Universidad de las Ciencias Informáticas

Sistema de almacenamiento de la información del Sistema de Información de laboratorios

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR:
Karel Cruz Martínez

TUTOR:
Ing. William Sóñora Cruz

ASESOR:
Lic. Ernesto Sarduy Alonso

Ciudad de La Habana
Julio 2007

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste, firmo la presente a los 5 días del mes de Julio del año 2007.

Karel Cruz Martínez	William Sóñora Cruz
_____	_____
Firma del autor	Firma del tutor

DATOS DE CONTACTO

Del tutor:

William Sóñora Cruz.

Ingeniero Informático graduado en el Instituto Superior Politécnico José A. Echevarría “ISPJAE”

Correo electrónico: williamcs@uci.cu

Del asesor:

Ernesto Sarduy Alonso.

Licenciado en Ciencias de la Computación en la Universidad Central de las Villas “Martha Abreu”.

Correo electrónico: ernestos@uci.cu

AGRADECIMIENTOS

A mis padres, por estar siempre ahí, por su constancia y dedicación en esa difícil tarea que ha sido educarme. Por ser mi ejemplo, por ser esa fuente inagotable de cariño, por aconsejarme. Ustedes y mi hermana son lo más grande que tengo. Los amo mas que a nada en este mundo, ojalá los tenga por siempre.

A toda mi familia, especialmente: A mis abuelas; Oscilia y Crecencia, que tanto me quieren y me cuidan, las amo. A mis primos Harold, Orly, Jorgito (el yoyo), Mailyn, Laura, Elizabeth, los quiero. A mis tíos Guille, Esther, Iza, Maira, Olga.

A TODOS mis amigos de la UCI, gracias por estar a mi lado en los momentos buenos y malos en estos 5 años, por sus consejos, por ayudarme de corazón, por ser mi familia, sin ustedes de veras no lo hubiese logrado, gracias mil en especial a: Abel (warrior), Rodolfo (el fofi), Liver (Doc. Dominic), Ariesky (pincho), Victor, Reinel (el negro), Maikel (el pollo), Eduardo, Néstor, Pedro, Alain, Anier (el chamo), Arley, Adonis, Raydel, Julio, Yunior, Yeneirys (nena), Dinia, Yai y a todos los que me faltaron, les ruego que me disculpen, sepan que los llevaré en mi corazón a donde quiera que vaya.

A mi "bro" Leshter, por ser mi hermano desde el principio, por estar ahí siempre, por ser "mi compañero de armas".

A quien todo este tiempo se mantuvo a mi lado, esa personita tan linda que me dio todo sin pedir nada a cambio: Daylenis, te "kero" mucho.

Al equipo de desarrollo de LIS.

A todas aquellas personas que de una forma u otra ayudaron al logro de este trabajo.

A mis profesores, que tanto me enseñaron. Gracias.

A mis amigos del barrio en especial a Yusmel (el poto), Adrian, Amaury.

A mis profesores del pre-universitario, especialmente a mi profesor de matemáticas Guido.

A mi tutor William.

DEDICATORIA

A mis padres, porque se lo merecen todo.

A la UCI.

A las personas tanto dentro como fuera del recinto universitario que les pueda servir este trabajo como guía, referencia, bibliografía o estudio.

RESUMEN

En la actualidad, los Laboratorios Clínicos de los hospitales, no tienen un sistema que gestione su información, lo que hace más complejo y difícil el trabajo que allí realiza. Sumado a ello,los volúmenes masivos de información de estos centros, hacen que sea cada vez más necesario, desarrollar un sistema digital que garantice el almacenamiento y control de su información.

Con este trabajo se persigue el objetivo fundamental de diseñar una base de datos manipulada a través de un sistema automatizado para la mejora de los procesos de la gestión de información relacionados con los laboratorios clínicos en los Hospitales y Centros Hospitalarios Cubanos. La misma, permitirá un adecuado control Clínico-Administrativo del Laboratorio y optimizará en gran medida el desempeño del personal que labora en estos centros.

Para ello se hará estudio y uso de las más diversas técnicas, metodologías y herramientas actuales, dentro de las cuales se citan a: Rational Rose, CASE Studio, PostgreSQL, EMS PgManager, entre otras.

Con el trabajo se pretende contribuir al desarrollo de un Sistema para Laboratorios automatizado que contribuirá con la formación y desempeño profesional del personal del laboratorio, haciendo su trabajo más fácil y dinámico, lo que permitirá: tener información instantánea de todo lo procesado por el Laboratorio; un eficiente control clínico-administrativo del área; un preciso control estadístico, que a su vez proporciona un conocimiento agregado, tener una valoración precisa del trabajo realizado.Todo lo anterior se traduce en un considerable ahorro de tiempo y recursos.

ÍNDICE

ÍNDICE DE CLASES

Descripciones de clases persistentes

ÍNDICE DE TABLAS

Descripciones de tablas de la base de datos

INTRODUCCIÓN

En Cuba, la salud representa un papel primordial en el bienestar del pueblo, por lo que la revolución desde sus inicios le ha otorgado una atención privilegiada con respecto a otros sectores. Desde su creación en 1960, el Ministerio de Salud Pública (MINSAP), ha sido el órgano rector en el sector de la salud, el cual con el de cursar de los años se ha solidificado fuertemente en la puesta en práctica de resoluciones y programas de salud, siempre con el objetivo de una construcción social de la salud.

En el país, existe un Sistema Único de Salud, que garantiza una cobertura total y un acceso gratuito a todos los servicios que brinda. La existencia del Programa del Médico y Enfermera de la Familia, el amplio Programa de Objetivos, Propósitos y Directrices para Incrementar la Salud de la Población Cubana hasta el año 2000, la amplia red de hospitales, policlínicos y clínicas estomatológicas, los Institutos de Investigación, la formación de los recursos humanos y la Industria Médico Farmacéutica, como parte del Sistema Nacional de Salud (SNS), hacen necesaria la existencia de un Sub-Sistema Nacional de Información de Ciencias Médicas que sea capaz de garantizar la información científico-técnica con la eficiencia y la eficacia que requieren los profesionales y técnicos de la Salud.

A pesar de ello, a partir del año 2000, el MINSAP, se ha propuesto la informatización del sector de la salud, en busca de la optimización de los Servicios de Salud que se brindan a la población; mayor productividad y competencia en el desempeño de sus profesionales y técnicos, control en la administración de sus recursos, eficacia, y otros aspectos importantes.

En los últimos años la Informática Médica está siendo objeto de gran atención en la comunidad científica y profesional, tanto nacional como internacionalmente.

En la actualidad goza de aceptación general, la idea de que sólo la técnica se muestra insuficiente para dar respuesta satisfactoria a las necesidades de salud de la población.

Los sistemas de salud pueden ser interpretados como un nuevo camino, una estrategia, una filosofía o simplemente una forma diferente de pensar y actuar para alcanzar la salud de los pueblos. Para algunos, incluyendo al autor de este trabajo, es un nuevo enfoque, cualitativamente superior al tradicional de la salud pública, es incluso un nuevo paradigma. Es “una estrategia” o una “Función Esencial” de la Salud Pública.

Teniendo su origen en los ’70, los Sistemas Informáticos para la Salud constituyen hoy en día uno de los pilares de apoyo fundamentales en la rama de la Informática Medica. En la actualidad existe una amplia gama de software dedicados a este fin, tanto así, que ya no se concibe en el mundo moderno la existencia de una entidad hospitalaria o vinculada al sector de la salud, sin la mediación de un sistema informático que se encargue de gestar al menos una parte de los procesos Clínico-Administrativos.

Un Laboratorio Clínico es el lugar donde se realizan análisis clínicos que contribuyen al estudio, prevención, diagnóstico y tratamiento de los problemas de salud de los pacientes. También se le conoce como Laboratorio de patología Clínica.

Los laboratorios de análisis clínicos, de acuerdo con sus funciones, se pueden dividir en:

1) Laboratorios de Rutina
2) Laboratorios de Especialidad

Los laboratorios de rutina tienen cuatro departamentos básicos:

- Hematología
- Inmunología
- Microbiología
- Química Clínica (o Bioquímica)

Los laboratorios de rutina pueden encontrarse dentro de un hospital o ser externos a este. Los laboratorios hospitalarios, con frecuencia tienen secciones consideradas de urgencia, donde se realizan estudios que servirán para tomar decisiones críticas en la atención de los pacientes graves. Estudios tales como citometría hemática, tiempos de coagulación, glicemia, urea, creatinina y gases sanguíneos.

En los laboratorios de pruebas especiales se realizan estudios más sofisticados, utilizando metodologías como amplificación de ácidos nucleicos, estudios cromosómicos, citometría de flujo y cromatografía de alta resolución, entre otros. Estas pruebas requieren instalaciones y adiestramiento especial del personal que las realiza. Con frecuencia, estos laboratorios forman parte de programas de investigación.

A pesar de los muchos avances logados, por el gobierno en aras de una alta calidad en los servicios hospitalarios, los Laboratorios Clínicos están distantes de cumplir con esa meta; ya que la mayoría del trabajo realizado tiene un carácter manual y una forma de de almacenamiento ya obsoleta para la actualidad, como lo es el papel, lo que trae como consecuencia directa un mayor esfuerzo por parte del personal a cargo de la manipulación de dicha información.

En el mundo existen varios sistemas que se encargan de la solución total o parcial de este problema en los laboratorios. Entre ellos se encuentra Care2x, Alfa21®, Coya Laboratorios (Team 2006), MediLab LIS y El GalenLAB (Softel 2005) que es la única alternativa cubana de solidez hasta el momento.

Con este trabajo se pretende lograr un Sistema para Laboratorios totalmente automatizado, ello acarreará como resultado toda una revolución al antiguo sistema imperante lo que puede traducirse como un aporte significativo a esta área de la Salud en general.

Además, contribuirá con la formación de personal concerniente al laboratorio, suministrando en gran medida una mejora a su desempeño profesional, haciendo su trabajo mucho más fácil y dinámico, lo que supone ganancias en conocimientos y tiempo, ya que entre muchas otras cosas, permitirá: tener información de forma instantánea referente a cualquier paciente, análisis o muestra que haya sido procesado por el Laboratorio; permitirá un eficiente control clínico-administrativo del área; archivará, controlará y llevará un preciso control estadístico, que a su vez proporciona un conocimiento agregado que permitirá a técnicos y especialistas tener e incluso dar una valoración lo más acertada posible acerca de un determinado análisis o examen realizado.

Situación problémica:

Actualmente en Cuba no se cuenta con un proceso de gestión de Información de los Servicios de Laboratorios clínicos automatizado, estos en su totalidad, son realizados de forma manual o a través de aplicaciones (software) no estandarizadas a nivel nacional, ni internacional.

Los diferentes datos que deben ser, almacenados y manipulados, son recogidos en formularios en formato papel, en el caso tradicional o en la minoría de los casos, almacenados en sistemas de computo (computadoras), que por lo general no presentan un estado de la técnica optimo para este desempeño. Ello, trae como consecuencia directa, que sea constante el riesgo de presencia de errores, no solo a la hora del llenado de los formularios, sino también a la hora de la obtención de ciertos datos, como son los relacionados con las estadísticas propias del Laboratorio. Esto

independientemente del tiempo prolongado que supone la búsqueda y análisis de los registros individuales de cada paciente.

El trabajo manual conlleva muchos inconvenientes al personal de la Salud, ya sea por la demora de la recogida de datos o bien por los errores que se pudieran cometer durante la toma y procesamiento de los mismos; esto provoca entre otras cosas que no se cuente con registros médicos actualizados. Otra agravante lo constituye el hecho de que este tipo de desempeño laboral no satisface la demanda de atención actual existente, por lo que la cantidad de pacientes atendidos en un día se ve reducida, llegando en muchas ocasiones a estar por debajo de la media diaria.

Otra cuestión importante es la referente a que: como no hay un control automatizado de este proceso, se lidia con la posibilidad de tener almacenados registros dobles o repetidos, como por ejemplo: varias solicitudes de un mismo análisis, hechas por un solo paciente; lo cual tendría un efecto bastante negativo en el momento en que esa información fluye por los distintos departamentos del laboratorio, teniendo como consecuencia inmediata la perdida de tiempo y el malgasto de recursos.

Problema científico:

¿Como automatizar el proceso de gestión de información de los laboratorios clínicos en los hospitales cubanos?

Objeto de la investigación:

Procesos de gestión de la información en los laboratorios.

Objetivo de la Investigación:

Diseñar una base de datos manipulada a través de un sistema automatizado para la mejora de los procesos de la gestión de información relacionados con los laboratorios clínicos en los Hospitales y Centros Hospitalarios Cubanos.

Campo de Acción:

Procesos de gestión de la información de los Laboratorios clínicos en los Hospitales Cubanos.

Tareas de Investigación:

- Definir las herramientas a utilizar para el diseño de la base de datos.
- Obtener los artefactos de la metodología RUP que describen la base de datos.

- Analizar la estrategia de integración de la solución con otros módulos o sistemas.
- Implementar la capa de acceso a datos y procedimientos almacenados del Módulo.
- Hacer una Validación teórica del diseño de la BD propuesto.
- Realizar una valoración de los resultados y hacer propuestas de iteración en el diseño.

Posibles resultados:

Obtener el diseño de la base de datos del Sistema de Información de Laboratorios que forma parte del Sistema de gestión Hospitalaria.

Diseño metodológico de la Investigación:

Con el objetivo de obtener la toda información posible que nos permita el alcance de las metas propuestas, se emplearán los métodos Teóricos y Empíricos, así como la metodología informática siguiente:

✡ *Métodos Teóricos*

Entrevistas: Posibilitó obtener la información sobre cómo se espera que funcione el nuevo sistema de información de laboratorios (LIS).

Análisis de documentos: Permitió determinar los problemas que presenta el sistema que usan hoy para la realización del trabajo de laboratorios, analizando la forma en que se recopila, almacena y procesa la información, y además las mejoras que se le pueden hacer al mismo y se observo el posible futuro comportamiento de los planes de proyecto.

✡ *Métodos Empíricos*

Análisis y síntesis: al utilizar este método se pudo realizar el procesado de la información recopilada por los métodos empíricos.

Históricos lógicos: facilitan la posibilidad de analizar y sacar mayor cantidad de conocimientos agregados de los sistemas anteriores como el GalenLab.

Enfoque sistémico: Se ha tomado el problema central y se ha fraccionado en sub-problemas y al final se procedió a la integración de estos sub-problemas dándole una calidad mayor al proceso debido a un mayor acabado de las particularidades.

Metodología informática

Metodología de desarrollo de software RUP: El Proceso Unificado es un proceso de desarrollo de software (conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software). Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

El Proceso Unificado está basado en componentes. Utiliza el lenguaje unificado de modelado (UML) para preparar todos los esquemas de un sistema de software. Los verdaderos aspectos definitorios del proceso unificado se resumen en tres fases claves: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

Es además una forma disciplinada de asignar tareas y responsabilidades en un grupo de desarrollo de software planteando quién hace, qué, cuándo y cómo. Como metodología que es nos ayuda en el proceso de desarrollo del software, siguiendo las fases en orden de una manera iterativa se puede obtener un producto con la calidad requerida.

Estructuración del contenido:

El contenido de este Trabajo se encuentra dividido en tres capítulos fundamentales. En el capitulo inicial se hace un estudio del arte en general, se analizan diversos sistemas que se encuentran operando en el sector. Se estudian las metodologías, técnicas y tendencias mas recientes con el objetivo de acrecentar el conocimiento acerca de las alternativas posibles y herramientas que se van a utilizar en el entorno de desarrollo proyecto.

El segundo capitulo está centrado en la descripción y análisis del sistema propuesto, donde se tocan temas como la Integración con otros módulos y sub-sistemas, la arquitectura de la base de datos y los diferentes recursos empleados en la optimización del código del lenguaje del consultas. Se presenta el diseño de la BD, describiendo cada una de sus tablas y atributos.

El tercer capitulo esta encaminado más bien a hacer una validación teórica del diseño, en éste se manejan los temas vinculados a la integridad, normalización y control de redundancia de la información. Se tocan además los temas de seguridad y trazabilidad de las acciones. Al finalizar éste capítulo se hace una valoración general de los resultados obtenidos y se recomiendan diferentes alternativas que mejorarán el sistema en iteraciones futuras.

En presente capitulo se aborda un análisis profundo acerca de **LIS** como sistema automatizado. Se hará alusión a sistemas homólogos, siendo algunos de ellos grandes exponentes del género en el mercado mundial. Se observarán sus principales ventajas y características, gracias a las cuales ostentan hoy en día su elevada distinción, de esta manera se puede establecer un nivel comparativo dada similitud con muchas de las metas de estos sistemas.

Dentro del estudio está como objetivo fundamental, indagar acerca de las principales tecnologías que dan soporte a este tipo de *softwares*. Se Observarán las tendencias, técnicas y metodologías mas aplicadas en la actualidad, y por esta vía poder establecer lineamientos a la hora de seleccionar un entorno de desarrollo adecuado. Incluido en esta pesquisa se encuentra una valoración de estado del arte en la universidad.

En el transcurso del capitulo, se citan algunos conceptos técnicos, y se proporcionará una fundamentación de las principales herramientas y lenguajes a los que se recurren para dar un óptimo desarrollo al proyecto.

1.1 Sistema de Información de Laboratorios. LIS como concepto

El Sistema para Laboratorios es el resultado de años de experiencia de tecnólogos médicos, esta experiencia nos ha permitido diseñar un sistema homogéneo que cumple con todos los requisitos principales de los laboratorios, esto adquirido en adición, de las recomendaciones y comentarios de laboratorios privados y públicos, que han aportado su peritaje y necesidades técnicas. De esta manera, cada nueva versión del sistema incorpora funcionalidades, ofrecimientos y cambios sugeridos o recomendados por los propios usuarios.

