Universidad de las Ciencias Informáticas

Facultad 6

Título: "LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del módulo Sección de Mejoramiento de la Calidad".

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autoras: Rosayda Valiente Mesa

Idelsis Castillo Pérez

Tutora: Ing. Elennis Díaz Laurencio

Co-Tutora: Lic. Miulkenia Navarro Reyes

Cuidad de la Habana, Junio 2008

"Año 50 de la Revolución"

"La posibilidad de realizar un sueño es lo que hace que la vida sea interesante"

Paulo Coelho

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 26 días del mes de Junio del año 2008.

_____ _____

Idelsis Castillo Pérez *Rosayda Valiente Mesa*

_____

Elennis Díaz Laurencio

Datos de Contacto

Ing. Elennis Díaz Laurencio
Universidad de las Ciencias Informáticas, Habana, Cuba
E-mail: ediaz@uci.cu

Lic. Miulkenia Navarro Reyes
Universidad de las Ciencias Informáticas, Habana, Cuba
E-mail: mnavarror@uci.cu

AGRADECIMIENTOS

Queremos agradecer primeramente a la Revolución Cubana por su grandeza y avances obtenidos en el campo de las tecnologías, la informática y las comunicaciones, especialmente a nuestro comandante Fidel Castro, por ser el creador de la Universidad de las Ciencias Informáticas y paradigma de las nuevas generaciones y de su pueblo.

Al consejo de dirección de la UCI y a los profesores, por ser un pilar fundamental en nuestro proceso de formación.

A nuestra tutora y co-tutora, por su apoyo, preocupación y aporte importante en este trabajo.

A los especialistas del CIGB por su valiosa contribución.

A nuestros compañeros de estudios con los que pasamos 5 años maravillosos de universidad.

A todos nuestros compañeros de proyecto en especial a Anabel, Jose y Denny, por su ayuda incondicional en todo momento.

A Roberto Acosta González por el apoyo prestado, la paciencia, la preocupación y los sabios consejos.

A nuestros padres y familiares que de una forma u otra han colaborado y apoyado incondicionalmente el desarrollo y culminación de esta tesis.

A todos, muchas gracias por haber estado para nosotras en algún momento de nuestras vidas.

DEDICATORIA

La vida, es la testigo muda de un largo camino en el que tropezamos muchas veces, y casi siempre ella misma nos tiende la mano amiga que ha de levantarnos. Lo que ayer parecía un itinerario angosto cubierto de esperanzas y añoranzas, hoy ha revolucionado de tal forma, que en mi memoria subsistirán por siempre inmaculados recuerdos de mis años universitarios. Ineludiblemente el crono ha marcado el indicio del fin de una etapa que ha dejado marcas imborrables en mi vida y paralelo a esto acuden a mi, sentimientos de nostalgia. Nostalgia por las personas que conocí y que tal vez no volveré a ver, nostalgia porque dejo atrás una parte de mi vida que no volveré a vivir. Aún no se si estoy preparada para enfrentar la realidad del mundo, pero eso lo he de comprobar con el decursar del tiempo.

Pudiera escribir los nombres de las personas que ayudaron a hacer mi sueño realidad, con el temor de olvidar mencionarlos a todos.

Primeramente le agradezco y le dedico mi trabajo a mis padres: María Oneida e Idelsides, por todo el amor, dedicación y empeño que han mostrado en todos estos años. Por ser un persevero ejemplo que siempre he de seguir. Por enseñarme y solidificar los valores necesarios para crecer espiritual y profesionalmente. Por sus consejos colmados de sabiduría y precisión. Por su ilimitada capacidad para entenderme. Quiero que sepan que son las personas que más quiero en el mundo y espero que estén muy orgullosos de mí. Gracias por existir.

A mis hermanos Osmel, Osmani y Osvaldo por apoyarme y estar siempre presentes; al tanto de mis estudios, de mis problemas. Viviendo mis momentos de alegría y también los de tristeza.

A mi prima Yoannia por su compresión y cariño. Porque siempre supe encontrar en ella una voz amiga y una mano dispuesta a ayudarme.

