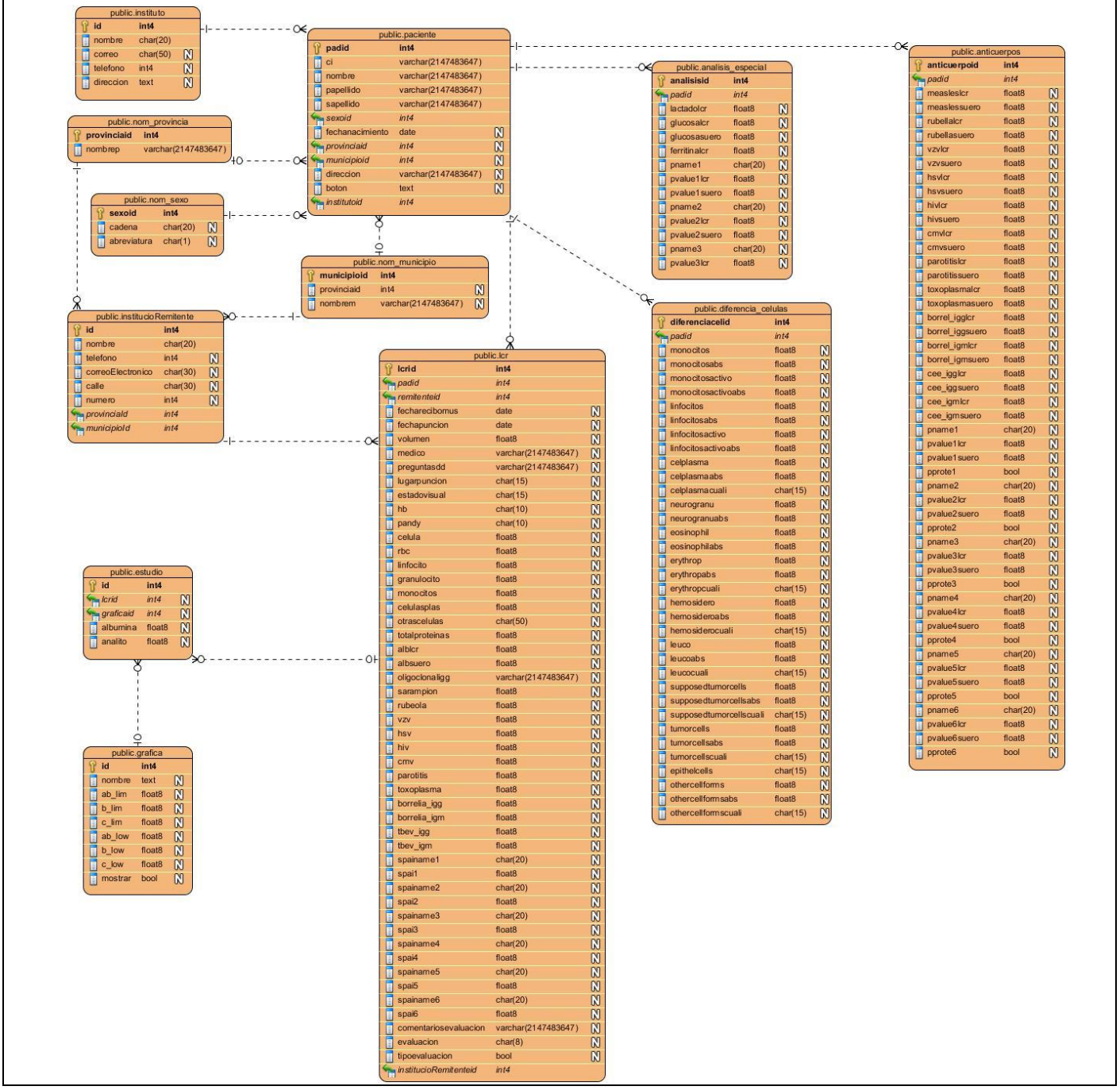


3.4 Modelo de Datos

Un modelo de datos es una definición lógica, independiente y abstracta de los objetos y operadores que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Este modelo proporciona una representación visual y física de los datos persistentes del sistema, que en el futuro serán la base de datos. Es uno de los artefactos más importante dentro del diseño. Los objetos permiten modelar la estructura de los datos y los operadores además de su comportamiento. Se obtiene a partir del diagrama de clases persistentes y su forma se expresa mediante un diagrama de UML, siendo sus elementos esenciales las entidades, atributos y relaciones entre entidades. [59]

Tabla 16 Diagrama Entidad-Relación

Diagrama Entidad-Relación



3.5 Descripción algunas de las tablas de la base de datos

En las tablas se representan elementos que existen y están bien diferenciados entre sí, que tienen propiedades y entre los cuales se establecen relaciones; son objetos concretos pero también puede ser algo no tangible, como un suceso cualquiera o un concepto abstracto. Cada tabla creada debe tener un nombre único en la base de datos.