Es un sistema de información y manejo administrativo para Laboratorios Clínicos, de fácil manejo y muy amigable para el usuario final. Cumple con todas las necesidades de la vida diaria de los Laboratorios Clínicos, pues ha sido diseñado luego de un estudio de las necesidades de este campo.

El sistema agiliza el seguimiento y análisis de resultados, por parte del personal de laboratorio, siempre cumpliendo con los requisitos de seguridad y confidencialidad. Es un sistema que se diseña para convertir datos de naturaleza compleja en información de valor, proveyéndoles a los usuarios el conocimiento necesario para evaluar eficientemente las opciones de tratamiento del paciente.

Tiene la capacidad de presentar la información en informes diseñados por el usuario, y mediante consultas directas a la base de datos, donde residen los datos clínicos.

1.1.1 Funcionalidades básicas de un Sistema de laboratorios

- Entrada de Datos Automatizadas.
- Transmisión Automatizada de Resultados vía interfaces, desde el equipo de laboratorio hacia el Servidor Central.
- Tan pronto como los resultados son transferidos y sean validados por el tecnólogo, los mismos están disponibles para todo el personal autorizado (médicos, enfermeras, tecnólogos, etc.).
- Una vez el tecnólogo o médico realiza la prueba en el instrumento y verifica los resultados, el sistema permite actualizar el Expediente Médico Electrónico del paciente.
- Recibo de muestras.
- Manejo Clínico y Administrativo.
- Cantidad ilimitada de Solicitudes y Muestras Procesadas.
- Múltiples listados internos para asegurar el control de la información dentro del Laboratorio
- Resumen del final del día, con amplia información científica, estadística administrativa y contable.
- Interactúa con los demás módulos del SIH.

1.1.2 Funcionalidades de control de Calidad

- Definir su propio Catálogo de Pruebas organizadas en Áreas y Perfiles (Rutinas), incluyendo entre otros: química sanguínea, hematología, hormonas, marcadores tumorales, inmunología[15], microbiología, citologías, funcionalismo renal, gases, marcadores hepáticos,...
- Registrar las Solicitudes de Exámenes en el momento de recibirlas, imprimir Reportes, Facturas y Hojas de Trabajo.
- Localizar cualquier Solicitud de exámenes al instante. Registrar o verificar resultados usando búsquedas para cada Perfil. Imprimir Resultados total o parcialmente, con posibilidad de definir el orden de los Perfiles y de las Pruebas.
- Llevar el control de las Solicitudes de Exámenes y de los Resultados. (Solicitudes recibidas, anuladas, pendientes de resultados, por entregar, canceladas...).
- Conservar la historia de resultados de cada Paciente con el fin de obtener estadísticas y resúmenes.
- Definir Usuarios y permisos de acceso. Seguridad integrada con el manejador de bases de datos.
- Obtener informes, auxiliares y estadísticas de utilidad comprobada para la administración.

En resumen

Se trata de una solución modular, proyectada específicamente para adaptarse con facilidad a los diversos modelos de gestión utilizados por las más modernas organizaciones sanitarias. LIS permite modelar y gestionar procesos de trabajo de alta complejidad, facilitando los servicios de diagnóstico de laboratorio, constituyendo una herramienta robusta y flexible para optimizar sus procesos.

1.2 Estado del arte a nivel internacional

A continuación, se explican algunos sistemas, que se han desarrollado con la finalidad de llevar a cabo la automatización de los Laboratorios clínicos en Cuba y el mundo. Existen muchos otros ejemplos que son buenos exponentes de la industria del *software* médico, no obstante esta pequeña muestra ha sido seleccionada por su semejanza en muchos o algunos de los aspectos al Sistema que se espera lograr con el desarrollo de este trabajo.

🗁*ALFA21®*

La División Informática de Salamanca Laboratorio Clínico ostenta **ALFA21®**, un avanzado sistema de Gestión de Laboratorios de Análisis Clínicos que operan tanto en el sector público como en el privado. Realizado sobre Windows con las herramientas de programación más actuales, en completa compatibilidad con su entorno.

Alfa21 posee una arquitectura abierta que se adapta permanentemente a la innovación tecnológica. Su estructura modular le permite crecer al ritmo de sus necesidades, desde una sencilla aplicación mono-puesto en Windows95/98/Me/2000/XP hasta una compleja red sobre Windows NT/2000/2003 Server.

La aplicación se encuentra preparada para trabajar con bases de datos Access y Cliente-Servidor SQL SERVER. Asimismo, el sistema está optimizado para el trabajo en entornos Terminal Server, Citrix, etc., lo que permite la interconexión entre laboratorios, independientemente de que se encuentren situados en diferentes ubicaciones físicas.

Ha sido diseñado para evitar la saturación de la base de datos, lo que agiliza el trabajo diario y disminuye los tiempos de espera. La aplicación le garantiza la integridad de los datos almacenados y su transportabilidad a nuevos sistemas si su laboratorio así lo requiere. También es posible la exportación de sus datos a cualquier aplicación compatible y la utilización de los más variados periféricos.

Cumple los principios de seguridad de la Ley Orgánica 15/1999, de 13/XII, de Protección de Datos de Carácter Personal, y el R.D. 994/1999, de 11/VI, por el que se aprueba el Reglamento de Medidas de Seguridad de los ficheros automatizados que contengan datos de carácter personal.

Cada usuario posee un nombre y una contraseña propios que lo identifican y determinan los módulos del programa a los que tiene acceso y además, personaliza y limita las acciones que cada usuario puede realizar en la aplicación. [1].

🗁 *CMLab MCS*

Software diseñado para automatizar la gestión administrativa y apoyar las operaciones del Laboratorio Clínico / Bioquímico.

Se destaca por su estabilidad, facilidad de uso, adaptabilidad y excelente desempeño, que lo convierten en un poderoso auxiliar del laboratorio clínico, de microbiología o bacteriología, y de patología.

CMLab MCS es reconocida como una aplicación de excelente calidad que proporciona la solución total para los requerimientos de información del laboratorio clínico/bioquímico. Este éxito es resultado de un sólido conocimiento de las necesidades de sus usuarios, y de una política de atención personalizada.

Es una aplicación para Windows que está diseñada y desarrollada sobre el modelo cliente/servidor. Proporciona un rendimiento excelente en redes multiusuario con Windows 2000 Server, y SQL Server 2000. Asimismo, puede ser usada en ambientes mono-usuario con Windows 2000, o Windows XP.

El motor de bases de datos cliente/servidor usado en CMLab MCS permitirá que el Laboratorio almacene información durante años, sin necesidad de recurrir a actividades de desincorporación, cortes o resúmenes; asegurando no obstante, la integridad y disponibilidad de los registros. [2].

🗁 *MediLab LIS*

MediLab LIS es un sistema de información que permite ejecutar de forma rápida, eficiente y eficaz los procesos medulares de un laboratorio médico. Desde el registro del paciente e identificación de la(s) muestra(s) hasta el reporte final de resultados, MediLab LIS facilita la ejecución que cada una de las actividades que se realizan en su laboratorio, permitiendo de esta forma reducir los tiempos de procesamiento, los costos de operación e incrementar la capacidad de recepción de muestras sin colapsar el laboratorio.

MediLab LIS es una solución pre-construida que posee una aplicación fácil de usar, altamente disponible y segura que permite almacenar, manipular y compartir la información.

Emplea la base de datos más rápida del mercado y está construido y diseñado sobre una plataforma tecnológica de última generación que ofrece resguardo de la seguridad y permite la escalabilidad del sistema según se requiera.

MediLab construye sus aplicaciones de forma que funcionen en cualquier plataforma tecnológica, de forma que el usuario no esté limitado a una en particular. Ha sido probado, ofreciendo alto rendimiento, confiabilidad y seguridad en los sistemas operativos más populares, incluyendo: Windows, Linux y UNIX.

Funcionalidades básicas:

- Permite el trabajo simultáneo de cualquier número de personas.
- Maneja roles de seguridad para que un usuario solo acceda a aquellas funcionalidades que le han sido asignadas.
- Definición de las pruebas y perfiles que el laboratorio realiza.
- Selección del orden en que se imprimen los resultados de los análisis realizados.
- Mantenimiento y actualización de la información de pacientes y exámenes realizados.
- Visualización de facturas y recibos generados.
- Recepción de pacientes e identificación de la(s) muestra(s).
- Generación de factura o recibo.
- Generación del cuaderno de trabajo.
- Carga y reporte de resultados.
- Entrega de resultados.
- Reporte de arqueo de caja. [3]

🗁*CONLAB LIS® - Laboratorio*

Puede integrar casi cualquier tipo de servicios, sistemas, departamentos, clínicas, procesos, datos, comunicación, etc. que existen en un hospital.

CONLAB LIS® - Laboratorio es un sistema de información y manejo administrativo para Laboratorios Clínicos, de fácil manejo y muy amigable para el usuario final debido a su sistema autodidáctico. Esta desarrollado con productos de Microsoft®, lo cual garantiza un constante mejoramiento de la

aplicación a costos razonables. Cumple con todas las necesidades de la vida diaria de los Laboratorios Clínicos de Centro, Sur América y Europa, pues ha sido diseñado luego de un estudio de las necesidades de la región.

Principales características:

- Plataforma en Visual Basic, Access, SQL Server y Windows NT/2000
- Crecimiento Ilimitado
- Ingreso de pacientes desde sitios remotos como Laboratorios satélites, y oficinas de Médicos
- Impresión de recibo de pago para el paciente y etiquetas con código de barra para identificación de las muestras.
- Hojas de flebotomía y recibo de las muestras tomadas o llegadas al Laboratorio.
- Múltiples listados de trabajo, configurables por el usuario. Hojas de trabajo por: Instrumento, sección, departamento, operario, o por prueba.
- Captura de resultados de igual forma a los listados de trabajo.
- Interfaces listas para su uso, con más de 400 equipos.
- Verificación de los resultados anormales por filtro automático.
- Por estar fuera de rango o del chequeo Delta.
- Firma electrónica de los resultados.
- Envío de resultados por fax, mail, o a estaciones remotas. [4].

1.3 Estado del arte a nivel nacional

En el país también se han desarrollado algunos sistemas con el objetivo de mejorar el trabajo en los laboratorios, de ellos el que mas sobresale a grandes rasgos es el GalenLab desarrollado por Softel en el año 2005.

🗁*GalenLab*

GALEN LAB está diseñado para ser utilizado por los técnicos, médicos, enfermeras y personal administrativo de Medios de diagnósticos independientes o pertenecientes a un centro hospitalario, que necesitan del sistema para optimizar su trabajo y elevar su eficiencia. Cuenta con un sistema de ayuda en línea que brinda al usuario toda la información que necesite sobre el proceso que está efectuando y posee un estricto control de acceso que permite a cada técnico visualizar solamente la

información y opciones del sistema relacionadas con su actividad. No permite el acceso a ninguno de los módulos e informaciones a partir de puntos no autorizados.

Está desarrollado para una plataforma Windows de 32 bits (Windows NT o superior y Windows 98 o superior) con una configuración Cliente/Servidor y el uso del gestor de bases de datos relacional SQL Server como reservorio de la información y está implementado sobre el lenguaje de POO (Programación Orientada a Objetos) como Delphi.

El personal que usa GALEN LAB, mantiene una relación continua y estable con el sistema y no requiere de conocimientos profundos del software de soporte para trabajar con el mismo. Es un personal con un nivel de instrucción informático aceptable que le permitirá una rápida asimilación del Sistema, el cual cuenta con un manual muy bien documentado que describe las características del su uso y está dividido en seis capítulos: *Generales, Requerimientos e instalación, Configuración, Solicitudes*, *Resultados y Autoanalizadores.* Los cuatro últimos se corresponden con los cuatro módulos funcionales que conforman el sistema. [5].

1.4 Análisis de tecnologías y tendencias a nivel internacional

🗁*1.4.1 Herramientas CASE*

Las Herramientas CASE (***C**omputer **A**ided **S**oftware **E**ngineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. [6].

🗁*1.4.1.1 Porque usar una herramienta CASE?*

Cuando se hace la planificación de la base de datos, la primera etapa del ciclo de vida de las aplicaciones de bases de datos, también se puede escoger una herramienta CASE (*Computer-Aided Software Engineering*) que permita llevar a cabo el resto de tareas del modo más eficiente y efectivo posible. Una herramienta CASE suele incluir:

- Un diccionario de datos para almacenar información sobre los datos de la aplicación de bases de datos.

- Herramientas de diseño para dar apoyo al análisis de datos.
- Herramientas que permitan desarrollar el modelo de datos corporativo, así como los esquemas conceptual y lógico.
- Herramientas para desarrollar los prototipos de las aplicaciones.

El uso de las herramientas CASE puede mejorar la productividad en el desarrollo de una aplicación de bases de datos. Y por productividad se entiende tanto la eficiencia en el desarrollo, como la efectividad del sistema desarrollado. La eficiencia se refiere al coste, tanto en tiempo como en dinero, de desarrollar la aplicación. La efectividad se refiere al grado en que el sistema satisface las necesidades de los usuarios. Para obtener una buena productividad, subir el nivel de efectividad puede ser más importante que aumentar la eficiencia. [7].

1.4.1.3 Rational Rose

Existen herramientas Case de trabajo visuales como Rational Rose, que permiten realizar el modelado del desarrollo de los proyectos, en la actualidad la mejor y más utilizada en el mercado mundial es Rational Rose.

Es una herramienta para el Modelado Visual mediante UML de sistemas *software*, con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de *software* (UML), o sea que permite a todo el equipo de desarrollo comunicarse con un lenguaje y una herramienta.

Rational Rose le permite visualizar, entender, y refinar sus requerimientos y arquitectura antes de enfrentarse al código lo que permite evitar esfuerzos desperdiciados en el ciclo de desarrollo.

Posibilita completar una gran parte de las disciplinas (flujos fundamentales) del proceso unificado de Rational (RUP):

- Modelado del negocio
- Captura de requisitos
- Análisis y diseño
- Implementación
- Control de cambios y gestión de configuración.

Rational también ofrece:

- Diseño dirigido por modelos que redundan en una mayor productividad de los desarrolladores, admitiendo UML, COM y Booch.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Soporte OLE.
- Disponibilidad en múltiples plataformas. [8]

🗁*1.4.1.4 Case Studio*

CASE Studio es una herramienta profesional de potente utilidad para el diseño de Bases de Datos. Facilita herramientas para la creación de diagramas de relación, modelado de datos y gestión de estructuras. Tiene soporte para trabajar con una amplia variedad de formatos de base de datos (Oracle, SQL, MySQL, PostgreSQL, Access, etc.) y permite además generar scripts SQL, aplicar procesos de retro-ingeniería (*ingeniería inversa*) a las Bases de Datos, usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF.

A través de los diagramas de relación que proporciona se puede tener una visión más clara del contenido y estructura de la base de datos, facilitando la gestión y mantenimiento de la misma. [9]

1.5 Modelo de Entidad Relación (MER)

🗁*1.5.1 Que es un MER?*

Los diagramas o modelos entidad-relación (a veces denominado por su siglas, *E-R* "*Entity relationship*") son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976. El modelo entidad-relación está formado por un

conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

En la actualidad es el modelo que mas se usa para modelar problemas reales y administrar datos dinámicamente. Cuando fue creado a finales de los años setenta, no tardó en consolidarse como un nuevo paradigma en los modelos de bases de datos. Su idea fundamental se basa en el concepto de *tablas*, que a su vez se componen de *registros* (las filas de una tabla) y *campos* (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia. Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado *modelo entidad-relación extendido*. [10] y [11].

1.6 Base de datos

🗁*1.6.1 Que es una BD?*

Base de datos es un conjunto de información almacenada en memoria auxiliar, que permite acceso directo y un conjunto de programas que manipulan esos datos. Es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina, accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo. [12].

🗁*1.6.2 Metodología de diseño de BD*

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en sub-problemas y se resuelve cada uno de estos sub-problemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Un *esquema conceptual* es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para

manipularla. Un *modelo conceptual* es un lenguaje que se utiliza para describir esquemas conceptuales.

El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un *esquema lógico* es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un *modelo lógico* es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un *esquema físico* es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos. [13].

🗁 *1.6.3 Tipos de BD*

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:

Las bases de datos se pueden dividir en cuatro tipos básicos:

1. Bases de datos de fichero plano (o ficheros por bloques).
2. Bases de datos relacionales.
3. Bases de datos orientadas a objetos.
4. Bases de datos híbridas.

🗁 *Base de datos de fichero*

Las bases de datos de fichero plano consisten en ficheros de texto divididos en filas y columnas. Estas bases de datos son las más primitivas y quizás ni tan siquiera merezcan considerarse como tales. Pueden ser útiles para aplicaciones muy simples, pero no para aplicaciones medianas o complejas, debido a sus grandes limitaciones.

Base de datos relacional

Las bases de datos relacionales son las más populares actualmente. Su nombre proviene de su gran ventaja sobre las bases de datos de fichero plano: la posibilidad de relacionar varias tablas de datos entre sí, compartiendo información y evitando la duplicidad y los problemas que ello conlleva (espacio de almacenamiento y redundancia).

Existen numerosas bases de datos relacionales para distintas plataformas (Access, Paradox, Oracle, Sybase) y son ampliamente utilizadas. Sin embargo, tienen un punto débil: la mayoría de ellas no admite la incorporación de objetos multimedia tales como sonidos, imágenes o animaciones.