A mis amigas, más que amigas, hermanas Rosayda, Jenely y Marlien con las que he compartido risas y llantos, alegrías y tristezas. Debo agradecer a la UCI por haberlas conocido y contar con su preciada amistad.

A familiares y amigos en general que estuvieron al tanto de mis estudios y aportaron lo mejor de sí, para que pudiera culminarlos exitosamente.

A Deimer por su amor, cariño y apoyo en todo momento.

A las niñas del apartamento Yane, Evy, Yayi, Zoila, Aylin, Yune, Kenia, inseparables compañeras y amigas.

A todos muchas gracias.

Idelsis Castillo Pérez.

A mi madre Violeta G. Mesa Menzies por ser mi fuente de inspiración. Porque su fuerza y amor me han dirigido por la vida. Porque sus brazos siempre se abren cuando necesito un abrazo; su corazón sabe comprender cuando necesito una amiga y sus ojos sensibles se endurecen cuando necesito una lección. ¡¡¡Por hacer de mi lo que soy hoy…..muchas gracias!!!

A mi hermano Rassy por el simple hecho de existir y hacerme feliz. Por ser tan especial, por quererme mucho y cuidarme. ¡¡¡¡Estoy muy orgullosa de ti!!!!!

A mi abuelita por su comprensión y dedicación constante.

A Alejandro por ser como un padre, por su apoyo y confianza. Por estar siempre dispuesto a ayudarme, brindándome su amor y cariño. ¡¡¡Gracias por estar siempre para mí!!!

A Enier por su amor, eterna comprensión, constante estímulo e infinita ayuda. ¡¡¡Sabes cuanto te adoro!!!

A los familiares y amigos que siempre se preocuparon por mis estudios.

A mis amigas Marlien, Idelsis y Jenely, que han constituido fuente indispensable de sostén durante estos 5 años. ¡¡¡¡¡ De más está decirle que no habrá forma de olvidarlas nunca!!!!!

A las chicas del 67204: Yayi, Zoila, Yane, Evy, Aylin, Yune y Kenia pues con ellas compartí muy buenos momentos.

A Gleidy por siempre estar cuando la necesitaba.

¡¡¡¡¡¡¡A todos gracias por ser parte de una etapa inolvidable en mi vida!!!!!!!

Rosayda Valiente Mesa.

RESUMEN

En la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología, se gestiona a diario un gran volumen de información relacionada con los ensayos que se realizan en los distintos laboratorios. Esta información es de vital importancia en el control y aseguramiento de la calidad de las producciones del Centro. Para contribuir a la gestión y control de los datos que se generan, se está realizando un Sistema de Gestión de Información de los Laboratorios (LIMS, por sus siglas en inglés Laboratory Information Management System) para esta dirección, el cual no está ajeno a los problemas existentes en el almacenamiento de información. El presente trabajo aborda el desarrollo de la Base de Datos del módulo Sección de Mejoramiento de la Calidad, con el objetivo de almacenar la información referente a los procesos que se desarrollan, contribuyendo al manejo de los datos que se generan en este módulo.

Palabras Claves: Base de Datos, Modelo de Datos y Validación.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La instrumentación tecnológica es una prioridad en la comunicación de hoy en día, porque las tecnologías de la comunicación son la diferencia entre una civilización desarrollada y otra en vías de desarrollo.

Las Tecnologías de la Información y las Comunicaciones (TIC) en la actualidad ocupan un lugar fundamental en el desarrollo de la sociedad. El concepto de las TIC surge con la unión tecnológica de la electrónica, el software y las infraestructuras de las telecomunicaciones. Las TIC proporcionan herramientas que posibilitan encontrar soluciones novedosas ante los desafíos sociales actuales.

Cuba no está exenta a esta revolución tecnológica, por lo que en los últimos años ha emprendido el reto de la informatización de la sociedad, alcanzando resultados satisfactorios en áreas tan esenciales como la Educación, la Salud y la Investigación.