Tabla 17 Descripción de la tabla Paciente

Nombre: Paciente		
Descripción: almacena los datos relacionados con los pacientes		
Atributo	Tipo	Descripción
padId	INTEGER	Llave primaria: representa el identificador de un paciente. Atributo que no cambia su valor y mantiene la integridad y relación entre los datos
hi	VARCHAR	Guarda número de historia clínica
nombre	VARCHAR	Contiene el nombre del paciente
PApellido	VARCHAR	Contiene el primer apellido del paciente
SApellido	VARCHAR	Contiene el segundo apellido del paciente
SexoId	INTEGER	Contiene el id para el nomenclador sexo
FechaNacimiento	DATE	Contiene la fecha de nacimiento del paciente
ProvinciaId	INTEGER	Contiene el id para el nomenclador provincia
MunicipioId	INTEGER	Contiene el id para el nomenclador municipio
Direccion	VARCHAR	Contiene la dirección del paciente

Tabla 18 Descripción de la tabla Lcr

Nombre: Lcr		
Descripción: almacena los datos correspondientes a las muestras LCR que posee un paciente.		
Atributo	Tipo	Descripción
LcrId	INTEGER	Llave primaria: representa el identificador de una muestra. Atributo que no cambia su valor y mantiene la integridad y relación entre los datos.
PadId	INTEGER	Llave foránea: contiene el id de un paciente.
FechaReciboMus	DATE	Contiene la fecha en que la muestra fue recibida en el laboratorio.
FechaPuncion	DATE	Contiene la fecha en que al paciente se le realizó la punción.
Medico	VARCHAR	Contiene el nombre del médico que orientó la realización de extracción de la muestra al paciente para posterior estudio.
Impresión diagnóstica	VARCHAR	Contiene dudas que presente el médico que orientó la realización de análisis.
LugarPuncion	CHAR	Contiene de qué forma se le realizó la punción al paciente.
EstadoVisual	CHAR	Contiene el estado en que estaba la muestra.

Hb	CHAR	Contiene detalles sobre la hemoglobina del paciente.
Volumen	DOUBLE	Contiene el volumen en ml.
Pandy	CHAR	Contiene el resultado del test de Pandy, prueba general para la detección de globulinas en el LCR.
Célula	DOUBLE	Contiene la cantidad total de células de la muestra
Glóbulos rojos	DOUBLE	Contiene la cantidad de glóbulos rojos de la muestra
Linfocito	DOUBLE	Contiene el % de este tipo de célula presente en la muestra con respecto a la cantidad total de células.
Granulocito	DOUBLE	Contiene el % de este tipo de célula presente en la muestra con respecto a la cantidad total de células.
Monocitos	DOUBLE	Contiene el % de este tipo de célula presente en la muestra con respecto a la cantidad total de células.
CelulasPlas	DOUBLE	Contiene el % de este tipo de célula presente en la muestra con respecto a la cantidad total de células
OtrasCelulas	DOUBLE	Contiene el % de este tipo de célula presente en la muestra con respecto a la cantidad total de células.
TotalProteinas	DOUBLE	Contiene el total de proteínas presente en la muestra LCR.
AlbLcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de LCR.
AlbSuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero.
IgGLcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR.
IgGSuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero.
IgALcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR.
IgASuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero.
IgMLcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR.
IgMSuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero.
IgELcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR.
IgESuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero
C3cLcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR
C3cSuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero
C4Lcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR
C4Suero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero
MblLcr	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra LCR
MblSuero	DOUBLE	Contiene el total de proteínas de este tipo presente en la muestra de suero
OligoclonalIgG	varchar	Contiene el estado de las proteínas
Sarampion	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
Rubeola	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
VZV	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
HSV	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
HIV	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
CMV	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente

Parotitis	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
Toxoplasma	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
Borrelia_IgG	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
Borrelia_IgM	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
TBEV_IgG	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
TBEV_IgM	DOUBLE	Contiene el índice de anticuerpos específicos de este tipo presente
SpAIName1	CHAR	Contiene el nombre de un anticuerpo introducido por el especialista
SpAI1	DOUBLE	Contiene el índice de anticuerpos específicos del anticuerpo introducido por el especialista
SpAIName2	CHAR	Contiene el nombre de un anticuerpo introducido por el especialista
SpAI2	DOUBLE	Contiene el índice de anticuerpos específicos del anticuerpo introducido por el especialista
SpAIName3	CHAR	Contiene el nombre de un anticuerpo introducido por el especialista
SpAI3	DOUBLE	Contiene el índice de anticuerpos específicos del anticuerpo introducido por el especialista
SpAIName4	CHAR	Contiene el nombre de un anticuerpo introducido por el especialista
SpAI4	DOUBLE	Contiene el índice de anticuerpos específicos del anticuerpo introducido por el especialista
SpAIName5	CHAR	Contiene el nombre de un anticuerpo introducido por el especialista
SpAI5	DOUBLE	Contiene el índice de anticuerpos específicos del anticuerpo introducido por el especialista
SpAIName6	CHAR	Contiene el nombre de un anticuerpo introducido por el especialista
SpAI6	DOUBLE	Contiene el índice de anticuerpos específicos del anticuerpo introducido por el especialista
ComentarioSEvaluacion	VARCHAR	Contiene un comentario emitido por el especialista acerca del análisis
Evaluacion	CHAR	Contiene una evaluación emitida por el especialista

3.6 Diagrama de despliegue

Un diagrama de despliegue muestra la configuración de nodos que participan en la ejecución y los componentes que residen en ellos. Un nodo puede contener instancias de componentes *software*, objetos, procesos (caso particular de un objeto). Las instancias de componentes *software* pueden estar unidas por relaciones de dependencia, posiblemente a interfaces (ya que un componente puede tener más de una interfaz). [53]

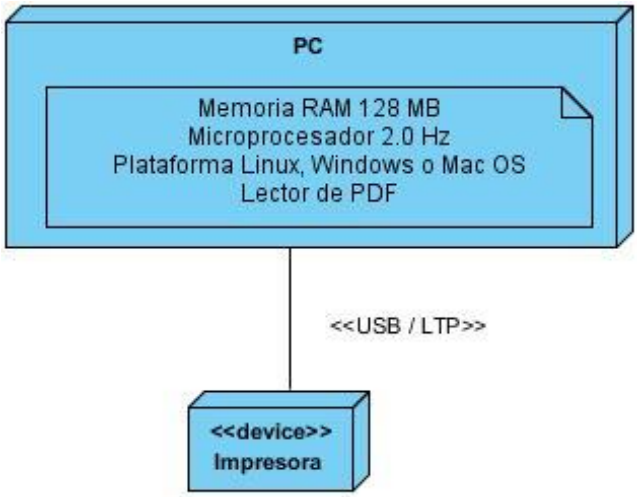
El diagrama de despliegue de la presente solución consta de:

Estación cliente: nodo donde estará la aplicación, cuya función es la ejecución del sistema e interactuar con el mismo según las necesidades del usuario. En ella estará la base de datos de la aplicación en un fichero.