Base de datos orientada a objeto

Las bases de datos orientadas a objetos incorporan el paradigma de la Orientación a Objetos (OO) a las bases de datos. La base de datos está constituida por objetos, que pueden ser de muy diversos tipos, y sobre los cuales se encuentran definidas unas operaciones. Las bases de datos orientadas a objetos pueden manejar información binaria (como objetos multimedia) de una forma eficiente. Su limitación suele residir en su especialización, ya que suelen estar diseñadas para un tipo particular de objetos (por ejemplo, una base de datos para un programa de CAD).

Base de datos híbrida

Las bases de datos híbridas combinan características de las bases de datos relacionales y las bases de datos orientadas a objetos. Manejan datos textuales y datos binarios, a los cuales se extienden las posibilidades de consulta. Es una tecnología reciente y aún existen pocas en el mercado. [14].

1.7 SGBD

1.7.1 Que es un SGBD?

Los sistemas de bases de datos surgieron con el objetivo de resolver los problemas que planteaban los sistemas de ficheros. El SGBD (Sistema Gestor de base de datos) o DBMS (*Data-Base Management System*) es un conjunto de programas que permiten a los usuarios definir, crear y mantener la base de datos, además de proporcionar un acceso controlado a la misma. [15]

Todos los accesos a la base de datos se realizan a través del SGBD, el que proporciona un lenguaje de definición de datos que permite a los usuarios definir la base de datos como tal, y un lenguaje de manejo de datos que permite a los usuarios la inserción, actualización, eliminación y consulta de datos.

Proporciona además acceso controlado, seguridad, integridad, concurrencia y controla la recuperación ante fallos; además de proporcionar un mecanismo de vistas que permite mostrar a los usuarios sólo aquellos datos que les interesan.

El SGBD se compone de un lenguaje de definición de datos DDL (*Data Definition Language*), de un lenguaje de manipulación de datos DML (*Data Manipulation Language*) y de un lenguaje de consulta SQL (*Structured Query Language*). [16].

1.7.2 Tipos de SGBD

La arquitectura de un sistema de base de datos está influenciada por el sistema informático que soporta la instalación del SGBD, lo que reflejará muchas de las características propias del sistema subyacente en el SGBD.

Arquitectura

1.7.2.1 SGBD Centralizado

Un sistema de base de datos centralizado es aquel que se ejecuta en un único sistema computacional sin tener, para tal efecto, que interactuar con otros computadores. El rango de estos sistemas comprende desde los sistemas de bases de datos mono-usuario ejecutándose en computadores personales hasta los sistemas de bases de datos ejecutándose en sistemas de alto rendimiento.

Normalmente los sistemas de base de datos mono-usuarios no suelen proporcionar muchas de las facilidades que ofrecen los sistemas multiusuario, en particular no tienen control de concurrencia y tienen precarios o inexistentes sistemas de recuperación.

1.7.2.2 SGBD Cliente servidor

Con el crecimiento de los computadores personales (PC) y de las redes de área local (LAN), se han ido desplazando hacia el lado del cliente la funcionalidad de la parte visible al usuario de la base de datos (interfaces de formularios, gestión de informes, etc.) de modo que los sistemas servidores provean la parte subyacente que tiene que ver con el acceso a las estructuras de datos, evaluación y procesamiento de consultas, control de concurrencia y recuperación.

Los sistemas servidores pueden dividirse en 2 tipos: los servidores transaccionales (que sirven para agrupar la lógica del negocio en un servicio aparte, proveen una interfaz a través de la cual los clientes pueden enviar peticiones como lo son ODBC o RPC) y los servidores de datos(los cuales envían datos

a más bajo nivel y que descansan en la capacidad de procesamiento de datos de las maquinas clientes).

Existen 2 arquitecturas dominantes en la construcción de motores de base de datos cliente-servidor: los motores multiprocesos y los motores multi-hilos.

- ***Motores e BD multiprocesos (Multi-process Database Engines)***

  Algunos motores de base de datos confían en múltiples aplicaciones para realizar su trabajo. En este tipo de arquitectura, cada vez que un usuario se conecta a la base de datos, ésta inicia una nueva instancia de la aplicación de base de datos. Con el fin de coordinar a muchos usuarios que accesan los mismos conjuntos de datos estos ejecutables trabajan con un coordinador global de tareas que planifica operaciones para todos los usuarios.

  La comunicación entre aplicaciones de este tipo se realiza por medio de un sistema propietario de comunicaciones interprocesos.

- ***Motores de base de datos multi-hilos (Single-Process Multi-threaded Database Engines)***

  Los motores de base de datos multi-hilos abordan el problema del acceso multiusuario de una manera distinta, pero con principios similares. En lugar de confiar en que el sistema operativo comparta los recursos de procesamiento, el motor toma la responsabilidad por sí mismo, lo que en la práctica se asocia a una mejor portabilidad del sistema. Motores de base de datos comerciales como Sybase Adaptive Server o *Microsoft SQL Server* son ejemplos de este enfoque.

  Las ventajas de este tipo de motores radican en una mayor eficiencia en el uso de recursos para determinadas plataformas. Mientas un sistema multiprocesos consume entre 500 Kb y 1 Mb por conexión, un motor multi-hilos consume entre 50 y 100 Kb de RAM diferencia que puede ser utilizada en caché de datos y procedimientos. Otra ventaja es que no hay necesidad de un mecanismo de comunicación interprocesos.

1.7.2.3 SGBD Cliente servidor

Los sistemas paralelos de base de datos constan de varios procesadores y varios discos conectados a través de una red de interconexión de alta velocidad. Para medir el rendimiento de los sistemas de

base de datos existen 2 medidas principales: la primera es la productividad (*throughput*) que se entiende como el número de tareas que pueden completarse en un intervalo de tiempo determinado.
La segunda es el tiempo de respuesta (*response time*) que es la cantidad de tiempo que necesita para completar una única tarea a partir del momento en que se envíe. Un sistema que procese un gran número de pequeñas transacciones puede mejorar su productividad realizando muchas transacciones en paralelo. Un sistema que procese transacciones más largas puede mejorar tanto su productividad como sus tiempos de respuesta realizando en paralelo cada una de las sub-tareas de cada transacción.
Las ganancias en este tipo de SGBD se pueden dar en términos de velocidad (menor tiempo de ejecución para una tarea dada) y *ampliabilidad* (capacidad de procesar tareas más largas en el mismo tiempo).

1.7.2.4 SGBD Distribuido

En un SGBD distribuido, la base de datos se almacena en varios computadores que se pueden comunicar a sus vez por distintos medios de comunicación (desde redes de alta velocidad a líneas telefónicas). No comparten memoria ni discos y sus tamaños pueden variar tanto como sus funciones pudiendo abarcar desde PC hasta grandes sistemas. Se denomina con el término de emplazamientos o nodos a todos aquellos computadores que pertenecen a un sistema distribuido.

Las principales diferencias entre las bases de datos paralelas y las bases de datos distribuidas son las siguientes: las bases de datos distribuidas se encuentran normalmente en varios lugares geográficos distintos, se administran de forma separada y poseen una interconexión más lenta. Otra diferencia es que en un sistema distribuido se dan dos tipos de transacciones, las locales y las globales. Una transacción local es aquella que accede a los datos del único emplazamiento en el cual se inició la transacción. Por otra parte una transacción global es aquella que o bien accede a los datos situados en un emplazamiento diferente de aquel en el que se inició la transacción, o bien accede a datos de varios emplazamientos distintos.

Un sistema de base de datos distribuido se conoce por:

- Los distintos emplazamientos están informados de los demás.
- Aunque algunas relaciones pueden estar almacenadas sólo en algunos emplazamientos, éstos comparten un esquema global común.

- Cada emplazamiento proporciona un entorno para la ejecución de transacciones tanto locales como globales.[17]

1.7.3 SGBD más usados a nivel mundial

SGDB Libres:

- **PostgreSQL**
- **MySQL**
- Firebird versión 6 de Interbase.
- LICENSE versión 1.0.
- SQLite Licencia Dominio Público
- Sybase ASE Express Edition para Linux

SGBD Propietarios o comerciales:

- dBase
- FileMaker
- FoxPro
- **IBM Informix**
- Universal Database (DB2 UDB)
- MAGIC
- Microsoft Access
- **Microsoft SQL Server**
- **Oracle**
- Open Access
- **Paradox**
- PervasiveSQL
- Postgress(DBMS)
- **Sybase ASA**
- Sybase IQ
- WindowBase

1.7.3 SGBD Analizados. MySQL y PostgreSQL

Se analizaron las dos alternativas Open Source mas difundidas en el mundo, con el objetivo de establecer comparaciones en cuanto al nivel de prestaciones, utilidades que ofrecen, etc.

1.7.3.2 MySQL

MySQL es un SGBD relacional, multi-hilo y multi-usuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso.

🗁 *Lenguajes de programación*

Existen varias APIs que permiten, a aplicaciones escritas en diversos lenguaje de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Eifel, Smalltalk, Java (con una implementación nativa del *driver* de Java), Lisp, Perl, PHP, Python, Ruby, REALbasic (Mac), FreeBASIC, y Tcl; cada uno de estos utiliza una API específica. También existe un interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL.

MySQL es muy utilizado en aplicaciones web como MediaWiki o Drupal, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. Es una Base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Características de la versión 5.0.22

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte a multi-plataforma.
- Procedimientos almacenados.
- Triggers.
- Vistas actualizables.
- Soporte a VARCHAR.
- INFORMATION_SCHEMA.
- Modo Strict.
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle.
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB[15] para transacciones e integridad referencial).
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos de recuperación (savepoints) con InnoDB.
- Soporte para SSL.
- Query caching (chequeo de querys).

- Sub-Select`s (o SELECTs anidados).
- Indexar y buscar campos de texto completos usando el motor de almacenamiento MyISAM.
- Soporte completo para Unicode.
- Conforme a las reglas ACID usando los motores InnoDB, BDB y Cluster.
- Uso de multihilos mediante hilos del kernel.
- Usa tablas en disco b-tree para búsquedas rápidas con compresión de índice.
- Tablas hash en memoria temporales.
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL.
- Completo soporte para operadores y funciones en cláusulas select y where.
- Completo soporte para cláusulas group by y order by, soporte de funciones de agrupación.
- Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).
- Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando *named pipes* y en sistemas Unix usando ficheros *socket* Unix.

Características distintivas

Las siguientes características son implementadas únicamente por MySQL:

- Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example en 5.x), permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

🗁 *Tipos de compilación del servidor*

Hay tres tipos de compilación del servidor MySQL:

- Estándar: Los binarios estándar de MySQL son los recomendados para la mayoría de los usuarios, e incluyen el motor de almacenamiento InnoDB.
- Max (No se trata de MaxDB, que es una cooperación con SAP): Los binarios incluyen características adicionales que no han sido lo bastante probadas o que normalmente no son necesarias.
- MySQL-Debug: Son binarios que han sido compilados con información de depuración extra. No debe ser usada en sistemas en producción porqué el código de depuración puede reducir el rendimiento.[18]

1.7.3.2 PostgreSQL

Es una mejora del sistema gestor de bases de datos POSTGRES, un prototipo de investigación de DBMS de nueva generación. PostgreSQL Mantiene los potentes modelos de datos y riqueza de tipos de POSTGRES, reemplaza el lenguaje de consulta PostQuel con un sub-juego extendido de SQL. PostgreSQL es gratis y las fuentes son accesibles públicamente. Es un potente motor de base de datos, es un servidor de base de datos relacional libre, liberado bajo la licencia BSD.

PostgreSQL es un SGBD libre (open source) de gran potencia, pudiéndose comparar con los gestores comerciales mas populares como Oracle. Está diseñado para soportar volúmenes masivos de datos, sin que ello afecte en lo absoluto en su rendimiento. Ejemplo de ello es que puede soportar tuplas de hasta 1600 campos y retornar a su vez millones de estas tuplas en apenas unos segundos.

Características

Mediante un sistema denominado MVCC (Acceso concurrente multi-versión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

*Es full **ACID** Compliant (**A**tomicity, **C**onsistency, **I**solation and **D**urability):*

- Atomicidad (Indivisible): es la propiedad que asegura que la operación se ha realizado o no, por lo tanto ante un fallo del sistema no puede quedar a medias. Esto quiere decir que las

transacciones se ejecutan completamente o se anulan, sin correr el riesgo de que un imprevisto en mitad del proceso pueda dejar el resultado en un estado no autoconsistente.

- Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto solo se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos. Ello significa que la base de datos siempre se transforma de un estado válido a otro estado válido.
- Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error. Lo que significa que los resultados de las diferentes transacciones son invisibles hasta que estén completamente acabadas.
- Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. Esto quiere decir que una vez acabadas las operaciones sobre los datos, éstos tienen que "sobrevivir" a posibles fallos en el sistema de ficheros.

Otras características:

- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Comunidades muy activas, con varias de ellas en castellano.
- Bajo “Costo de Propiedad Total” (TCO) y rápido “Retorno de la Inversión Inicial” (ROI).
- Altamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio: C, C++, Java (vía PL/Java), PHP (PL/PHP), Perl (PL/Perl), Phyton (PL/ Phyton), Ruby (PL/Ruby), Scheme (PL/Scheme) y lenguaje propio llamado PL/pgSQL* (similar al de Oracle).
- Drivers: Odbc, jdbc, .Net, etc.
- Soporte de todas las características de una base de datos profesional (triggers, Store Procedures/funciones, secuencias, vistas, vistas materializadas, etc.)
- Soporte de tipos de datos de SQL92 y SQL 99.
- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.

- Utilidades para limpieza de la base de datos (Vacum).
- Utilidades para el análisis y optimización de Query's.
- Almacenaje especial para tipos de datos grandes (TOAST).
- Varios tipos de índices.
- Clusterización de datos en base a índices (si es data estática).
- El mejor SO para correr PostgreSQL es BSD y Unix, por su sistema dinámico de I/O (mas eficiente que otros OS).

Limites:

- Máximo de base de datos: ILIMITADO.
- Máximo de tamaño de tabla: 32TB.
- Máximo de tamaño registro: 1.6TB.
- Máximo de tamaño de campo: 1GB.
- Máximo de registros por Tabla: ILIMITADO.
- Máximo de campos por tabla: 250 a 1600 (depende de los tipos de datos usados).
- Máximo de índices por tabla: ILIMITADO.
- Numero de lenguajes en los que se puede programar: Aproximadamente 10 (pl/Pgsql, pl/Java, pl/Perl, pl/Phyton, tcl, pl/Php, C, C++, Ruby, entre otros).
- Métodos de almacenamiento de índices: 4 (B-tree, R-tree, Hash y GisT).

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (*Querys*).

Las funciones permiten subir bloques de código que se ejecuten en el servidor y pueden escribirse en variedad de lenguajes. Las funciones pueden se definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (*Stored procedures*). [19].

1.8 Análisis de las Tecnologías en el entorno Universitario

Como parte del estudio de las tendencias y tecnologías se realizó una encuesta a diferentes proyectos productivos del contexto universitario -12 en total- con el objetivo de conocer acerca de los

entornos de desarrollo que estos utilizan. Dicha encuesta arrojó diferentes datos los que fueron recogidos en la siguiente tabla.

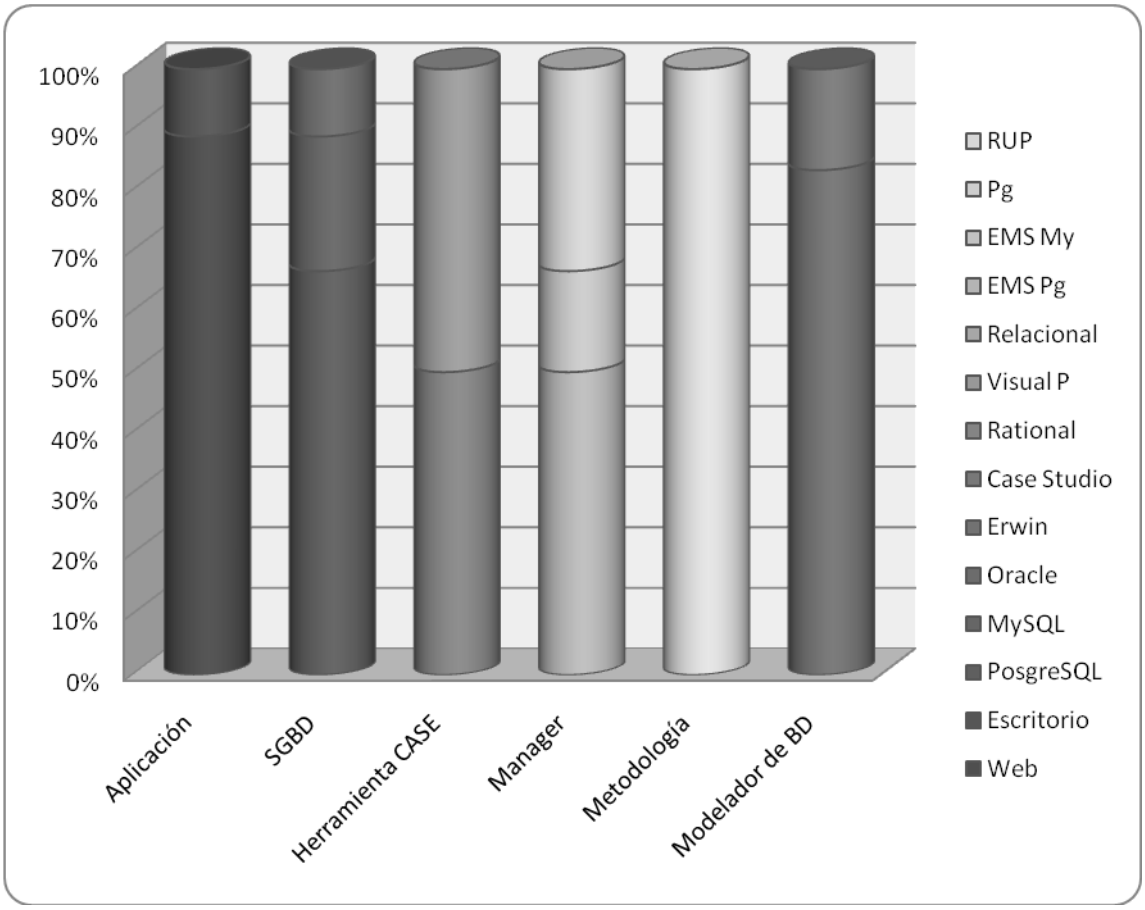


Tabla de encuestas 1

1.9 Entorno de desarrollo

Tras haber analizado cuidadosamente las metodologías, tendencias, tecnologías y herramientas candidatas y tras haber hecho un pequeño balance de las tecnologías de la periferia universitaria, se eligió el siguiente entorno de desarrollo:

🗁 *1.9.1 Herramienta CASE y metodología*

Para el desarrollo de la Ingeniería del Software, se prefirió a RUP ya que en ella reside un mayor adiestramiento por parte del equipo de desarrollo y en consecuencia de esto, se optó por el uso del Rational Rose, que es una herramienta CASE que confiere un óptimo soporte al modelado visual, ofreciendo distintas perspectivas del sistema.