Los efectos de esta labor han llegado a centros importantes del país, que se informatizan exitosamente logrando grandes avances en el procesamiento de la información que generan. Entre ellos está el Centro de Ingeniería Genética y Biotecnología (CIGB) con una trayectoria de 20 años de investigación científica y la obtención de productos reconocidos a nivel mundial.

El trabajo realizado por el CIGB ha tenido gran impacto en la biomedicina, salud animal, mejoramiento vegetal y la bioindustria, ha desarrollado nuevas vacunas y fármacos para la salud humana que se encuentran actualmente en uso dentro del sistema de salud cubano, así como en diferentes países.[1]

Una de las máximas del CIGB es la calidad de los productos desarrollados y elaborados en el centro. Esta institución se caracteriza por su eficacia y seguridad, por tal motivo en la estructura del mismo existe la **Dirección de Calidad**.

Su objetivo se pone de manifiesto a través de los Departamentos de **Control de la Calidad y Aseguramiento de la Calidad.**

"El **Departamento de Control de la Calidad** tiene entre sus funciones fundamentales las relacionadas con el muestreo, las especificaciones, los ensayos y la evaluación de la calidad de los productos" [1] que se generan en el centro. Para el desempeño de las mismas, cuenta con la ayuda de dos grupos de trabajo y dos secciones:

- Grupo de Desarrollo.
- Grupo de Liberación Analítica.

- Sección biológica compuesta por cinco laboratorios:
    - Laboratorio de Ensayos Biológicos I.
    - Laboratorio de Ensayos Biológicos II.
    - Laboratorio de Biología Molecular.
    - Laboratorio de Microbiología.
    - Laboratorio de Inmunoquímica.
- Sección físico-química compuesta por tres laboratorios:
    - Laboratorio de Sistemas Críticos.
    - Laboratorio Análisis Químico.
    - Laboratorio de Cromatografía y Electroforesis.

"El **Departamento de Aseguramiento de la Calidad** garantiza que se lleven a cabo las acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que los productos y servicios satisfacen los requisitos de calidad establecidos. Vela por el cumplimiento de las Buenas Prácticas de Producción (BPP), Buenas Prácticas de Laboratorio (BPL) y Buenas Prácticas Clínicas (BPC). Este Departamento está compuesto por dos Secciones y dos grupos de trabajo:" [1]

- **Sección de Mejoramiento de la Calidad (SMC).**
- Sección de Inspección, Auditoría y Liberación de lotes.
    - Grupo de Inspección y Auditorias.
    - Grupo de Liberación de Lotes.
- Grupo de Documentación.
- Grupo de Metrología. (Ver Anexo No 1)

La Sección de Mejoramiento de la Calidad tiene como función principal el mejoramiento de la calidad de las producciones del Centro. Estos procesos generan grandes volúmenes diarios de información impresa, lo que dificulta el almacenamiento organizado, confiable y duradero de la misma.

Toda la documentación generada, en la mayoría de los casos, es revisada, supervisada y aprobada por las personas con rangos superiores, lo que en ocasiones puede provocar pérdida de tiempo considerable, cuando se necesita de un tiempo de respuesta mínimo, para la toma de decisiones.

Es importante señalar que la cantidad de información generada, es constantemente consultada por los trabajadores de la SMC en la obtención y generación de diversos reportes; al no estar la información digitalizada, resulta más compleja la búsqueda y el intercambio de la misma.

En aras de mitigar estos problemas, se realizaron una serie de visitas al centro y se comenzó un estudio sobre todos los procesos que allí se llevan a cabo, con el objetivo de determinar el estado real del flujo de información en el módulo Sección de Mejoramiento de la Calidad, así como su relación con los demás grupos del área de Calidad del CIGB y otros centros fuera del mismo.

El intercambio de documentos entre grupos y secciones, se hace personalmente; lo que puede ocasionar extravío y deterioro de la información, por lo que se hace necesario controlar el momento en que se realiza y los participantes.

Actualmente, el almacenamiento de toda la información del módulo está en formato duro, únicamente los datos más significativos se transcriben a formato digital, dificultando a los trabajadores el acceso y manejo de datos, por lo que se identifica como ***problema científico:*** ¿Cómo mejorar el proceso de almacenamiento de la información de la Sección de Mejoramiento de la Calidad en la Dirección de Calidad del CIGB?