Impresora: es el periférico directamente conectado a la estación cliente, el cual permitirá la impresión en papel de los informes realizados por el sistema. La conexión con la estación cliente es a través del puerto USB o LTP.

Tabla 19 Diagrama de despliegue

Diagrama de despliegue



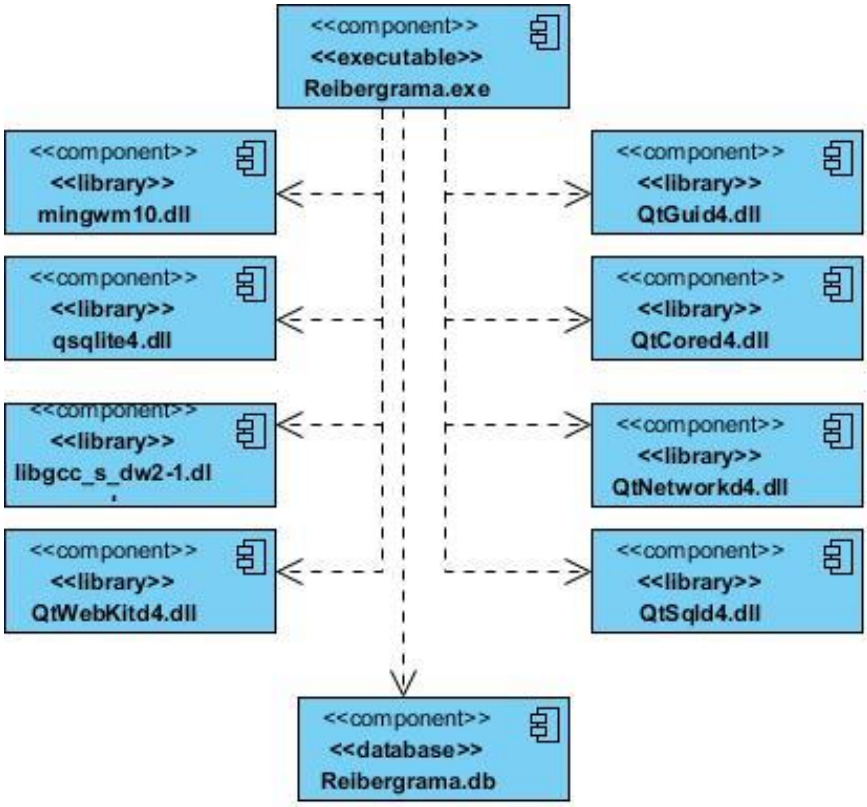
3.7 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos *software* que entran en la fabricación de aplicaciones informáticas permitiendo conocer a los desarrolladores y clientes la estructura física que tiene el sistema y cómo se relacionan sus partes. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. [60]

En el diagrama de componentes que se muestra a continuación se observan las relaciones que existen entre los elementos utilizados para la ejecución del sistema.

Tabla 20 Diagrama de componentes

Diagrama de componentes



3.8 Estándares de codificación

La adopción de estándares de estilo y codificación son de vital importancia para asegurar la calidad del *software*. El uso de los mismos tiene innumerables ventajas tales como:

- Asegurar la legibilidad del código entre distintos programadores, facilitando la depuración del programa.
- Proveer una guía para el encargado de mantenimiento/actualización del sistema, con código claro y bien documentado.
- Facilitar la portabilidad entre plataformas y aplicaciones.

Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Si bien los programadores deben implementar un estándar de forma prudente, este debe estar bien definido a nivel departamental, por tanto, al comenzar un proyecto

de *software* es necesario establecer un estándar de codificación único para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

Las convenciones de código o estándares de codificación son importantes para los programadores por un gran número de razones [61]:

- El 80% del coste del código de un programa va a su mantenimiento.
- Casi ningún *software* es mantenido toda su vida por el autor original.
- Las convenciones de código mejoran la lectura del *software* lo que permite entender código nuevo de manera más óptima y rápida.
- Si distribuyes tu código fuente como un producto, necesitas asegurarte de que está bien hecho y presentado como cualquier otro producto.