Se seleccionó Rational, entre otras cosas porque presenta una plataforma independiente, que al estar ajustado al uso de un lenguaje estándar común facilita a la comunicación entre los miembros de equipo de desarrollo, y permite el monitoreo del tiempo de desarrollo del software. Además de que ofrece un diseño dirigido por modelos, lo que confiere una mayor productividad admitiendo UML, COM, entre otros y por otra parte está centrado en casos de uso y enfocado al negocio lo que genera un software de mayor calidad.

1.9.2 Gestor y modelador de la base de datos

En la selección del SGBD, PostgreSQL en su versión 8.2 fue sin dudas el más favorecido por los arquitectos y la alta dirección del proyecto, ya que ya que proporciona robustez en cuanto a la integridad y seguridad de los datos, además de que cuenta con poderosas herramientas como lo es su Control Concurrido Multi-Versión (MVCC), que permitirá el trabajo de múltiples usuarios sobre los mismos registros sin la ocurrencia de bloqueos. Otra de las ventajas que determinaron la inclinación por este gestor es el hecho de que su motor esta implementado para soportar grandes volúmenes de datos, si que esto afecte el rendimiento.

Para modelar la base de datos se escogió e CASE Studio, ya que constituye una potente herramienta que entre sus muchas facilidades nos ofrece generar scripts SQL y aplicar procesos de ingeniería inversa. Además genera detallados informes en HTML y RTF.

1.9.4 Manager

Ya que se seleccionó PostgreSQL como soporte para la base de datos, se usará el PostgreSQL Manager 2005. Esta es una herramienta de alto desempeño para la administración y desarrollo de las bases de datos de PostgreSQL. Trabaja con cualquiera de las versiones de PostgreSQL anterior a la versión 8.2. Soporta todos los últimos rasgos/características de PostgreSQL, incluido el *FILLFACTOR parameter* (factor de llenado de parámetros) en tablas e índices, la construcción de índices concurrentes, la creación de dominios basados en otros dominios y otros.

Ofrece poderosas herramientas para usuarios avanzados como el *Visual Database Designer* (Diseñador Visual de Bases de Datos), *Visual Query Builder* (Constructor Visual de Consultas), entre otras. Tiene una nueva e innovadora interfaz gráfica del usuario con un sistema bien descrito y tan claro, que ni siquiera usuarios inexpertos tendrían problemas a la hora de desarrollar. [20].

Conclusiones

En el capitulo, se apreció el sistema, desde un punto de vista conceptual mucho mas amplio, teniendo en cuenta los requisitos y funciones que implica un sistema informático LIS y las necesidades existentes en los laboratorios clínicos y las condiciones con que cuenta actualmente el país.

Se pudieron establecer pautas con respecto al conocimiento de las tendencias y tecnologías actuales a nivel mundial y en el entorno universitario. Se hizo un estudio profundo de las herramientas que concurrirán en el proyecto. Apoyándonos principalmente en el fundamento de la inclinación por un SGBD como PostgreSQL, ya que proporciona robustez en cuanto a la integridad y seguridad de los datos, Al mismo tiempo se tratan algunos conceptos técnicos asociados que son de vital importancia. Conjuntamente, se analizaron algunos de los sistemas que actualmente se encuentran liderando el mercado internacional, sus características y semejanzas que tendrá el sistema se presenta en este proyecto.

En este capitulo, hará una detallada descripción y análisis del sistema que se propone a nivel de datos. Se examinarán estrategias de integración con las otras bases de datos de los restantes módulos del SIH, indagando acerca de cómo interactúa el sistema con otros módulos y posibles subsistemas. Se hará un análisis a fondo de la arquitectura con una minuciosa descripción y fundamentación de la misma.

Posteriormente se procederá al análisis de optimización del acceso a datos, es decir, se estudiará la manera de optimizar en la medida de lo posible las Query's (consultas), procedimientos, vistas y funciones que serán manipuladas. Asimismo, se hará una selección y argumentación de los requisitos funcionales y no funcionales del sistema y se dará a conocer el modelo de objetos obtenido a partir del diagrama de clases de diseño, donde son reveladas las clases que son candidatas potenciales a convertirse en las futuras tablas que tendrá el sistema de bases de datos.

Seguidamente, se ilustrará el diseño lógico de la base de datos, haciendo uso del diagrama de entidad relación logrado a través de la herramienta seleccionada previamente para el modelamiento de la BD; mas adelante, se procederá a la descripción de tablas autónomas y las de configuración del sistema.

2.1 Estrategia de integración con otros módulos

El Sistema de Inscripción Hospitalaria o SIH en su proyección es un Sistema Informático de gran magnitud, por lo que se concibió su descomposición en diferentes módulos. Éstos se encargan del almacenamiento por separado de la información, por lo que cada uno manipula una base de datos individual, y se idearon bajo la premisa de que pudiesen operar de manera independiente en caso que se requiera, pudiendo asentar solo los módulos con los cuales se optimizarán los servicios que se prestan en el Centro Hospitalario.

Para lograr una integración de todos sus módulos y componentes, toda la gestión de información se decidió llevarla a cabo a través de la lógica del negocio. Cualquier componente, servicio web o sistema externo, hace diferentes peticiones las cuales son manejadas a través del uso de capas de abstracción. Para ello todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa.

La comunicación entre diferentes componentes se realiza mediante llamadas a servicios web o de forma directa a nivel de negocio, en caso de utilizarse servicios web, la información que es transmitida debe cumplir con los estándares internacionales que hay establecidos para así facilitar la integración entre nuevos componentes y otros sistemas hospitalarios.

La base de datos es accesada de forma directa mediante clases controladoras y los componentes reutilizados son integrados mediantes interfaces sencillas.

Este es un esquema que se decidió seguir fundamentalmente por motivos de seguridad, ya que de este modo, se garantiza que ningún subsistema haga inserciones o tenga acceso directo a la base de datos ajenas, de igual forma ocurre con sistemas externos, los cuales obtendrán la información a través de diferentes peticiones que estarán soportadas por protocolos y estándares internacionales pero que nunca “escribirán” en la BD a la que hacen dichas peticiones. Para mas información remitirse al trabajo de tesis desarrollado por los arquitectos del proyecto.

🗁*2.1.1 Esquema de interacción*

Interacción inter módulos

2.2 Descripción de la Arquitectura. Fundamentación

La arquitectura designada para la base de datos es hasta el momento Centralizada, por lo que toda la información estará congregada en un solo servidor y es controlada a través de un solo equipo informático u ordenador. Se utilizará un sistema centralizado donde hay un único servidor que dirige todo desde un solo sitio y al cual todos hacen pedidos de información, por lo que el repositorio y el

procesamiento de los datos se encuentra en un lugar fijo, y es accedido desde terminales o estaciones de trabajo que no hacen procesamiento, solamente realizan peticiones, búsquedas, consultas, adiciones, actualizaciones o eliminaciones de datos.

Es importante señalar que dentro de la centralización, la base de datos se encuentra modularizada en su contexto ya que como se ha explicado previamente, la información se encuentra físicamente en un solo lugar, pero internamente mantiene una estructura fragmentada en varias porciones (las bases de datos de los restantes módulos del proyecto) lógicamente bien definidas.

Para próximas y tempranas iteraciones del proyecto se prevé una arquitectura Distribuida. Ya que permitiría que la información no esté almacenada en su totalidad en un solo lugar físico sino que se distribuye a lo largo de una red de computadoras geográficamente separadas (partiendo de que los laboratorios ya se encuentran distribuidos de forma lógica en el territorio nacional); donde cada locación (nodo) constituye un sistema de base de datos por si mismo, pero que a su vez estos nodos se pondrán de acuerdo para el trabajo cooperativo. De esta forma la información radicará donde es generada y donde realmente se necesita. [21] (***ver anexo 1***)

2.3 Análisis de optimización de Querys

En bases de datos relacionales el lenguaje de consultas SQL es lo más utilizado para obtener información desde la base de datos. La complejidad que pueden alcanzar algunas consultas puede ser tal, que el diseño de una consulta puede tomar un tiempo considerable, obteniendo no siempre una respuesta óptima.

En SQL una consulta puede expresarse de distintas maneras, todas ellas diferentes. Cada una de las formas sugiere una estrategia para encontrar la respuesta y por lo tanto algunas pueden ser más óptimas que otras.

El problema que se plantea entonces es, el encontrar la manera más apropiada de resolver la consulta, sin que, el proceso de determinar este resultado exceda un tiempo razonable y un gasto de recursos adicional. El objetivo principal de este proceso es, encontrar o bien la mejor alternativa para solucionar la consulta, o de lo contrario, la alternativa que mejor cumpla las características de eficiencia, entre las estudiadas, cuando no sea posible estudiarlas todas.

Este proceso tiene que ser lo bastante rápido como para que sea conveniente para el motor efectuar este cálculo sin afectar el rendimiento en la construcción del resultado de la consulta. Se entiende

entonces como proceso de optimización de consultas al conjunto de técnicas, algoritmos y reglas que le permiten, al motor de base de datos, elegir una alternativa entre varias con la cual pueda "rescatar"los datos de la manera más óptima. [22].

El procesamiento de consultas tiene varias etapas a seguir para resolver una consulta SQL, las características del modelo relacional permiten que cada motor de base de datos elija su propia representación que, comúnmente, resulta ser el álgebra relacional.

Existen distintos métodos para optimizar consultas relacionales, sin embargo el enfoque de optimización basada en costos combinado con heurísticas que permitan reducir el espacio de búsqueda de la solución es el método mayormente utilizado por los motores de base de datos relaciones de la actualidad, en todo caso, independiente del método elegido para optimizar la consulta, la salida de este proceso debe ser un plan de ejecución, el cual comúnmente es representado en su forma de árbol relacional.

Sin embargo el optimizador no siempre escoge el plan más óptimo, ya que una búsqueda exhaustiva de la estrategia óptima puede consumir demasiado tiempo de proceso. Se dice entonces que el optimizador escoge una estrategia "razonablemente eficiente".

Una de las ventajas que se logra comprendiendo el proceso de optimización de consultas es la de entender que es lo que hace realmente un optimizador y con esto evitar la construcción de malas consultas. [23].

2.3.1 Aspectos a tener en cuenta

Uno de los primeros puntos a tener en cuenta fue el realizar un cuidadoso diseño, que fuese flexible y ajustable a las necesidades dominantes del sistema y evitar por todos los medios tener que añadir campos/columnas a las tablas, para de esta forma no tener que reprogramar toda la entrada de datos nuevamente. Por lo tanto, se procuró por todos los medios posibles crear un sistema funcional que pueda crecer y adaptarse fácilmente a nuevos requisitos.

Se normalizaron las tablas hasta llevarlas a una tercera forma normal (ver capitulo 3, epígrafe 3.3) y se evitó la sobreutilización del JOIN, ya que un uso excesivo e indiscriminado del mismo pudiera traer como consecuencia demora de ejecución de la consulta a la hora de devolver los datos, alrededor de 10 vinculaciones dentro de una misma consulta, pueden acarrear inconsistencia.

2.3.2 Análisis de sentencias

Para los procedimientos/funciones que hacen extracciones de datos (select), se hiso una optimización a partir del filtrado de cada uno los datos que contiene la tabla de la que se extrae información. Esto se logra pasando cada atributo en el argumento de la función y mas tarde filtrando la búsqueda con una sentencia *where*, de esta forma cada vez que se hace una búsqueda, esta pude ejecutarse desde el negocio de forma dinámica, pudiendo así filtrar la búsqueda por uno, varios o por ningún parámetro en específico. Con esto no solo se logró que se extrajeran únicamente los datos que verdaderamente se necesitan, sino que se evitó el uso del SELECT (*).

A continuación se muestra un ejemplo abstracto de una sección de código SQL que se utilizó para muchas de las funciones SELECT y UPDATE.

```
FUNCTION Nombre (p_atributo1 Tipo, p_atributo2 Tipo,…)
        SELECT atributo1, atributo2,…  FROM Tabla
        WHERE
                (atributo1 = p_atributo1 OR p_atributo1 IS NULL)…
```

Al igual que con los procedimientos de selección de datos, se optimizaron algunos de los procedimientos que actualizan nuevos registros en la base de datos, haciéndolo de forma dinámica, de manera que la sentencia *update* actualice en las tuplas solo los campos argumento con un valor distinto de nulo. De esta forma se evitó tener que actualizar la tupla completa restringiendo aún más la actualización

Ejemplo de sección de código SQL:

```
FUNCTION Nombre (p_atributo1 Tipo, p_atributo2 Tipo,…)
        UPDATE FROM Tabla
       SET
                atributo1 = p_atributo1
        WHERE
                (atributo1 = p_atributo1 AND p_atributo1 IS NOT NULL)…
```

Nomenclatura	
Nombre	*nombre de la función*
Tabla	*nombre de la tabla*
p_atributo1, p_atributo2	*nombre de los atributos que son argumentos de la función*
Tipo	*tipo de atributo (Varchar, Integer, etc.)*
Atributo	*campo de la tabla*

2.3.3 Índices

Los índices son estructuras de acceso que se utilizan para acelerar el acceso a los registros en respuesta a ciertas condiciones de búsqueda. Algunos tipos de índices, los denominados caminos de acceso secundario, no afectan al emplazamiento físico de los registros en el disco y lo que hacen es proporcionar caminos de acceso alternativos para encontrar los registros de modo eficiente basándose en los campos de indexación.

Los tipos de índices que más se utilizan son los que se basan en ficheros ordenados (índices de un solo nivel) y las estructuras en forma de árbol (índices multinivel, árboles B y árboles B+). [24].

Del empleo inteligente de los índices se obtuvieron muchas ventajas y la primera de ellas fue que se agilizaron mucho las consultas que incluyen condiciones (WHERE, LIKE) de esta forma se que evitó que el gestor haga un recorrido secuencial de la tabla completa (*full scan*). Además, al no tener que recorrer toda la tabla se evita la sobrecarga de la CPU de la máquina, garantizando así el rendimiento del servidor, pudiendo aprovechar la ganancia de tiempo en realizar otras funciones y procesamientos de la información.

También se tuvo en cuenta no hacer un uso desmedido (a lo que anteriormente se refería el término inteligente), ya que el uso excesivo de los índices trae como consecuencia que se ralentizan tanto los INSERT como los UPDATES, porque el gestor debe actualizar los mismos cada vez que se añade o modifica un registro. Por otra parte se tiene que ocupan un gran espacio en disco, ya que es información adicional la que esta guardando de cada fila. Después de hacer un estudio previo de las consultas mas usadas y de mayor envergadura, fueron eliminados todos aquellos índices que en realidad no hicieron falta.

Cabe señalar con respecto a esto que en la base de datos de el proyecto sólo se usaron los índices en algunas de las tablas llamadas autónomas del sistema. Esquema que se decidió llevar por varias razones, una de ellas es que estas tablas son las que van a almacenar la gran mayoría de los datos, por lo que tienen un peso preponderante en el volumen de información masivo que se almacenará en un futuro. La otra razón importante en este sentido es que en tablas pequeñas no merece la pena utilizar índices, ya que la ganancia será inapreciable. [25].

2.3.4 Query Planner

Se utilizó la función EXPLAIN ANALYZE para ver la cantidad de registros devueltos por cada consulta, así como los tiempos de retorno (incluido el "costo" por cada sub-consulta anidada). A partir de estos valores de retorno, se pudo comparar y modificar las sentencias del código para hacerlas más eficientes y con un mejor desempeño y así ver de las "*n*" respuestas, cual era la más óptima. Está es una de las ventajas que otorga el Planner de PostgreSQL. [26].

Ejemplo:

```
EXPLAIN ANALYZE Select id_solicitud FROM lis_solicitud
```

2.3.5 Uso del comando NEXT

Mediante ciclos utilizados en PostgreSQL, se logró una mayor consistencia de los datos ya que al ser una base datos muy grande en registros almacenados, la base datos puede colapsar a la hora de devolver los datos ya que muchos gestores de base datos devuelven los registros al instante pero en PostgreSQL, con el comando NEXT va devolviendo cada registro uno por uno y evita que haya fallos a la hora de devolver los datos.