El problema planteado se enmarca en el ***objeto de estudio***: El proceso de gestión de la información de la Sección de Mejoramiento de la Calidad en la Dirección de Calidad del CIGB.

El objeto delimita el ***campo de acción***: El proceso de almacenamiento de la información de la Sección de Mejoramiento de la Calidad en la Dirección de Calidad del CIGB.

Para dar solución al problema se define como ***objetivo:*** Desarrollar una Base de Datos para el módulo Sección de Mejoramiento de la Calidad del Sistema de Gestión de la Información de los Laboratorios en la Dirección de Calidad del CIGB.

A partir del análisis del objetivo general se derivaron los siguientes **objetivos específicos:**

- Diseñar el Modelo de Datos para la BD.
- Implementar las funciones de la BD.
- Validar teórica y funcionalmente el diseño de la BD.

Para lograr los objetivos propuestos, se realizarán las siguientes ***tareas***:

- Análisis de los procesos y la documentación generada en la Sección de Mejoramiento de la Calidad de la Dirección de Calidad del CIGB.

- Estudio de las tendencia, tecnologías, métodos y herramientas actuales de las bases de datos a nivel mundial.
- Estudio de las actividades y artefactos asociados al rol de Diseñador de BD propuestos por RUP.
- Estudio sobre la generación de la Capa de Mapeo Objeto-Relacional en Symfony.
- Diseño del diagrama de clases persistentes.
- Diseño del Modelo físico de datos.
- Análisis de la integridad de los datos en la BD.
- Análisis de redundancia de información en la BD.
- Normalización de la BD.
- Implementación de disparadores y procedimientos almacenados de la BD.
- Análisis de la seguridad de la BD.
- Análisis de la trazabilidad de la acciones.
- Búsqueda de herramientas para pruebas de carga intensiva.
- Análisis de optimización de consultas.

El trabajo consta de Introducción, 3 capítulos, conclusiones, recomendaciones, bibliografías y anexos.

En el Capítulo 1 **Fundamentación Teórica**: se brinda información referente a las BD en las aplicaciones LIMS. Se abordan varios aspectos de las bases de datos como su surgimiento y desarrollo, elementos, tipos de modelos y arquitectura. Se describe el Sistema Gestor de Base de Datos, así como la metodología y tecnologías a utilizar. Se especifica como se va a realizar el mapeo de objetos a la BD. Además, se detalla la herramienta CASE a emplear para la realización del diagrama de clases persistentes y el diseño del modelo físico. Se muestran las actividades y artefactos asociados al Rol de Diseñador de BD.

En el Capítulo 2 **Descripción y análisis de la solución propuesta**: se describe la solución propuesta basada en los requisitos funcionales y no funcionales del sistema. Se detalla la estrategia de integración de la solución con otros módulos. Se describe y fundamenta la arquitectura que se va a utilizar en el desarrollo de la BD. También, se muestra el modelo de objetos del negocio y el diagrama de clases del diseño como punto de partida para elaborar el modelo de datos. Además, se realiza una descripción de todas las clases y de las tablas que conforman la BD.

En el Capítulo 3 **Validación del diseño realizado**: se valida el diseño realizado teórica y funcionalmente, utilizando varios parámetros con el fin de obtener un diseño óptimo. Además, se realiza un análisis de optimización de consultas utilizando el framework Symfony.

Capítulo 1
Fundamentación Teórica

En este capítulo, para lograr una mejor comprensión de la necesidad de este trabajo, se brinda información referente a las BD en las aplicaciones LIMS. Se abordan varios aspectos de las bases de datos como su surgimiento y desarrollo, elementos, tipos de modelos y arquitectura. Se describe el Sistema Gestor de Base de Datos, así como la metodología y tecnologías a utilizar. Se especifica como se va a realizar el mapeo de objetos a la BD. Además, se detalla la herramienta CASE a emplear para la realización del diagrama de clases persistentes y el diseño del modelo físico. Se muestran las actividades y artefactos asociados al Rol de Diseñador de BD.