A continuación se presentan algunos de los estándares de codificación definidos y aplicados al sistema:

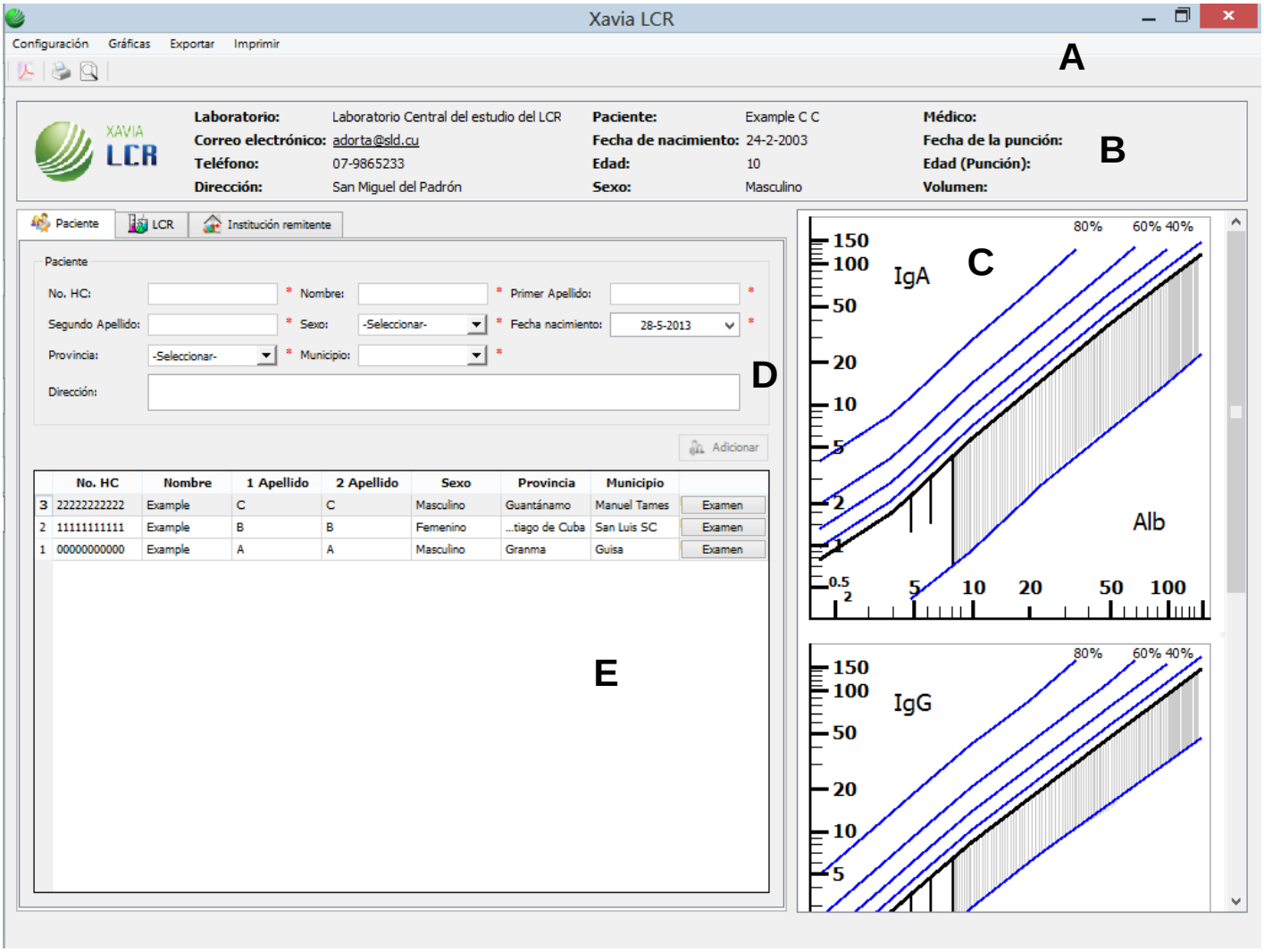
- Se debe utilizar como idioma el español, las palabras no se acentuarán.
- Todos los ficheros fuentes deben comenzar con un comentario en el que se lista el nombre de la clase, información de la versión, fecha y copyright.
- Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas. Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:
  - Entre las secciones de un fichero fuente
  - Entre las definiciones de clases e interfaces
- Se debe usar siempre una línea en blanco en las siguientes circunstancias:
  - Entre métodos
  - Entre las variables locales de un método y su primera sentencia
  - Antes de un comentario de bloque o de un comentario de una línea
  - Entre las distintas secciones lógicas de un método para facilitar la lectura
- Se debe dar un espacio en blanco en la siguiente situación:
  - Entre una palabra clave del lenguaje y un paréntesis

- Respecto a las normas de inicialización, declaración y colocación de variables, constantes, clases y métodos:
    - Todas las instancias y variables de clases o métodos empezarán con minúscula. Las palabras internas que lo forman, si son compuestas, empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo de peso "$", aunque ambos están permitidos por el lenguaje.
    - Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales.
    - Los nombres de las variables declaradas como constantes deben aparecer totalmente en mayúscula separando las palabras con un subguión ("_").
    - Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúscula. Mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas.
    - Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula y la primera letra de las siguientes palabras que lo forman en mayúscula.
- Respecto a la identación y longitud de la línea:
    - Se deben emplear cuatro espacios como unidad de identación. La construcción exacta de la identación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada ocho espacios.
    - Evitar las líneas de más de ochenta caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

3.9 Interfaces principales

A continuación se muestran las interfaces principales de la aplicación desarrollada:

Figura 3 Interfaz principal



En la interfaz principal **Figura: 3** se muestran varias áreas:

Área A: barra de título donde se especifica el nombre de la aplicación, barra de menú con las funcionalidades que posee y una barra de herramienta con las principales funcionalidades de la aplicación.