Ejemplo de sección de código SQL:

```
DECLARE NewRecordSet RECORD;
BEGIN

  FOR NewRecordSet IN
    SELECT id_solicitud
    FROM lis_solicitud
    WHERE id_solicitud = id_s OR id_solicitud IS NULL
    LOOP
    RETURN NEXT NewRecordSet;
    END LOOP;
END;
```

2.4 Selección y argumentación de los requisitos del sistema

2.4.1 Requisitos Funcionales

- El sistema debe permitir recopilar los datos asociados a la Emisión de una Solicitud, debe proporcionar al medico o especialista la posibilidad de por así decirlo, de reservar un examen.
- Debe guardar los datos referentes a una nueva solicitud, como son las fechas de emisión, admisión y liberación, además de recomendaciones que se pueden hacer previamente. Para

ello debe contar con procedimientos de almacenado y funciones que garanticen la entrada de los datos en las tablas pertinentes, en este caso la tabla: lis-solicitud.

- Debe recoger todos los datos que sean referentes a las distintas muestras que se pueden tomar para un determinado análisis, siempre guardando relación entre la solicitud y los análisis pertinentes. Para ello se dispone de las tablas: lis_dato_muestra y lis_dato_almacenaje.

- Debe informar sobre las nuevas solicitudes que han arribado al centro. Contará con vistas que muestren las solicitudes donde el criterio de búsqueda sería la fecha de admisión. Para ello contará con la tabla **lis_solicitud**, que tendrá todos los datos de las Solicitudes.

- Debe poder hacer **búsquedas de Historias Clínicas** de pacientes, para ello debe disponer de procedimientos y funciones que optimicen las búsquedas a través de diferentes criterios, como son: el número de historia clínica, el nombre de un paciente, etc.

- El sistema tendrá limitaciones para algunos de los usuarios, lo cual estará condicionado por sus privilegios, por lo que debe permitir **Autentificarse**, restringiendo el acceso a personal ajeno y garantizando un acceso determinado según roles definidos, de esta forma se cumple con las directivas de seguridad que avalan la integridad y confiabilidad de los datos.

- Debe mostrar los datos referentes a cualquier resultado que haya sido emitido, en el momento que lo requiera el personal del laboratorio. Para sufragar esta necesidad el sistema debe contar con un conjunto de vistas y procedimientos que hagan la gestión de esta información.

- Debe dar la posibilidad de **modificar** cualquier **Resultado** que haya sido hecho previamente, para ello contará con procedimientos que actualicen de forma correcta los datos que han sido modificados e insertados nuevamente en la base de datos.

- El sistema debe proveer a los usuarios de de la posibilidad de **buscar** cualquier **Solicitud** procesada por el Laboratorio, para ello debe tener implementadas funciones y vistas que garanticen búsquedas a diferentes niveles y por diferentes criterios.

- Al igual que con las solicitudes, el sistema debe permitir **buscar** cualquier **Análisis** que se haya hecho en el laboratorio. Contara con procedimientos que responden a búsquedas generales y avanzadas.

2.4.1 Requisitos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Muestran las características que hacen al producto atractivo, usable, rápido o confiable. No alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importarle con que propiedades o cualidades se relacionen.

- **Requerimientos de soporte**

  Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información.

- **Requerimientos de seguridad y privacidad**

  La información debe transmitirse de manera segura, se debe garantizar la seguridad a todos los niveles (Interfaz, negocio y Acceso a datos) restringiendo las funcionalidades mediante roles de usuarios garantizando que la información sea accesible al usuario autorizado.

- **Requerimientos de confiabilidad**

  La información debe transmitir a través de canales seguros. Se debe chequear constantemente la integridad de los datos.

- **Requerimientos de interfaz interna**

  El sistema debe contar con una capa de acceso a datos y un conjunto de vistas y funciones que medien entre el servidor e bases de datos y el negocio, que favorezcan una rápida gestión de la información.

- **Requerimientos de hardware**

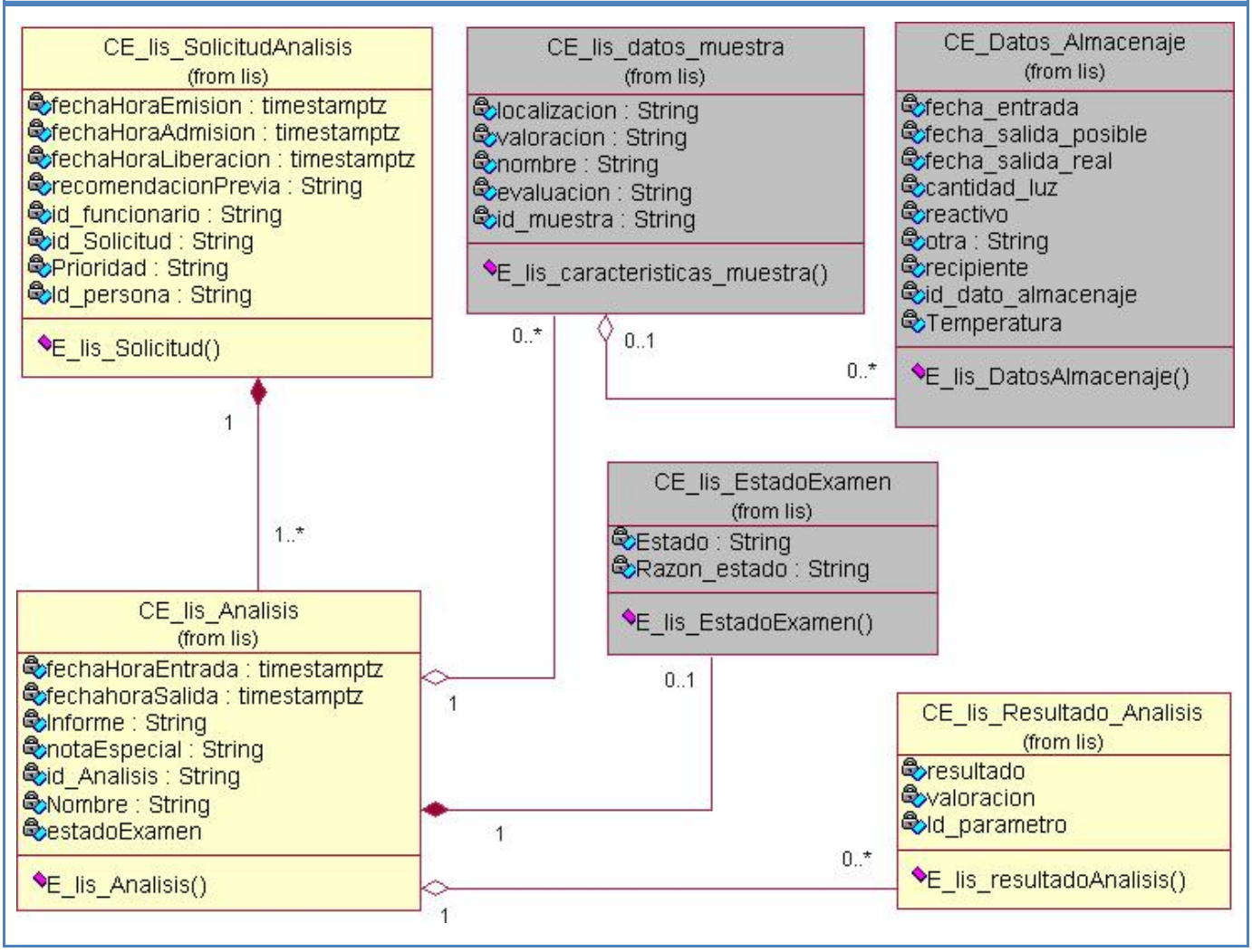
  Requerimientos para una estación de trabajo: 256Mb RAM (Recomendado 512Mb), 1GHz, 10Gb HDD.
  Requerimientos para un servidor: 512Mb RAM (Recomendado 1Gb RAM o superior), 1GHz o superior, 60Gb HDD o superior, ya que se debe lidiar con el crecimiento exponencial de la Base de datos en un tiempo relativamente corto.

- **Restricciones en el diseño y la implementación**

  Se utilizará además un grupo de bibliotecas de clases definidas, dentro de ellas se encuentran Hermes7, para la comunicación HL7 y NpgSql para la conexión al servidor de bases de datos. El servidor debe contar con el Framework 2.0. La comunicación de de las terminales clientes con el servidor será a través de conexiones de fibra óptica

2.5 Modelo de objetos del diagrama de clases persistentes

Modelo de Objetos del diagrama de clases persistentes obtenido a partir del diagrama de clases de diseño



2.6 Descripción de las clases

Nombre: CE_lis_Solicitud_Analisis	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha_hora_emision	TimeStamptz
Fecha_hora_Admision	TimeStamptz
Fecha_hora_liberacion	TimeStamptz
Recomendación_previa	Varchar
Id_persona	Varchar
Id_solicitud	Varchar
Id_funcionario	Varchar
Prioridad	Varchar
Responsabilidad # 1	
Nombre	Solicitud de análisis
Descripción	Constructor

Tabla 1 CE-LIS. Solicitud Análisis

Nombre: CE_lis_Analisis	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha_hora_entrada	TimeStamptz
Fecha_hora_salida	TimeStamptz
Informe	Varchar
Nota_especial	Varchar
Nombre	Varchar
Estado_examen	Varchar
Id_analisis	Varchar
Responsabilidad # 1	
Nombre	Análisis
Descripción	Constructor

Tabla 2 CE-LIS. Análisis

Nombre: CE_Resultado_Analisis	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_parametro	Varchar
Valoración	Varchar
Resultado	Varchar
Responsabilidad # 1	
Nombre	Resultados
Descripción	Constructor

Tabla 3 CE-LIS. Resultado Análisis

Nombre: CE_Datos_Almacenaje	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_dato_almacenaje	Varchar
Fecha_entrada	TimeStamptz
Fecha_salida_posible	TimeStamptz
Fecha_salida_real	TimeStamptz
Cantidad_luz	Integer
Reactivo	Varchar
Otra	Varchar
Recipiente	Varchar
Temperatura	Varchar
Responsabilidad # 1	
Nombre	
Descripción	

Tabla 4 CE-LIS. Datos Almacenaje

Nombre: CE_lis_Datos_Muestra	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_muestra	Varchar
Localización	Varchar
Valoración	Varchar
Nombre	Varchar
Evaluación	Varchar
Responsabilidad # 1	
Nombre	Datos de la muestra
Descripción	Constructor

Tabla 5 CE-LIS. Datos Muestra

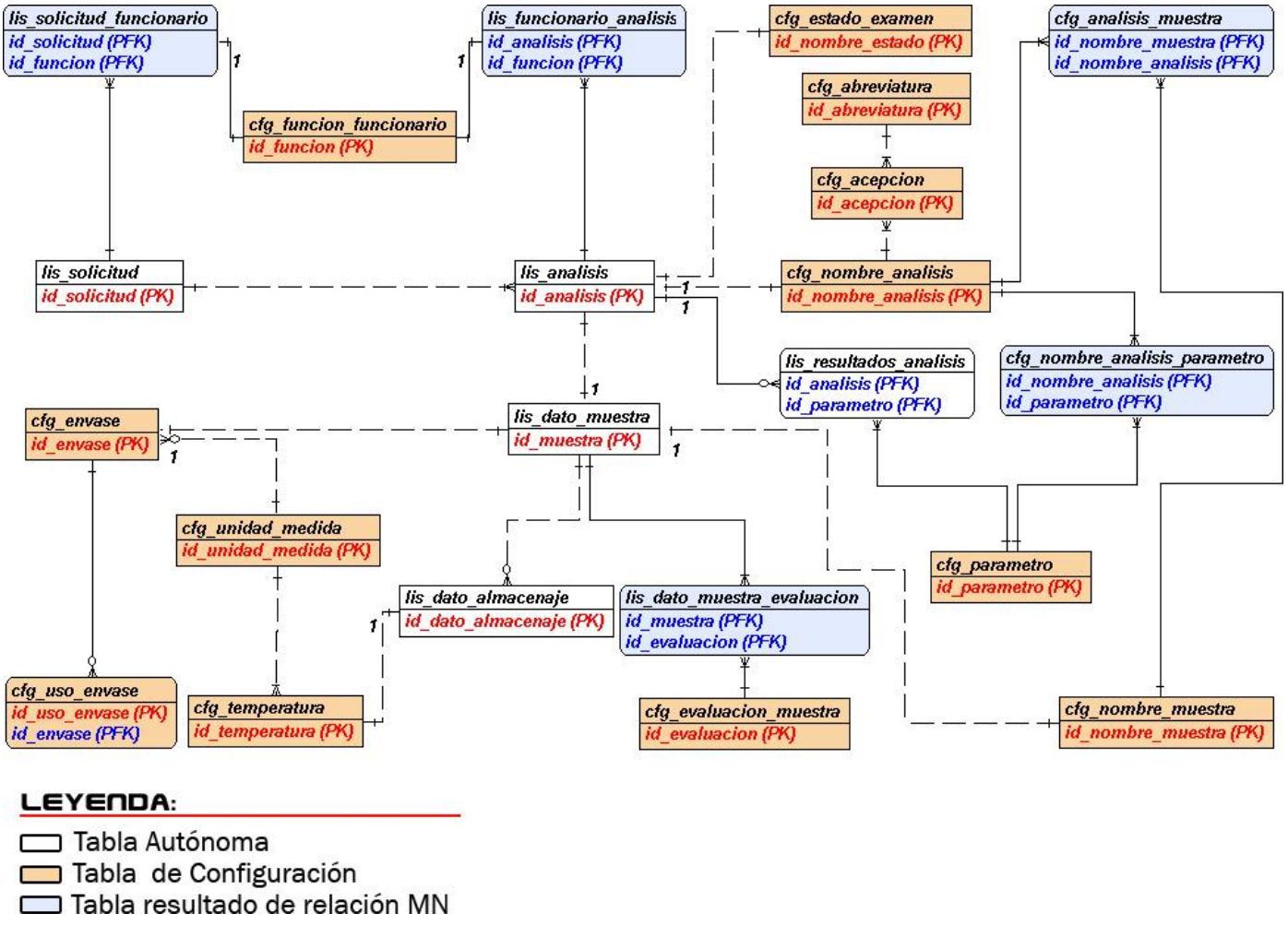
Nombre: CE_Estado_Examen	
Tipo de clase: Entidad	
Atributo	**Tipo**
Estado	Varchar
Razón_de_estado	Varchar
Responsabilidad # 1	
Nombre	
Descripción	

Tabla 6 E CE-LIS. Estado Examen

2.7 Diseño de la BD

2.7.1 DE-R

Diagrama Entidad-Relación del submódulo de Servicios



DER 1

Nota:

En el diagrama solo se muestran las relaciones, la cardinalidad y las llaves primarias. No se muestran las llaves foráneas y demás atributos por motivos de resolución de imagen. Para mas información consúltese el epígrafe 7.4 Descripción de tablas.

2.7.2 Sub-módulos LIS

LIS-Esquema de interacción de submódulos

2.7.3 Estadística de la BD

Entidades		**Total**
Independientes	15	
Dependientes	7	22
Atributos		**Total**
Llaves primarias (PK)	16	
Llaves primarias-foráneas (PFK)	13	
Llaves foráneas (FK)	17	
Atributos no primos	33	79
Relaciones		**Total**
Identificativas	15	
No-Identificativas	13	25
Índices		**Total**
b-tree	4	4
Restricciones		**Total**
Dominios	4	4
Funciones		**Total**
Select, Insert, Update, Delete	160	
Vistas	3	163
Roles		**Total**
Administrador	1	
Otros	3	4

Tabla estadística 1

7.4 Descripción de las tablas

Nombre: lis_análisis		
Descripción: Tabla que almacenará los datos referentes a cada análisis; registros de fechas de entrada y salida, Informes, notas adjuntas al mismo, entre otros…		
Atributo	**Tipo**	**Descripción**
Id_análisis (PK)	Varchar	Identificador de análisis Llave primaria. Representa un identificador que será único para cada análisis que se procese en el sistema.
Id_solicitud (PFK)	Varchar	Identificador de solicitud Llave primaria-foránea originaria de la tabla **lis_solicitud**.
Id_nombre_análisis (FK)	Varchar	Identificador de nombre de análisis Llave foránea originaria de la tabla **cfg_nombre_analisis**.
Fecha_entrada	TimeStamptz	Fecha de entrada del análisis al sistema. Fecha en que se comienza a procesar.
Fecha_salida	TimeStamptz	Fecha en que se le da salida al análisis y se emiten resultados.
Informe	Varchar	Informe del especialista (medico, técnico, etc.)
Nota_especial_técnico	Varchar	Nota especial que se quiera añadir al análisis por parte del personal del laboratorio que opera directamente con el análisis/examen (técnico).

Descripción de tabla 1- lis_analisis

Nombre: lis_solicitud		
Descripción: Tabla que almacenará los datos referentes a cada solicitud, vinculada con las tabas **ia_persona** y **cfg_funcionario** a través de id´s, además de almacenar las fechas de emisión, admisión y liberación.		
Atributo	**Tipo**	**Descripción**
Id_solicitud (PK)	Varchar	Identificador de solicitud Llave primaria. Representa un identificador que será único para cada solicitud que se haga en el sistema.
Id_persona (FK)	Integer	Identificador de persona: Llave foránea originaria de la tabla **ia_persona** (esta tabla no pertenece a la BD de LIS).
Id_prioridad (FK)	Integer	Identificador de prioridad Llave foránea que originaria de la tabla **cfg_prioridad_paciente** (no pertenece a la BD de LIS)
recomendaciones prev	Varchar	Recomendaciones previas que hace el proscriptor de la solicitud, en este caso el médico especialista, este campo es opcional por lo que puede ser nulo (**null**).
Fecha_emision	TimeStamptz	Fecha en la que se emite o se redacta la solicitud.
Fecha_liberacion	TimeStamptz	Fecha en la que se le da admisión/entrada al sistema. Esta fecha no se inserta a la hora de hacer la solicitud, por lo que en un principio adopta el valor de **null**.
Fecha_admision	TimeStamptz	Fecha en la que se libera la solicitud. Al igual que la anterior, esta fecha se inserta tiempo después de haber hecho la solicitud, por tanto en un principio adopta el valor de **null**.

Descripción de tabla 2- lis_solicitud

Nombre: lis_resultado_análisis		
Descripción: Tabla que recoge los datos de los resultados de cada análisis.		
Atributo	**Tipo**	**Descripción**
Id_análisis (FK)	Varchar	Identificador de análisis Llave primaria-foránea originaria de la tabla **lis_análisis**.
Id_parametro (FK)	Varchar	Identificador de parámetro Llave primaria-foránea originaria de la tabla **lis_parámetro**.
Informe	Varchar	Informe completo de los resultados del análisis.

Descripción de tabla 3- lis_resultado_análisis

Nombre: cfg_nombre_muestra		
Descripción: Tabla nomencladora que almacena los nombres que reciben las muestras.		
Atributo	**Tipo**	**Descripción**
Id_nombre_muestra	Integer	Identificador de nombre de muestra: Llave primaria que responde a un identificador único para cada nombre de muestra que se registra
Descripción	Varchar	Nombre de la muestra (heces, orina, sangre).

Descripción de tabla 4-cfg_nombre_muestra

Nombre: lis_dato_almacenaje		
Descripción: Tabla que recoge los datos necesarios para almacenar la muestra. Condiciones para el almacenaje		
Atributo	**Tipo**	**Descripción**
Id_muestra (PFK)	Varchar	Identificador de muestra: Llave primaria-foránea originaria de la tabla **lis_dato_muestra.**
Id_temperatura (FK)	Integer	Identificador de temperatura: Llave primaria-foránea originaria de la tabla **cfg_temperatura**.
Código_producto	Varchar	Identificador de producto: Llave foránea originaria de la tabla **cfg_producto (**no se encuentra en la Base de datos de LIS**)** que responde a un identificador único para cada producto.
Dato_adicional	Varchar	Dato adicional asociado a la muestra.
Fecha_hora_inicial	TimeStamptz	Fecha inicial de almacenamiento.
Fecha_hora_salida_est.	TimeStamptz	Fecha de salida estimada del almacén
Fecha_salida_real	TimeStamptz	Fecha de salida real del almacén.
Motivo_estadía	Varchar	Motivo por el cual se encuentra almacenada la muestra
Fotosensible	Boolean	Indica si la muestra es fotosensible o no.

Descripción de tabla 5- lis_dato_almacenaje

Nombre: lis_dato_muestra		
Descripción: Tabla que recoge los datos referentes a las muestras para su almacenamiento.		
Atributo	**Tipo**	**Descripción**
Id_muestra (PK)	Integer	Identificador de muestra: Llave primaria que responde a un identificador único para cada muestra ej. Muestra de sangre.
Id_nombre_muestra (FK)	Integer	Identificador de nombre de muestra: Llave primaria-foránea originaria de la tabla **lis_nombre_muestra**.
Id_análisis (FK)	Varchar	Identificador de análisis: Llave primaria-foránea originaria de la tabla **lis_análisis**.
Id_funcionario (FK)	Integer	Identificador de nombre de análisis: Llave primaria-foránea originaria de la tabla **lis_nombre_análisis**.
Fecha_recepción	TimeStamptz	Fecha e la que se recibe la muestra.
Localización	Varchar	Localización física de la muestra en el almacén.
Nota_especial_muestra	Varchar	Alguna nota especial que se le quiera añadir a la muestra.

Descripción de tabla 6- lis_dato_muestra

Nombre: lis_solicitud_funcionario		
Descripción: Tabla que se genera como resultado de una relación M a N entre las tablas **ia_funcionario** y **lis_solicitud**.		
Atributo	**Tipo**	**Descripción**
Id_funcionario (PK)	Integer	Identificador de funcionario Llave primaria-foránea originaria de la tabla **ia_funcionario**.
Id_solicitud (PFK)	Varchar	Identificador de solicitud Llave primaria-foránea originaria de la tabla **lis solicitud**.
Id_función (PFK)	Integer	Identificador de funcionario Llave primaria-foránea originaria de la tabla **cfg_función_funcionario**.

Descripción de tabla 5- lis solicitud_funcionario

Nombre: cfg_análisis_muestra		
Descripción: Tabla nomencladora que se genera como resultado de una relación M a N entre las tablas **cfg_nombre_muestra** y **cfg_nombre_análisis**		
Atributo	**Tipo:**	**Descripción:**
Id_nombre_anisáis	Varchar	Identificador de nombre de análisis: Llave primaria-foránea originaria de la tabla **cfg_nombre_análisis**.
Id_nombre_muestra	Integer	Identificador de nombre de muestra: Llave primaria-foránea originaria de la tabla **cfg_nombre_muestra**.
Muestra_defecto	Boolean	Indica si es la muestra que se toma por efecto para ése análisis.

Descripción de tabla 6- cfg_analisis_muestra

Nombre: lis_funcionario_análisis		
Descripción: Tabla que se genera como resultado de una relación M a N entre las tablas **Ia_funcionario** y **lis_análisis**.		
Atributo	**Tipo:**	**Descripción:**
Id_funcionario (PK)	Integer	Identificador de funcionario Llave primaria-foránea originaria de la tabla **ia_funcionario**.
Id_análisis (PFK)	Varchar	Identificador de análisis. Llave primaria-foránea originaria de la tabla **lis_análisis**.
Id_función (PFK)	Integer	Identificador de funcionario Llave primaria-foránea originaria de la tabla **cfg_función_funcionario**.

Descripción de tabla 7- lis_funcionario_analisis

Nombre: cfg_parámetro		
Descripción: Tabla nomencladora que almacenará en la BD los datos referentes a cada parámetro.		
Atributo	**Tipo**	**Descripción**
Id_parámetro (PK)	Varchar	Identificador de parámetros Llave primaria que responde a un identificador único para cada parámetro ej.: hemoglobina, linfocitos, etc.(responde al tipo de análisis dentro del sub-servicio correspondiente)
Id_subservicio (FK)	Integer	Identificador de sub-servicios. Llave foránea originaria de la tabla **cfg_subservicio.**
Tiempo_limite_horas	Integer	Tiempo de duración para la obtención de un resultado.
nombre	Varchar	Nombre del parámetro

Descripción de tabla 8- cfg_parametro

Nombre: cfg_acepción		
Descripción: Tabla nomencladora que recoge las acepciones (sinónimos) de los nombres de los Análisis.		
Atributo	**Tipo:**	**Descripción:**
Id_acepción (PK)	Integer	Identificador de acepción: Llave primaria que responde a un identificador único para cada acepción (sinónimo u homologo que pudiera tener algún análisis, según dialecto medico local).
Id_nombre_análisis (PFK)	Varchar	Identificador del nombres de análisis: Llave primaria-foránea originaria de la tabla **cfg_nombre_análisis**.
Id_abreviatura (FK)		Identificador de abreviatura: Llave foránea originaria de la tabla **cfg_abreviatura**.
Descripción	Varchar	Nombre del análisis. Ej.: Hemograma, Eritro, Urea, Glicemia, etc.
Defecto	Boolean	Determina si el nombre que se esta usando para un determinado análisis es el que se usa por defecto.

Descripción de tabla 9- cfg_acepcion

Nombre: cfg_uso_envase		
Descripción: Tabla nomencladora que recoge los diferentes usos que tiene cada envase del laboratorio.		
Atributo	**Tipo**	**Descripción**
Id_uso_envase (PK)	Integer	Identificador de uso Llave primaria que responde a un identificador único para cada uso o destinación del envase.
Id_envase (PFK)	Varchar	Identificador de envase: Llave primaria-foránea originaria de la tabla **cfg_envase**.
Descripción	Varchar	Descripción de uso o destinación.

Descripción de tabla 102-cfg_uso_envase

Nombre: lis_dato_muestra_evaluación		
Descripción: Tabla nomencladora que se genera como resultado de una relación M a N entre las tablas **Lis_dato_muestra** y **cfg_evaluación_muestra**		
Atributo	**Tipo**	**Descripción**
Id_muestra (PFK)	Varchar	Identificador de muestra: Llave primaria-foránea originaria de la tabla **lis_dato_muestra**.
Id_evaluación (PFK)	Varchar	Identificador de evaluación: Llave primaria-foránea originaria de la tabla **cfg_evaluación_muestra**.

Descripción de tabla 113-lis_dato_muestra_evaluacion

Nombre: cfg_temperatura		
Descripción: Tabla nomencladora que almacena los datos referentes a la temperatura. Contiene la temperatura específica o rango de temperaturas requeridas por una muestra.		
Atributo	**Tipo**	**Descripción**
Id_temperatura (PK)	Integer	Identificador de temperatura Llave primaria que responde a un identificador único para cada rango de temperaturas. El rango de temperaturas estará definido según la técnica y necesidades del laboratorio.
Id_unidad_medida (PFK)	Varchar	Identificador de unidad de medida: Llave primaria-foránea originaria de la tabla **cfg_unidad_medida**.
Temp_min	Integer	Temperatura inicial o minina del rango de temperatura
Temp_max	Integer	Temperatura final o máxima del rango de temperatura

Descripción de tabla 124-cfg_temperatura

Nombre: cfg_unidad_medida		
Descripción: Tabla nomencladora que almacena las unidades de medida que se usan en el laboratorio		
Atributo	**Tipo:**	**Descripción:**
Id unidad medida	Integer	Identificador de unidad de medida: Llave primaria que responde a un identificador único para cada unidad de medida
Unidad medida nombre	Varchar	Nombre de la unidad de medida.

Descripción de tabla 135-cfg_unidad de medida

Nombre: cfg_nombre_análisis_parámetro		
Descripción: Tabla nomencladora que se genera como resultado de una relación M a N entre las tablas **Cfg_nombre_análisis** y **cfg_parámetro**		
Atributo	**Tipo**	**Descripción**
Id_nombre_análisis (PFK)	Varchar	Identificador de unidad de medida: Llave primaria-foránea originaria de la tabla **cfg_nombre_análisis**.
Id_parámetro (PFK)	Varchar	Identificador de parámetro: Llave primaria-foránea originaria de la tabla **cfg_parámetro**.

Descripción de tabla 14-cfg_nombre_analsis_parametro

Nombre: cfg_nombre_análisis.		
Descripción: Tabla nomencladora que contiene los identificadores de los nombres de los análisis.		
Atributo	**Tipo**	**Descripción**
Id_nombre_análisis (PK)	Varchar	Identificador de nombre de análisis. Llave primaria que responde a un identificador único para cada nombre de análisis.
Id_subservicio (FK)	Integer	Llave foránea: Identificador único perteneciente a la tabla **cfg_subservicio** (no se encuentra en la BD de LIS). Servicio que ofrece el laboratorio/Nombre de departamento ej. Hematología[12], Inmunología[15], Microbiología, etc.

Descripción de tabla 15-cfg_nombre_analisis

Nombre: cfg_función_funcionario		
Descripción: Tabla nomencladora. Contiene la función que desempeñan los distintos funcionarios del laboratorio.		
Atributo	**Tipo**	**Descripción**
Id_función (PK)	Integer	Identificador de función: Llave primaria que responde a un identificador único para cada función de funcionario.
Descripción	Varchar	Función, cargo o papel que desempeña el funcionario.

Descripción de tabla 16-cfg_funcion_funcionario

Nombre: cfg_evaluación_muestra		
Descripción: Tabla nomencladora. Contiene los datos cualitativos de las evaluaciones posibles para cada muestra.		
Atributo	**Tipo**	**Descripción**
Id_evaluación (PK)	Varchar	Identificador de evaluación: Llave primaria que responde a un identificador único para cada evaluación.
Descripción	Varchar	Nombre cualitativo de la evaluación – ej. : Bien, Mal, Regular.

Descripción de tabla 17-cfg_evaluacion_muestra

Nombre: cfg_estado_examen		
Descripción: Tabla nomencladora que contiene los diferentes estados de examen/análisis.		
Atributo	**Tipo**	**Descripción**
Id_nombre_estado (PK)	Varchar	Identificador de estado: Llave primaria que responde a un identificador único para cada estado de examen.
Descripción	Varchar	Descripción o estado físico en que se encuentra examen/análisis; ej. : Emitido, Recepcionado, Procesado, Liberado, etc.

Descripción de tabla 18- cfg_estado_examen

Nombre: cfg_envase		
Descripción: Tabla nomencladora que contiene los datos referentes a los envase. Esta tabla puede contener información ajena al laboratorio (envases que se almacenan que no son pertenecientes al Laboratorio).		
Atributo	**Tipo**	**Descripción**
Id_envase (PK)	Varchar	Identificador de envase: Llave primaria que responde a un identificador único para cada envase.
Id_muestra (FK)	Varchar	Identificador de muestra: Llave foránea originaria de la tabla **lis_dato_muestra**.
Id_unidad_medida (FK)	Varchar	Identificador de unidad de medida: Llave foránea originaria de la tabla **cfg_unidad_medida**.
Descripción	Varchar	Descripción, especificación o nombre del envase.
Capacidad	Double precision	Capacidad del envase. Volumen.

Descripción de tabla 19- cfg_envase

Nombre: cfg_abreviatura		
Descripción: Tabla nomencladora que contiene las abreviaturas que se utilizan para cada Acepción/Nombre Científico de cada análisis.		
Atributo	**Tipo**	**Descripción**
Id_abreviatura (PK)	Serial	Identificador de abreviatura: Llave primaria que responde a un identificador único para cada abreviatura.
Descripción	Varchar	Abreviatura. Ej. : HG (Hemograma), HGD (Hemograma con diferencial), HE (Eritro).

Descripción de tabla 20- cfg_abreviatura

Conclusiones

En el capitulo, se analizó como integrar la base de datos de forma que se cumplan los principios de seguridad, garantizando a través de la integración a nivel de negocio que subsistemas adyacentes tengan un acceso controlado; y solo asuman permisos de lectura sobre las tablas, tanto las autónomas como las de configuración.

Asimismo se hizo una descripción de la arquitectura, determinando que por el momento es óptimo mantener y controlar la información a través de un solo equipo, recomendando en ciclos posteriores de desarrollo, migrar hacia Arquitecturas distribuidas.

Para optimizar,sentencias del lenguaje de consulta. Se obtuvieron sentencias de menor tiempo de ejecución y de mejor desempeño, de manera que se agilizó el tiempo de respuesta y procesamiento del servidor.

Se concretaron todos los requisitos funcionales y no funcionales del sistema, y se dió a conocer el modelo de objetos o diagrama de clases persistentes resultante del diagrama de clases del diseño, todo ello, a partir del trabajo desarrollado por el analista.

Se presentó el diseño del Diagrama de Entidad-Relación de la base de datos, que solo abarca el submódulo de servicios. Fueron revelados algunos datos estadísticos de la BD con el objetivo de suministrar un resumen matemático que facilite la apreciación del alcance del sistema. Además, fueron descritas cada una de las tablas, pasando por el nombre, tipo y descripción de cada uno de sus atributos.

En el presente capitulo, se hace una profunda valoración del trabajo realizado sobre la base de datos. Se muestra la vía de solución, que fue asumida para resolver la problemática planteada inicialmente.

Se hace una validación teórica del diseño realizado, donde se tomarán en cuenta aspectos como la Integridad en el SBD y se expondrán los distintos tipos de integridad que utilizaron y la implementación de algunos de ellos.

Otro aspecto fundamental, lo constituye la Normalización, para su análisis se abordan una serie de conceptos que definen y justifican el estado o grado de normalización que tiene la Base de datos actualmente. Como parte inherente de este contenido se encuentra realizar el estudio referente al control de la información que es redundante en la BD.

Posteriormente, se cita de temas como la Seguridad y la Trazabilidad de acciones, elementos a tener muy en cuenta a la hora de desarrollar una aplicación robusta, que mantenga un acceso controlado, resguardo de la información y seguimiento de las acciones que son llevadas a cabo sobre cada registro contenido en la base de datos.

3.2 Integridad de la BD

Durante la manipulación de los datos pueden producirse todo tipo de problemas como pudieran ser:

- Usuarios que manipulan los mismos datos al mismo tiempo: No se pueden destruir ni modificar los datos de forma anómala.
- Fallos en el hardware o errores del sistema: Se ha de asegurar que en el sistema a pesar de estos errores los datos siguen siendo válidos. Por ello, se han de establecer los procedimientos necesarios que verifiquen que los valores de los datos se ajusten a los requerimientos y restricciones extraídos del análisis del problema.

🗁*3.2.1 Termino integridad*

El término integridad de datos se refiere a la corrección y completitud de los datos en una base de datos. Cuando los contenidos de una base de datos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. [26] En el caso particular de LIS se tiene por ejemplo que:

- Pueden añadirse datos no válidos a la base de datos, tales como una solicitud que especifica un examen/análisis no existente (el laboratorio no ofrece el servicio).

- Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo si se reasigna un análisis a una solicitud no existente.
- Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un análisis a una solicitud sin ajustar las muestras que son necesarias para realizar dicho análisis.
- Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía.

3.2.2 Aplicación de restricciones

Las restricciones de integridad garantizan que el contenido de la base de datos sea conforme con las reglas establecidas para presentar el Universo del Discurso. La integridad de la base de datos no es más que la existencia de dos componentes importantes, que son: la exactitud (CORRECTNESS) y la completitud (COMPLETENESS). Es decir, que la integridad garantiza que todos los datos son correctos (validos) y relevantes. [27].