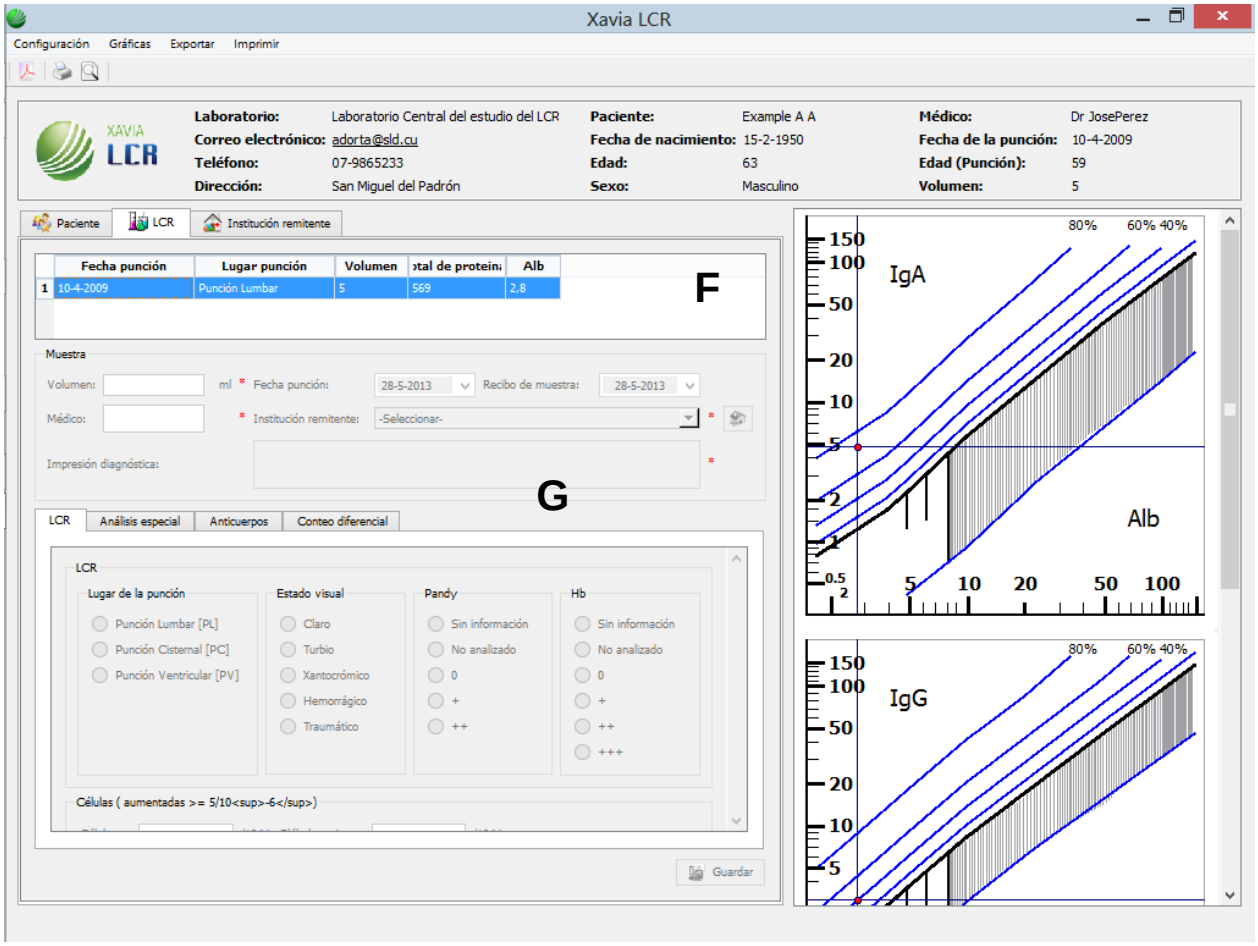
Área B: presenta información relacionada con el laboratorio donde está instalada la aplicación, así como de la muestra y el paciente que se está analizando en ese momento.

Área C: visualiza las gráficas con el estudio realizado al paciente.

Área D: se especifican los campos necesarios para buscar o adicionar un paciente en el sistema.

Área E: refleja un listado de los pacientes que ya están registrados en el sistema.

Figura 4 Interfaz registrar muestra



En la interfaz para registrar una muestra **Figura: 1** la áreas mostradas son las áreas A, B y C detalladas en la interfaz principal, así como las correspondientes para registrar una muestra en el sistema:

Área F: se especifican los campos necesarios para adicionar una muestra al paciente analizado en ese momento.

Área G: si el paciente analizado en ese momento posee algún estudio realizado con anterioridad, se visualiza un listado de las muestras que ya estén registradas en la aplicación de ese paciente.

3.10 Conclusiones parciales

Con la realización de los correspondientes artefactos del presente capítulo, se adquirió una visión detallada de las responsabilidades de cada clase del análisis para el correcto funcionamiento del sistema a desarrollar. Por otro lado, el modelado del diseño propicia las bases para la etapa de implementación, al ofrecer un mayor acercamiento o nivel de abstracción respecto a lo que el desarrollador tiene que implementar. Además, la propuesta de diagrama de despliegue presentada, brinda los elementos necesarios para un correcto proceso de despliegue de la aplicación informática.

Conclusiones

A partir de los resultados obtenidos se arribó a las siguientes conclusiones:

Se investigaron los sistemas existentes a nivel internacional y nacional (*CFS Laboratory, CFS Research Tool*, Neuroinmunolab) comprobando que no abarcan las necesidades del cliente; así como las tendencias y tecnologías seleccionadas para la construcción de la solución.

El análisis de los procesos que intervienen en la gestión de la información en el LABCEL permitió la identificación de los requisitos funcionales necesarios para el desarrollo del sistema adaptado a las características propias del centro.

Fueron elaborados los diagramas correspondientes a cada fase según la metodología RUP para un mejor entendimiento de los flujos del proceso de gestión así como de los componentes del sistema.