Las restricciones de integridad son las limitaciones que se impusieron para proteger la base de datos, de modo que nunca llegue a un estado inconsistente. Con respecto a esto se aplicaron varios tipos de restricciones de integridad: datos requeridos, restricciones de dominio, integridad referencial y de entidades y reglas de negocio; todas ellas se deben tener un estricto cumplimiento en la base de datos en todos sus estados o instancias.

Datos requeridos

Algunos de los atributos contienen valores en todo momento, es decir, no se admiten valores nulos (NOT NULL).

Reglas de integridad

Una de las funciones más importantes de un DBMS relacional es preservar la integridad de sus datos almacenados en la mayor medida posible. Existen básicamente 2 reglas de integridad asociadas con el modelo relacional, la Integridad de Entidad y la Integridad Referencial. Estas dos reglas son generales, se aplican a toda base de datos relacional y tienen que ver con las llaves primarias y externas respectivamente. Estas reglas se refieren a los estados de la base de datos. Dado que no existe un acuerdo de como se deben evitar los cambios de estado en la base de datos es que muchos SGBD detienen la ejecución de la tarea en curso cada vez que se incurre en una violación a una de estas reglas.

La regla de Integridad de Entidad norma sobre la imposibilidad de que un atributo que componga la llave primaria de una relación base acepte valores nulos (***NULL***).

La regla de integridad referencial para el modelo relacional formula que si una relación R2 incluye una llave externa FK que empareja a una llave primaria PK de alguna relación R1, entonces cada valor de FK en R2 debe:

a) Ser igual al valor de PK en alguna tupla de R1.
O por el contrario
b) Ser totalmente Nula, esto es, cada atributo en FK debe ser nulo. [28]

Restricciones del negocio

Además de las reglas de integridad anteriores, los usuarios o los administradores de la base de datos pueden imponer ciertas restricciones específicas sobre los datos, Por tanto, cualquier operación que se realice sobre los datos debe cumplir las restricciones que se imponen. [29]

Restricciones de Dominio

Se definió cada atributo sobre un dominio, o sea, se impuso una restricción sobre el conjunto de valores permitidos para cada atributo que puede estar determinado tanto por una longitud, como por un intervalo, en dependencia de la expresión condicional que se utilice. [30]

Ejemplos de Restricciones de Dominio implementadas

Nombre del Dominio: Positivo		
Descripción: Dominio para algunos valores enteros. Chequea que el valor sea siempre positivo (+).		
Atributo(s) que se restringe(n):	**Tipo:**	**Tabla a la que pertenece(n):**
Capacidad	Integer	Cfg_envase
Tiempo_limite_horas	Integer	Cfg_parametro
Check: value > 0		

Tabla de dominio 1

Nombre del Dominio: Temperatura		
Descripción: Dominio para las temperaturas máximas y mínimas. Se establece rango de -100 a 100 ºC.		
Atributo(s) que se restringe(n):	**Tipo:**	**Tabla a la que pertenece(n):**
Temp_min	Integer	Cfg_temperatura
Temp_max	Integer	Cfg_temperatura
Check: value >= - 100 and value <= 100		

Tabla de dominio 2

3.2.4 Tablas nomencladoras

Como estandarización de los datos, se crearon 12 tablas de Configuración (nomencladoras), que contienen datos previamente definidos, que a su vez pueden ser modificados mas tarde sólo por el administrador del sistema (seguridad), evitando así que se dé a lugar la inserción de un mismo dato de varias formas. De esta forma, se garantiza que la base de datos esté libre de duplicidades.

3.3 Normalización de la BD

3.3.1 Que es la normalización? Ventajas

Normalización es un conjunto de reglas que sirven para ayudar a los diseñadores a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataba de manipular los datos.

Una de las ventajas de la normalización, en el caso particular de esta base de datos una de las de mayor peso es el consumo de espacio. Una base de datos normalizada puede ocupar menos espacio en disco que una no normalizada. Hay menos duplicidad de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco ya que las reglas de Normalización están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas. [30].

3.3.2 Análisis previo

El primer punto que se analizó y se tuvo muy en cuenta fueron los requisitos existentes para procurar el diseño de una BD realmente funcional y útil. Por tanto, el primero de estas exigencias a cumplir fue la total eliminación de las Anomalías.

Los tipos de anomalías que fueron más frecuentes:

1. Actualización
2. Inserción
3. Eliminación

Conforme se fue avanzando en el proyecto, se hizo verídico que el llevar de un DE-R a un modelo de relaciones (a un modelo relacional), no bastaba para eliminar estas anomalías y asegurar la consistencia en la BD (Consistencia de los datos), haciendo referencia fundamentalmente a que se

mantuvieran, tanto lo atributos de las relaciones, como la información que contienen estas relaciones, así como también las Dependencias Funcionales (DF) que tienen cada una de ellas. Por lo que se transitó por un proceso mediante el cual se descompuso un grupo relaciones que no cumplían con una serie de requisitos en relaciones más pequeñas o sea en esquemas de relación más pequeños.

Teniendo como principales objetivos:

- ☒ Eliminar en primer lugar las Anomalías (como se expresó anteriormente).
- ☒ Evitar o eliminar en el mayor grado posible las redundancias. [31].

3.3.3 Grado actual de normalización de la BD de LIS

La base de datos se encuentra en 3ra Forma Normal (3FN) ya que en todas sus relaciones se cumple que:

Primero:

- ✓ Cada relación prohíbe que un atributo tenga como valor una tupla o un conjunto de valores (no existen atributos multivaluados).
- ✓ Prohíbe las relaciones dentro de relaciones.
- ✓ Solo se permiten valores atómicos.

✡ Por lo que cumple con la Primera Forma Normal (1FN). La que conceptualmente explica que NO se puedan tener en una relación atributos multivaluados (que tengan mas de un valor) o atributos compuestos.

Segundo:

- ✓ Los atributos no llave dependen totalmente de la llave primaria y no solamente de una parte de ella, sino de todos y cada uno de los atributos que la componen.

✡ Por lo que cumple con la normativa que establece la Segunda Forma Normal (2FN), que está muy relacionada con el concepto de Dependencia Funcional Total (DFT).

Tercero:

- ✓ Está en 2da FN.
- ✓ Ninguno de los atributos no primos de sus relaciones depende transitivamente de la clave primaria.

✡ Cumpliendo con el lineamiento establecido por la Tercera Forma Normal (3FN) que esta basada en el concepto de Dependencia Transitiva.

Se tomó la decisión de llegar a esta Forma Normal por varias razones, una de ellas es que está establecido que para lograr un buen diseño de una base de datos, esta debe estar al menos en tercera Forma Normal, además se tuvo en cuenta que un alto grado de normalización (dígase FNBC, 4ªFN o 5ªFN) no es objetivo ya que ocasionaría que el diseño se torne rígido y poco flexible, privando a diseños futuros de la adaptación e inclusión de nuevos requisitos. [32].

3.4 Análisis de redundancia de la información

Los datos en el sistema cumplen ciertas restricciones que aseguran la correcta introducción, modificación y borrado de los mismos. En la base de datos de LIS la redundancia de los datos es controlada de forma que no existan duplicaciones perjudiciales ni innecesarias, las redundancias físicas (conveniente muchas veces), en caso de existir, son tratadas automáticamente por el propio sistema para que no puedan existir inconsistencias. Por tanto, un dato se actualizará lógicamente por el usuario de forma única, y el sistema se preocupará de cambiar físicamente todos aquellos campos en los que el dato estuviese repetido en caso de existir redundancia física; Esto lo que se conoce/denomina también redundancia controlada por el sistema.

3.5 Análisis de la seguridad

Una base de datos *es* un conjunto de datos integrados, adecuado a varios usuarios y a diferentes usos. Es el propio uso concurrente de los datos el que plantea problemas de seguridad que el Administrador de la Base de Datos debe atenuar en la medida de lo posible con las funcionalidades que le proporciona el SGBD. La protección de los datos deberá llevarse a cabo contra fallos físicos, fallos lógicos y fallos humanos (intencionados o no). Estos fallos alteran indebidamente los datos o los corrompen, con lo que la BD ya no puede servir a los fines para los que fue creada.

El manejo de la base de datos será cada vez mayor, debido a la necesidad de realizar registros. La primordial importancia de la seguridad radica en que estos sean accesibles solamente por las personas autorizadas. Para ello resulta indispensable proporcionar privilegios a distintos tipos de usuarios con el fin de mantener la integridad, disponibilidad y confidencialidad de la información que se encuentra almacenada en la bases de datos. [33].

En la base de datos de LIS, Las tablas pertenecen a un rol y fueron otorgados privilegios granulares a estos roles, dentro de los cuales se incorporaron usuarios. De ese modo se obtiene acceso controlado a nivel de fila. La base de datos hasta el momento tiene definidos 4 roles, que pueden ser asignados a usuarios o grupos de usuarios, según el administrador del SBD (conocido también como superusuario) estime conveniente. Estos roles tienen diferentes niveles de acceso a las tablas e incluso diferentes privilegios sobre las mismas. Dichos privilegios han sido cuidadosamente chequeados, con el objetivo de limitar la lectura, escritura, inserción y actualización de los registros en el sistema.

3.5.1 Copias de seguridad y restauración

Deben realizarse copias de seguridad de la base de datos regularmente. Dado que Postgres gestiona sus propios ficheros en el sistema, *no se recomienda* confiar en los sistemas de copia de seguridad del sistema para las copias de respaldo de las bases de datos; ya que no hay garantía de que los ficheros estén en un estado consistente que permita su uso después de la restauración.

PostgreSQL proporciona dos utilidades para realizar las copias de seguridad de su sistema: *pg_dump* para copias de seguridad de bases de datos individuales y *pg_dumpall* para realizar copias de seguridad de toda la instalación de una sola vez. [34]

3.5.1 Encriptación y cifrado. Límite de conexiones

Hasta el momento no se ha especificado que el sistema tenga absoluta seguridad con respecto al personal que operará con la misma, en este caso el administrador o superusuario. Es de señalar que el contenido de las columnas está en “texto claro” por lo que el administrador de la base de datos los puede ver, lo que pidiera constituir un factor de riesgo, que compromete la privacidad y confiabilidad de los datos. Para evitar esto, en iteraciones siguientes, se pudiera usar la librería ***pg_crypto*** para cifrar simétricamente la columna de forma transparente.

En caso de que esté presente el riesgo de que las consultas sean “robadas” (intervenidas a través de la red), lo que se pudiera hacer sería cifrar las conexiones; para esto PostgreSQL soporte criptografía uni/bilateral basada en certificados X.509. Luego se hacen las conexiones a la vez que se generan certificados para aplicaciones clientes, y con esto se logra que solamente tengan acceso al clúster los usuarios con un certificado correctamente firmado. [35].

Igualmente como medida de seguridad, el administrador implantará un número finito de conexiones al servidor, configurando el fichero *pd_hba.conf*. De esta forma el servidor solo recibirá las conexiones de

los host en los cuales “confía” (que estén definidos en éste fichero). Para adicionar una conexión al servidor se establecen varios parámetros como son: la base de datos a la que se va a acceder, el usuario (el cual debe estar previamente definido en el sistema) que estará a cargo de esa conexión, la dirección IP del host desde el cual se efectúa la conexión, y el método deberá ser *trust*. Ejemplo:

 # IPv4 local connections:

# TYPE	***DATABASE***	***USER***	***CIDR-ADDRESS***		***METHOD***
host	all	all	10.7.5.50	255.255.255.255	trust

3.6 Trazabilidad de acciones

La trazabilidad es un sistema que deja rastro de las acciones que el usuario realiza con los datos. Por ejemplo, si un usuario cambia un registro, queda constancia del registro que se cambió, quien lo cambió y cuando. De esta forma, puede hacerse un seguimiento de todas las acciones que se han realizado sobre un registro en particular. O todas las acciones que ha realizado un usuario concreto. Pudiendo conformar un historial de las acciones llevadas sobre los datos. La trazabilidad puede realizarse sobre tablas o sobre procesos. Incluso pueden crearse trazabilidades sobre errores o conflictos de la aplicación (logs). [36].

Los logs, son los ficheros encargados de archivar la información acerca de todas las acciones que se llevan a cabo sobre los datos. Con respecto a estos logs, se pueden tomar ciertas medidas ya que son configurables, desde el fichero postgresql.conf. Comenzando por su directorio, el que puede ser absoluto o relativo al archivo llamado “data” en los archivos de instalación del propio gestor. Si se deseara cambiar la ubicación de los log, lo que debe hacerse es cambiarle el directorio, pudiendo ser muy efectivo asignarle un fichero con restricciones de lectura y escritura (proteger el archivo), de manera que el administrador tenga acceso exclusivo a dicho directorio. A continuación se muestra la cláusula pertinente a la dirección de los logs:

#log_directory = 'pg_log'

Los logs tienen un patrón de nombre que sigue la estructura siguiente:

#log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'

Lo que indica que cada fichero que se genere tendrá el nombre: postgresql-año-mes-dia_hora-minuto-segundo.log, lo que indica claramente el nombre del usuario, la fecha y hora exactas de su creación.

Se pueden activar opciones para que se guarden todas las trazas o para que se guarden solamente las ultimas que son generadas. Ello depende del valor que se le otorgue a la siguiente clausula:

#log_truncate_on_rotation = off/on

Si el valor es ***on***, significa que cualquier log existente con el mismo nombre será reemplazado por el nuevo log que se genera, siendo el caso contrario cuando el valor esta en ***off*** (por defecto), lo cual significa que se añadirán logs en todos los casos. En el caso particular de esta BD, se recomienda mantener este valor como viene por defecto, ya que al no sobrescribir los ficheros, no se pierde información acerca de acciones realizadas con anterioridad.

También se pueden establecer otras medidas con el objetivo de liberar espacio, teniendo en cuenta que serán muchos los ficheros que se generarán, a partir de las acciones que serán llevadas sobre la BD por los diferentes usuarios, esto puede lograrse de dos formas:

La primera es a través de:

#log_rotation_age = 15d → *Rotación automática en 15 días.*

A partir de esta clausula se puede establecer el tiempo que se estime conveniente mantener los logs. Pasado este período de tiempo, el sistema hará una rotación automática de los ficheros sobrescribiendo los más viejos por los nuevos. Para deshabilitar la rotación automática se debe poner el valor en cero (0).

La segunda forma es a través de:

#log_rotation_size = 10MB → *Rotación automática al llegar a 10 Mega Bytes.*

Con el uso de la misma se hace la rotación automática de ficheros cuando el archivo '*pg_log'* llegue a la cota de "***n***" Mega Bytes que ha sido impuesta. Ésta, al igual que la cláusula anterior puede ser anulada o deshabilitada introduciendo "cero" (0) como valor.

Conclusiones

Se analizó la integridad, aplicando restricciones de modo que la base de datos nunca llegue a un estado inconsistente; se garantizó que todos los datos sean válidos y relevantes. Se llevó la base de datos a una Tercera Forma Normal (3ªFN), lo que constituye un grado de Normalización aceptable para una primera iteración en el diseño.

Fueron creados los roles y otorgados privilegios granulares a estos, dentro de los cuales se incorporaron usuarios. Se dieron alternativas de copias de seguridad, recomendando las herramientas con las que cuenta el gestor para este propósito.

Para futuras iteraciones se recomienda la expansión de la BD a los módulos de Logística y Administración e integrarlos al sub-modulo de Servicios ya existente. Se recomienda el diseño de los mismos en el orden indicado, ya que de esta forma se va de un nivel menor a uno de mayor complejidad.

Cuando se llegue a la concepción del modelo definitivo del Sistema de Información para Laboratorios, se propone llevar el modelo relacional a una FN superior como la FNBC (Forma Normal de Boyle-Codd).

Realizar una validación funcional del diseño realizado, donde se traten temas como: Generación de código de programas para un llenado voluminoso e inteligente de la base de datos y Búsqueda o diseño de herramientas para pruebas de carga intensiva haciendo énfasis en:

1. La selección de las consultas más voluminosas y frecuentes.
2. Análisis y control de concurrencia.

CONCLUSIONES

Para concluir este documento se puede afirmar que se cumplieron con las tareas de la investigación que fueron fijadas, asintiendo a favor de que se logó diseñar la es Sistema de base de datos que dará soporte al Sistema de Información de Laboratorios, asociado a ello está el estudio y empleo a fondo las herramientas definidas u la obtención de los artefactos de la metodología RUP que describen la base de datos.

Por otra parte se analizaron cuestiones como la integración del sistema con otros módulos o sus-sistemas independientes. Se implementó una capa de acceso a datos con 160 funciones aproximadamente, satisfaciendo los requisitos funcionales que planteaba el Sistema.

Se llevó a cabo una validación teórica y funcional del diseño de la base de datos y se plasmó una exhaustiva valoración de los resultados obtenidos, a partir de los cuales se tuvo apoyo para hacer las propuestas para próximas iteraciones.

Además de esto, la base de datos ha sido diseñada para ser manipulada a través de un sistema digital el cual mejorará considerablemente de los procesos de la gestión de información relacionados con los laboratorios clínicos en los Hospitales y Centros Hospitalarios Cubanos.

REFERENCIAS BIBLIOGRAFICAS

[1]. Salamanca laboratorio Clínico. Alfa21-Software para Laboratorios Clínicos. (2006)[Revisado el 10 de Noviembre del 2006]. Disponible en: http://www.slclab.com/alfa21/introduccion.htm

[2]. CMLab MCS software para la gestión del laboratorio clínico. Millenium, casa del software. (2002) [Revisado el 12 de Noviembre 2006]. Disponible en: http://www.cmlab.com.ve/principal.asp

[3]. Sistemas de Información para el Manejo de Laboratorios Médicos Eficientes [Revisado el 12 Noviembre del 2006]. Disponible en: http://www.medilabit.com

[4]. Health care international Services (2005) [Revisado el 10 Noviembre del 2007]. Disponible en: http://www.hcisonline.com/online/software.asp?id=6

[5]. SOFTEL, Manual de Usuario GALEN LAB, Edición de la versión 5.0. [Revisado el 15 de noviembre de 2006]

[6]. Ferrara Amaro Carlos Alberto, G. F. N., Jiménez Trinidad Julio César, Montaño Flores Marco Antonio, Ocampo Mota Pamela. Herramientas CASE [Revisado el 25 de Noviembre del 2007]. Disponible en: http://www.dsic.upv.es/~alimartin/Material-webFDS/Presentaciones/Expo%20Herramientas%20Case.ppt.