Se obtuvo una aplicación de escritorio definida bajo una arquitectura Modelo Vista Controlador, que junto al uso de patrones de diseño guió el desarrollo del sistema que favorece el mejoramiento de los procesos relacionados con la gestión de la información en el estudio neuroinmunológico del LCR.

Recomendaciones

Para contribuir al éxito en la continuidad de la investigación, se hacen las siguientes recomendaciones:

- Hacer un procedimiento de implantación para un uso extensivo del sistema en otras instituciones del país que posean la misma línea de investigación.

Referencias bibliográficas

1. Contreras, A.J.D., et al., *Productividad, visibilidad e impacto de la producción científica del Laboratorio Central de Líquido Cefalorraquídeo en el período 2004-2009.* ACIMED Revista Cubana de Información en Ciencias de la Salud, 2010. 21.

2. Contrera, A.J.D., *Reibergrama: Análisis esencial en el estudio del líquido cefalorraquídeo.* Revista de Neurología, 1999. 28.

3. Contreras, A.J.D., et al., *Bases moleculares de la Neuroinmunología (II). El reibergrama y su uso en Neuroinmunología.* Revista Cubana de Pediatría., 2005.

4. Salud. *Enciclopedia de Salud, Dietética y Psicología*. 2011; Available from: http://www.enciclopediasalud.com/definiciones/liquido-cefalorraquideo.

5. Neurocirugía. *Neurocirugía Contemporánea*. 2010; Available from: http://neurocirugiacontemporanea.com/doku.php?id=liquido_cefalorraquideo.

6. Contreras, A.J.D., et al., *Síntesis intratecal de C3c e inmunoglobulinas en niños con meningoencefalitis.* Vaccimonitor, 2008. 17.

7. Contreras, A.J.D., et al., *Respuesta inmune humoral intratecal en pacientes pediátricos con meningoencefalitis por Coxsackie B5.* Revista de Neurología, 1999. 29.

8. Wormek, A.K. and S.E. Schleutermann, *CSF Laboratory 3.7 - Cerebrospinal Fluid Analysis.* WORMEK, 2012.

9. Reiber, H. and W. Albaum, *Statistical evaluation of intrathecal protein synthesis in CSF / Serum quotient diagrams.* Neurochemisches Labor , University Hospital Göttingen, 2012.

10. Contrera, A.J.D., *Desarrollo de un sistema computarizado de aplicación en la esfera de la Inmunología.* Frente de la electrónica, 1993. Revista CID. Electrónica y proceso de datos en cuba.

11. G., R.F., C.S. J., and A.A. Cabrera. *Metodologías tradicionales vs. metodologías ágiles* 2010; Available from: http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/METODOLOGIAS_TRADICIONALES_VS._METODOLOGIAS_AGILES.pdf.

12. Jacobson, I., G. Booch, and J. Rumbaugh, *El Proceso Unificado de Desarrollo*. 2000, Madrid: Pearson Educación S.A.

13. España, J.M. *En camino hacia la mejora continua de procesos*. Gestión empresarial 2007; Available from: http://coit.es/publicaciones/bit/bit163/64-66.pdf.

14. Club-BPM. *Club-BPM*. 2009; Available from: http://www.club-bpm.com/ApuntesBPM/ApuntesBPM01.pdf.

15. Technologies, A.L. *Modelado de negocios con BPMN 2.0 y UML*. 2012; Available from: http://www.infored.com.mx/p/modelado-de-negocios-con-bpmn-2-0-y-uml.html.

16. Mastermagazine. *Definición de UML*. 2008; Available from: http://www.mastermagazine.info/termino/7006.php.

17. Orallo, E.H. *El Lenguaje Unificado de Modelado (UML)* 2012; Available from: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

18. Revista_informática. *Clasificación de los lenguajes de programación*. 2010; Available from: http://larevistainformatica.com/clasificacion-de-los-lenguajes-de-programacion.html.

19. Jalón, J.G.d., et al. *Aprenda C++ como si estuviera en primero*. 1999; Available from: http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf.

20. Mora, S.L. *C++ paso a paso*. 2006; Available from: http://gplsi.dlsi.ua.es/libros/cpp1/.

21. Bulfon, G. *Introduccion a C#*. 2010; Available from: http://www.dotech.com.ar/notes/CSharp1.htm.

22. Marañón, G.Á. *Características del lenguaje Java*. 2006; Available from: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

23. Blanco, C. *Entornos de Desarrollo Integrado (IDE´s)* 2013; Available from: http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion/.

24. Gitorious. *Qt Project*. Qt 2013; Available from: http://qt.digia.com/Product/.

25. Edition, M.D.O.S. *MonoDevelop*. 2012; Available from: http://monodevelop.com/.

26. Interactivas, D.d.A. *Programación visual con Borland C++ Builder*. 2012; Available from: http://dis.um.es/~jfernand/0506/dai/primerasesion.pdf.

27. Microsoft. *Inicio de Visual C++*. 2012; Available from: http://msdn.microsoft.com/es-es/visualc/aa336448.aspx.

28. Perissé, C.M. *Herramientas case*. 2010; Available from: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

29. S.A.C., T.I. *Visual Paradigm para UML*. 2010; Available from: http://www.software.com.ar/visual-paradigm-para-uml.html.

30. Innova, G.d.s. *Rational Rose Enterprise*. 2007; Available from: http://www.rational.com.ar/herramientas/roseenterprise.html.

31. Sparxsystems. *Enterprise Architect - Herramienta de diseño UML*. 2007; Available from: http://www.sparxsystems.com.ar/products/ea.html.