[7]. Márquez M. M. A. Apuntes de Ficheros y Bases de Datos. Herramientas CASE (2001-02-12) [Revisado el 20 de Noviembre del 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node75.html

[8]. Rational Rose. (2001)[Revisado el 15 de diciembre de 2006]. Disponible en: http://www.indudata.com/1rational_rose.htm

[9]. CASE Studio. (2006)[Revisado en Enero del 2007]. Disponible en: http://case-studio.softonic.com/

[10]. Márquez M. M. A. El Modelo entidad-relación. (2002)[Revisado el 20 de Noviembre del 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node83.html

[11]. Márquez M. M. A. El Modelo relacional. (2002)[Revisado el 20 de Noviembre del 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node43.html

[12]. Concepto de base de datos. Disponible en: http://es.wikipedia.org/wiki/Base_de_datos#Tipos_de_bases_de_datos

[13]. Márquez M. M. A. Metodología de diseño de las bases de datos. (2002)[Revisado el 20 de Noviembre del 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node43.html

[14]. Valle J. Bases de Datos (2007) [Revisado en Marzo del 2007]. Disponible en: http://www.monografias.com/trabajos24/bases-de-datos/bases-de-datos.shtml

[15]. Andrés, M. M. M. Resumen [Revisado el 10 Noviembre del 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node9.html

[16]. Burbano D. J. Análisis comparativo de bases de datos de Código abierto vs. Código cerrado. (2006)[Revisado en Diciembre del 2007] http://www.monografias.com/trabajos35/comparativa-bases-datos/comparativa-bases-datos.shtml#sistemagestor

[17]. Neira M. C. Tipos de SGBD (2002) [Revisado en Diciembre del 2007]. Disponible en: http://macine.epublish.cl/tesis/index-3_3_.html

[18]. My Group. Net - Comunidad de programación. 2005. [2007]. Disponible en: http://www.Mygnet.net/artículos

[19]. A., E. Q. Introducción a PostgreSQL, 2007]. Disponible en: http://www.apesol.org.pe

[20]. EMS SQL Manager for PostgreSQL. 2005. [2007]. Disponible en: http://sqlmanager.net/products/postgresql/manager/

[21]. Desarrollo de Aplicaciones con Sistemas de Gestión de Bases de Datos. [Revisado el 20 Febrero del 2007]. Disponible en: http://alarcos.inf-cr.uclm.es/doc/aplicabbdd/Documentos/teoria/arquitecturas%20para%20bases%20de%20datos.pdf

[22]. Neira, M. C. Métodos de Optimización de Consultas para el leguaje SQL (2002) [Revisado el 4 de Abril del 2007]. Disponible en: http://macine.epublish.cl/tesis/index-Definici.html

[23]. Neira, M. C. Métodos de Optimización de Consultas para el leguaje SQL (2002) [Revisado el 4 de Abril del 2007]. Disponible en: http://macine.epublish.cl/tesis/index-Resumen-2.html

[24]. Andrés, M. M. M. Índices. [Revisado en Marzo del 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node24.html

[25]. PostgreSQL 8.1.9 Documentación (2005) [Revisado en Marzo del 2007]. Disponible en: http://www.postgresql.org/docs/8.1/static/sql-explain.html

[26]. Wikkipedia. Definición de Integridad de Datos. [Revisado en Mayo del 2007]. Disponible en: http://es.wikipedia.org/wiki/Integridad_de_datos

[27]. LaBDA Laboratorios de Bases de Dato Avanzadas. Control de restricciones de integridad en la BD (2005) [Revisado el 7 Mayo del 2007] http://basesdatos.uc3m.es/index.php?id=206

[28]. Andrés, M. M. M. Reglas de integridad. [Revisado en Mayo del 2007]. Disponible en: http://macine.epublish.cl/tesis/index-1_3_.html

[29]. MailxMais. Cursos gratuitos. Diseño lógico de bases de datos. [Revisado en mayo del 2007]. Disponible en: http://www.mailxmail.com/curso/informatica/disenobasesdatosrelacionales/capitulo8.htm

[30]. Wise B. Normalización de Bases de Datos y Técnicas de diseño (2001) [Revisado el 25 de Mayo del 2007]. Disponible en: http://bulma.net/body.phtml?nIdNoticia=483

[31]. Idem [29].

[32]. Teleconferencia 5 del plan de clases basadas en el libro DATE, C. J. Introducción a los Sistemas de Bases de Datos. Capítulos 10 y 11.

[33]. Idem [31]

[34]. De Santiago S. M. A. Monografías.com. Seguridad de las Bases de Datos. [Revisado en Junio del 2007]. Disponible en: http://www.monografias.com/trabajos26/seguridad-base-datos/seguridad-base-datos2.shtml

[35]. Krosing H. Guía del Administrador de PostgreSQL. [Revisado en Junio del 2007]. Disponible en: http://es.tldp.org/Postgresql-es/web/navegable/admin/x1401.html

[36]. Hernández E. N. [l-linux] Seguridad de datos usando query's firmados. [Revisado en Junio del 2007]. http://www.velug.org.ve/archivo/l-linux-2007-April/061951.html

[37]. Monte G J. L. Generadores de Código (2006). [Revisado el 20 Junio del 2007] http://www.moga.awardspace.com/wp-content/uploads/2007/03/gencod.pdf

[38]. Martínez, R. A. and L. CIENFUEGOS. PostgreSQL, 2007]. Disponible en: http://www.ehtcf.cu/linuxcf/2007/enero_PostgreSQL.pdf

BIBLIOGRAFÍA

Bibliografía consultada

1. Aspectos generales del Sistema Nacional de Salud (2000). Disponible en: http://www.sld.cu/sistema_de_salud/aspectos.html.

2. Colaboradores de Wikipedia. PostgreSQL (2007) Disponible en: http://es.wikipedia.org/wiki/PostgreSQL

3. DATE, C. J. Introducción a los Sistemas de Bases de Datos. 7ª Edición. Editorial: Félix Varela, p.

4. Halth Care International Services. Disponible en: http://www.hcisonline.com/online/sp/home.asp

5. Márquez M. M. A. Apuntes de Ficheros y Bases de Datos. p. Total de paginas 184 (2001-02-12). Disponible en: http://www3.uji.es/~mmarques/f47/apun/apun.pdf

6. Métodos de Optimización de Consultas para el lenguaje SQL. Matemáticas Y Ciencias de la Computación. Stgo. de Chile. Universidad de Santiago de Chile. Fac. de Ciencias, 2002. p. Total de páginas: 185. Disponible en: http://macine.epublish.cl/tesis/memoria.pdf

7. NOEMALIFE we care. Disponible en: http://www.magic-cms.net/noemalife/page_ES.asp

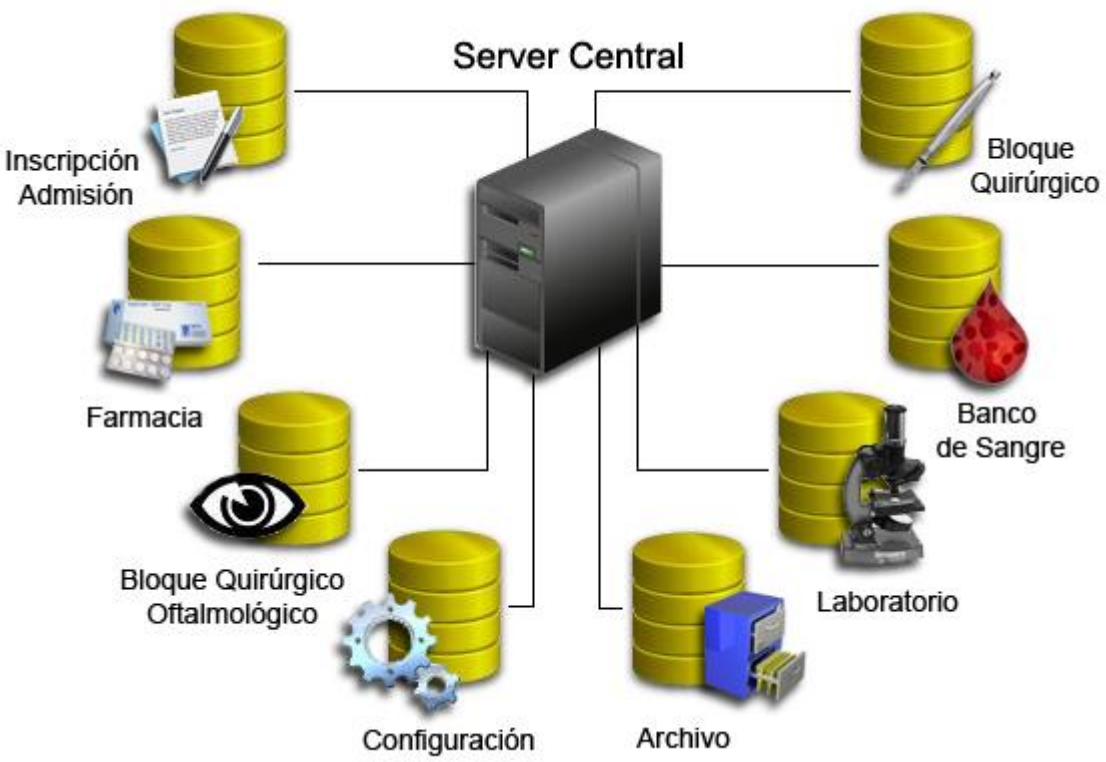
8. Pérez García, Reñiré. Impacto de la Informatización en la Sociedad Cubana. Ciencia, tecnología y sociedad, (abril 2005). Disponible en: http://www.monografias.com/trabajos24/informatizacion-cuba/informatizacion-cuba.shtml

9. SabiaMed. Disponible en: http://www.sabiamed.com

10. Sitio oficial de PostgreSQL. Disponible en: http://www.postgresql.org/

Anexo 1

Arquitectura central



API: Una API (del inglés *Application Programming Interface* - Interfaz de Programación de Aplicaciones) Una API representa un interfaz de comunicación entre componentes *software*.

Booch: Diagrama generado través de herramientas de modelado Booch. Método utilizado para describir un proceso iterativo, que entre sus ventajas tiene que está pensado para autocorregirse, para adoptar la adición de nuevos requisitos y para ser cooperativo. El nombre está otorgado en honor a su autor Grady Booch.

BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL, estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

B-tree(R-tree, Hash, GisT): Método de almacenamiento de índices.

Bugzilla: Es una herramienta basada en Web de Seguimiento de bugs (errores), (Bug Tracking System o BTS por sus siglas en inglés), originalmente desarrollada y usada por el proyecto Mozilla.

Certificados X.509: Es un estándar UIT-T (Unión Internacional de Telecomunicaciones) para infraestructuras de claves públicas. X.509 especifica, entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación.

Citrix: Es una compañía dedicada principalmente al desarrollo de *software* que ofrece un conjunto de productos que tratan de ofrecer un acceso más seguro a aplicaciones y contenidos.

COM: *Component Object Model*, es una plataforma de Microsoft para componentes de *software*. Esencialmente COM es una manera de implementar objetos neutral con respecto al lenguaje, de manera que pueden ser usados en entornos distintos de aquel en que fueron creados, a través de fronteras entre maquinas.

DDL: (*Data Definition Language*; Lenguaje de Definición de Datos) Es utilizado para describir todas las estructuras de información y los programas que se usan para construir, actualizar e introducir la información que contiene una base de datos.

Delphi: Es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual.

Utiliza como lenguaje de programación una versión moderna de Pascal llamada *Object Pascal*.

DML: (*Data Manipulation Language*; Lenguaje de manipulación de datos) Es utilizado para escribir programas que crean, actualizan y extraen información de las bases de datos. Siempre de acuerdo con las especificaciones y las normas de seguridad dictadas por al administrador.

Drupal: Es un sistema de administración de contenido para sitios Web. Es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web incluido en el producto.

GPL: La GPL (*General Public License* o Licencia Pública General) es una licencia creada por la *Free Software Foundation* y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

Hematología: Es la especialidad médica que se dedica al tratamiento de las enfermedades hematológicas. Se encarga del estudio e investigación de la sangre y los órganos hematopoyéticos (médula ósea, ganglios linfáticos, bazo, etc.). Comprende el estudio de la etiología, diagnóstico, tratamiento, pronóstico y prevención de las enfermedades de la sangre y órganos hemolinfoproductores.

HTML: Acrónimo inglés de *HyperText Markup Language*, que se traduce al español como Lenguaje de Marcas Hipertextuales. Es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

HTTP: El protocolo de transferencia de hipertexto (HTTP: *HyperText Transfer Protocol*) es el protocolo usado en cada transacción de la *World Wide Web* (www).

Inmunología: Es la parte de la biología que se ocupa del estudio del sistema inmune, entendiendo como tal al conjunto de órganos, tejidos y células que en los vertebrados tienen como función biológica el reconocer elementos extraños o ajenos dando una respuesta (respuesta inmune).

InnoDB: Tecnología de almacenamiento de datos de fuente abierta para MySQL, incluido como formato de tabla estándar en todas las distribuciones de MySQL AB a partir de las versiones 4.0. Su característica principal es que soporta transacciones de tipo ACID y bloqueo de registros e integridad referencial.

JDBC: Acrónimo de *Java Database Connectivity*, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

Kernel: (En lenguaje informático significa núcleo). Parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora. Es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

Linux: Es la denominación de un sistema operativo tipo-Unix. Es uno de los paradigmas más prominentes del *software* libre y del desarrollo del código abierto.

MaxDB: MaxDB es un sistema de administración de bases de datos para usarse como un repositorio de datos para las aplicaciones de SAP.

MediaWiki: Es un motor para *wikis* bajo licencia GPL, programado en PHP usando MySQL sobre Apache.

MER: Modelo de Entidad-Relación. Herramienta para el modelado de datos que expresa entidades relevantes para un sistema de información.

Microbiología: Es la ciencia encargada del estudio de los microorganismos, seres vivos pequeños, también conocidos como microbios. Es la rama de la biología dedicada a estudiar los organismos que son solo visibles a través del microscopio (virus, procariontes y eucariontes simples).

Microsoft SQL Server: Es un sistema de gestión de bases de datos relacionales (SGBD-R) basada en el lenguaje SQL.

Mozilla: La Fundación Mozilla es una organización sin ánimo de lucro, dedicada a la creación de *software* libre. Tiene como misión mantener la elección y la innovación en Internet.

ODBC: *Open Database Connectivity*: API de acceso a datos que soporta el acceso a cualquier fuente de datos para la cual exista un *driver* ODBC, el cual se encuadra dentro de los estándares ANSI e ISO para la Interfaz de llamadas de datos.

OLE: *Object Linking and Embedding:* Es un sistema de objeto distribuido y un protocolo desarrollado por Microsoft. Su uso principal es el manejo de documentos compuestos, pero también puede ser usado para transferir datos entre aplicaciones diferentes.

Pg_crypto: Librería que incluye PostgreSQL para la encriptación y almacenado seguro de datos en las Bases de Datos.

POO: Programación Orientada a Objetos

Química Clínica (o Bioquímica): Es la rama de la Química que estudia los seres vivos, especialmente de la estructura y función de sus componentes químicos específicos, como son las proteínas, carbohidratos, lípidos y ácidos nucleicos, además de otras pequeñas moléculas presentes en las células.

RPC: *Remote Procedure Call*: Una forma de comunicación entre aplicaciones que esconde la complejidad de la red utilizando un mecanismo de llamada de procedimientos ordinario. Es un proceso sincrónico firmemente acoplado.

RTF: Acrónimo inglés de *Rich Text Format* (Formato de Texto Rico), lenguaje de descripción desarrollado por Microsoft para intercambiar información entre programas multiplataforma de edición de texto.

RUP: *Rational Unified Process:* Es un proceso de desarrollo de *software* junto con el Lenguaje Unificado de Modelado (UML). Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

SAP: SAP AG, traducido del alemán al español; Sistemas, Aplicaciones y Productos, es el primer proveedor de aplicaciones de *software* empresarial en el mundo. Comercializa aplicaciones de *software* para soluciones integradas de negocios.

SQL: En inglés *Structured Query Language* (Lenguaje de Consultas Estructurado): Es el lenguaje que permite la comunicación con el Sistema Gestor de Bases de Datos.

SSL: *Secure Sockets Layer:* Es un protocolo criptográfico que proporciona comunicaciones seguras en Internet.

Sybase Adaptive Server: (ASA - *Sybase Adaptive Server Anywhere*) es un sistema administrador de bases de datos relacionales (RDBMS) de alto rendimiento.

TCP/IP: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), protocolos de red que se basan en Internet y que permite la transmisión de datos entre redes de computadoras.

Terminal Server: Los servicios de terminal son un componente de los sistemas operativos Windows que permite a un usuario acceder a las aplicaciones y datos almacenados en otro ordenador mediante un acceso por red.

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*): Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de *software*.

UNIX: Es un sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T. Familia de sistemas operativos que comparten criterios de diseño e interoperabilidad en común.

Valgrind: Es un conjunto de herramientas de Software Libre que ayuda en la depuración de problemas de memoria y rendimiento de programas.