32. Navathe, S. and R. Elmasri. *Fundamentos de Sistemas de Bases de Datos*. 2002.

33. Pecos, D. *PostgreSQL vs. MySQL*. 2010; Available from: http://danielpecos.com/docs/mysql_postgres/index.html#AEN11.

34. Ginestà, M.G. *Base de Datos en postgresSQL*. 2010; Available from: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02152.pdf.

35. Oracle, E.S. *El Sistema Gestor de Bases de Datos Relacionales Oracle*. 2004; Available from: http://www2.rhernando.net/modules/tutorials/doc/bd/oracle.pdf.

36. Maldonado, D.M. *SQLite, el motor de base de datos ágil y robusto*. 2008; Available from: http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.

37. Meunier, R., et al., *Pattern - Oriented Software Architecture A System of Patterns Frank Buschmann*. Volumen 1 ed. 1996, Alemania: Siemens AG.

38. Reynoso, C. and N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Estilos de Flujo de Datos. 2004, Buenos Aires: Universidad de Buenos Aires.

39. Shaw, M. and D. Garlan, *Software Architecture: Perspectives on an emerging discipline.* 1996: Prentice Hall Engineering/Science/Mathematics.

40. Reynoso, C. and N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Estilos Centrados en Datos. 2004, Buenos Aires: Universidad de Buenos Aires.

41. Burbeck, S. *Application programming in Smalltalk-80: How to use Model-View-Controller (MVC)*. 2000; Available from: http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

42. Garlan, D. and M. Shaw, *An introduction to software architecture*, ed. School of Computer Science Carnegie Mellon University Pittsburgh. 1994. 42.

43. Reynoso, C. and N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Arquitecturas Basadas en Componentes. 2004, Buenos Aires: Universidad de Buenos Aires.

44. Reynoso, C. and N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Estilos de Código Móvil. 2004, Buenos Aires: Universidad de Buenos Aires.

45. Reynoso, C. and N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Estilos heterogéneos. 2004, Buenos Aires: Universidad de Buenos Aires.

46. Reynoso, C. and N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Estilos Peer-to-Peer. 2004, Buenos Aires: Universidad de Buenos Aires.

47. Pinto, C.J.Q.L.F.V.F. and L. Neriz, *Análisis de Modelos de Procesos de Negocios en relación a la dimensión informática. .* Departamento de Sistemas de Información y Auditoría, 2002.

48. Fajardo, J.U., *Business Process Modeling Notation (BPMN)* 2010.

49. Laguna, M.A. *Requisitos*. 2010; Available from: http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.

50. Synergix. *Visión de Synergix de los Sistemas de Información y la Ingeniería del Software*. Tecnología y Synergix 2008; Available from: http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/.

51. Vega, M. *Casos de uso UML*. LSI-UGR 2010; Available from: http://lsi.ugr.es/~ig1/docis/casos%20de%20uso.pdf.

52. Gitorious. *Qt Project*. Model/View Programming 2013; Available from: http://qt-project.org/doc/qt-4.8/model-view-programming.html.

53. Jacobson, I., G. Booch, and J. Rumbaugh, *El Proceso Unificado de Desarrollo*. Capítulo 9 Diseño. 2000, Madrid: Pearson Educación S.A.

54. UCI. *Introducción a la Disciplina de Análisis y Diseño.* 2012; Available from: http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_10/Conferencia_10/Materiales_complementarios/Introduccion_a_la_Disciplina_Analisis_y_Diseno.pdf.

55. Toross, G. *Diseño de Sistemas*. 2009; Available from: http://www.chaco.gov.ar/utn/disenodesistemas/apuntes/oo/ApunteRUP.pdf.

56. Sommerville, I., *Ingeniería del Software*. 2005, Madrid: Pearson Educación.

57. Craig, *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. 1999, México: Pearson Educación.

58. Craig, *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. Parte IV Fase de Análisis y diseño. 1999, México: Pearson Educación.

59. Date, C.J., *Introducción a los sistemas de bases de datos.* 2001, México: Pearson Educación.

60. Pressman, R.S., *Ingeniería del software. Un enfoque práctico*. Capítulo 11 Diseno Componentes: Mc Graw Hill.

61. Calleja, M.A. *Estándares de codificación*. 2010; Available from: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf.

Bibliografía

1. **Booch, Grady, Rumbaugh, James and Jacobson, Ivar. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, 2000. 84-7829-036-2.
2. **Booch, Grady, Rumbaugh, James, Jacobson, Ivar. 2005.** El Lenguaje Unificado de Modelado. Addison Wesley. Madrid. ISBN: 0-201-57168-4.
3. **Club-BPM. 2009.** Club-BPM. [Online] noviembre 3, 2009. http://www.club-bpm.com/ApuntesBPM/ApuntesBPM01.pdf
4. **Craig**, *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. 1999, México: Pearson Educación.
5. **Dorta Contrera, Alberto Juan. 1993,** *Desarrollo de un sistema computarizado de aplicación en la esfera de la Inmunología.* Frente de la electrónica. Revista CID. Electrónica y proceso de datos en cuba.
6. **Dorta Contrera, Alberto Juan, Noris García E, Bu Coifiú Fanego R, Padilla Docal, Barbara**., *Bases moleculares de la Neuroinmunología (II). El reibergrama y su uso en Neuroinmunología.* Revista Cubana de Pediatría., 2005.
7. **Dorta Contrera, Alberto Juan**., *Reibergrama: Análisis esencial en el estudio del líquido cefalorraquídeo.* Revista de Neurología, 1999. 28.
8. **Dorta Contrera, Alberto Juan, Noris García E, Bu Coifiú Fanego R, Padilla Docal, Barbara**, *Productividad, visibilidad e impacto de la producción científica del Laboratorio Central de Líquido Cefalorraquídeo en el período 2004-2009.* ACIMED Revista Cubana de Información en Ciencias de la Salud, 2010. 21
9. **Dorta Contrera, Alberto Juan, et al. 2006.** *Barrera Sangre-Líquido Cefalorraquídeo.* Ciudad de la Habana : Academia , 2006. 959-270-070-2.
10. **Fajardo, J.U. 2010,** *Business Process Modeling Notation (BPMN)*.
11. **Gamma, Erich, et al. 1995.** *Design Patterns: Elements of Reusable Object-Oriented Software.* Madrid : Addison Wesley, 1995. 0-201-63361-2.
12. **Gitorious**. **2013** *Qt Project*.; http://qt-project.org.
13. **González , Carlos D. 2008.** usabilidadweb.com.ar. *Introducción a C++ y a la resolución de problemas.* [Online] 2008. http://www.usabilidadweb.com.ar/cpp.php.
14. **Larman, Craig. 1999.** *UML Y Patrones.* Mexico : PRENTICE HALL, 1999. p. 169. 970-17-0261-1.
15. **Pressman, R. 2004.** *Ingeniería de Software. Un enfoque práctico.* La Habana,Cuba : Félix Varela, 2004.
16. **Qt. 2010.** *APRENDA Qt4 DESDE HOY MISMO.* 2010.
17. **Qt Nokia.** QT. [Online] http://qt.nokia.com.
18. **Reiber, H. and W. Albaum. 2012**, *Statistical evaluation of intrathecal protein synthesis in CSF / Serum quotient diagrams.* Neurochemisches Labor , University Hospital Göttingen.

19. **Reynoso, C. and N. Kiccillof**, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*, Buenos Aires: Universidad de Buenos Aires.
20. **Sommerville, Ian. 2005.** *Ingeniería del Software.* Madrid : Pearson Educación, 2005. 84-7829-074-5.
21. **Stallman, Richard . 2004.** *Software libre para una sociedad libre.* España : Traficantes de Sueños, 2004. p. 99.**Stroustrup, Bjarne. 1997.** *The C++ Programming language.* 1997.
22. **Toross, Gustavo. 2009.** Universidad Tecnológica Nacional. *Diseño de Sistemas.* [Online] abril 03, 2009. http://www.chaco.gov.ar/utn/disenodesistemas/apuntes/oo/ApunteRUP.pdf.
23. **UCI. 2012.** Entorno virtual de aprendizaje. *EVA.* [Online] 2012. http://eva.uci.cu/file.php/158/Documentos/Bibliografia_general/Textos_Basicos/El_Proceso_Unificado_de_Desarrollo/03_Parte_1_El_proceso_unificado_de_desarrollo.pdf.
24. **UCI. 2012.** Entorno Virtual de Aprendizaje. [Online] 2012. http://eva.uci.cu/file.php/161/Documentos/Guias_del_estudiante_y_del_profesor/UD_2_Ingenieria_de_Requisitos/Semana_5/Conferencia_5/Modelado_del_contexto_con_casos_de_uso_del_negocio.pdf
25. **UML.** UML. *BPMN and Database Tool for Software Development.* [Online] http://www.visual-paradigm.com.
26. **Zona Qt. 2013,***Zona Qt.*[Online] **http://www.zonaqt.com**

Glosario de términos

Albúmina: proteína que se encuentra en gran proporción en el plasma sanguíneo, siendo la principal proteína de la sangre y una de las más abundantes en el ser humano y la más representada en el líquido cefalorraquídeo.

Analito: es el componente de interés analítico de una muestra. Son especies químicas cuya presencia o concentración se desea conocer. El analito es una especie química que puede ser identificado y cuantificado, es decir, determinar su cantidad y concentración en un proceso de medición química, constituye un tipo particular de mensurando en la metrología química.

Anticuerpo: son las moléculas de la inmunidad humoral específica y una de sus principales funciones fisiológicas es la defensa contra los microorganismos extracelulares y las toxinas producidas por los distintos agentes microbianos.

Aplicación: programa informático diseñado para permitir a un usuario realizar diversos tipos de trabajo.

Punción lumbar (punción raquídea o punción espinal): es un análisis médico común en el que se toman pequeñas muestras de líquido cefalorraquídeo para analizarlo mediante la inserción de una aguja, con un mandril en su interior (aguja de punción lumbar) hueca entre las vértebras lumbares. Es un examen para evaluar el líquido que rodea el cerebro y la médula espinal. El examen también se utiliza para medir la presión en dicho líquido.

Proteínas: las proteínas son biomoléculas formadas por cadenas lineales de aminoácidos arrolladas entre sí formando una estructura terciaria compleja.

Neurología: es la especialidad médica que estudia la estructura, función y desarrollo del sistema nervioso (central, periférico y autónomo) y muscular en estado normal y patológico, utilizando todas las técnicas clínicas e instrumentales de estudio, diagnóstico y tratamiento actualmente en uso o que puedan desarrollarse en el futuro. La Neurología se ocupa de forma integral de la asistencia médica al enfermo neurológico, de la docencia en todas las materias que afectan al sistema nervioso y de la investigación, tanto clínica como básica, dentro de su ámbito.

Inmunología: es la rama amplia de la biología y de las ciencias Biomédicas que se ocupa del estudio del sistema inmunitario, entendiendo como tal al conjunto de Órganos, Tejidos y Células que tienen como función reconocer elementos extraños o ajenos dando una respuesta (respuesta inmunitaria).