1.1 Proceso de gestión de la información en la SMC.

En el departamento de SMC se llevan a cabo varios procesos de forma simultánea, con el objetivo de comprobar y garantizar la calidad de las producciones del centro. Teniendo en cuenta como se desarrolla el flujo de información en el departamento se pueden obtener informes, programas, documentos, consultas y reportes que van a registrar todos los resultados referentes a los procesos que se realizan.

El Informe de Tendencias se realiza con el objetivo de analizar el comportamiento de las características de los diferentes productos en un período de tiempo determinado para, a partir de este informe, tomar medidas e investigar las causas reales o potenciales de los problemas. También sirve para mantener bajo control los diferentes procesos.

El documento Control de Cambios establece la política de cambios en los procesos y técnicas analíticas del CIGB garantizando el cumplimiento de las buenas prácticas de Fabricación y Laboratorio, para que cualquier variación con respecto a los procedimientos aprobados sea justificada y registrada y se valore su impacto en la calidad del producto y en el medio ambiente. Este procedimiento será aplicable a todos aquellos cambios planificados ya sean temporales o permanentes, que de una forma u otra puedan influir en la calidad, seguridad y eficacia del producto.

Los cambios pueden ocurrir en materiales, reactivos u otros insumos; en las etapas del proceso productivo o en los parámetros de operación establecidos.

También en el departamento se establecen los procedimientos necesarios para la elaboración y coordinación de los Programas de Monitoreo Ambiental (PMA) de las áreas limpias y de las diferentes producciones del CIGB. El PMA tiene como objetivo fundamental garantizar que las producciones del centro se realicen en ambientes adecuados y que las áreas limpias permanezcan dentro de los parámetros establecidos. Un área limpia es un local, con unas características ambientales especificadas y controladas en función de las partículas y la contaminación microbiológica. Es construida, mantenida y utilizada de forma tal que se disminuya la introducción, generación y retención de contaminantes, ya sean viables o no viables. El PMA se diseña y aplica para brindar evidencias de que la calidad del ambiente del área limpia se encuentra entre los límites especificados. La aplicación de este programa requiere de un análisis profundo del proceso y de las características del producto en cuestión.

El PMA incluye varios aspectos:

- Monitoreo al ambiente de las áreas limpias.
- Monitoreo de las superficies.
- Monitoreo del personal.
- Ensayo a los sistemas críticos utilizados en el proceso en cuestión.

El programa se elabora a partir de la clase correspondiente a cada área, previamente definida y plasmada en los planos oficiales de la planta en cuestión y en las especificaciones de áreas aprobadas. Este está compuesto por un conjunto de procedimientos generales que establecen su organización, desarrollo, métodos de muestreo y de ensayo, investigación de las desviaciones y acciones correctivas. Esta información general es complementada por los criterios específicos para el control ambiental de cada producción.

1.2 Bases de Datos en aplicaciones LIMS.

"Un LIMS proporciona un conjunto de herramientas basadas en Sistemas Informáticos que permiten la aplicación de técnicas de adquisición y gestión avanzada de la información producida en el laboratorio." [1]

La utilización de los **LIMS** ha favorecido en gran medida la manipulación de los datos que se obtienen en los laboratorios. Entre los beneficios más comunes que tiene su utilización se pueden mencionar:

- Aumento de la cantidad de información disponible y requisitos de manipulación de la misma.
- Aseguramiento de la calidad.

- Integridad de la información.
- Mejoras en el procesamiento de la información y en la productividad.
- Una integración más rápida con los instrumentos del laboratorio y las aplicaciones de la empresa.

Es por ello que muchas empresas se han dedicado a la fabricación de estos sistemas para el manejo en particular de sus propias necesidades. Cada Sistema de Gestión de la Información de los Laboratorios tiene asociada una base de datos, con el objetivo de almacenar de manera eficiente la información, que generan los procesos que se desarrollan en cada grupo de trabajo. La utilización de la BD, facilita el manejo de datos, dando la posibilidad de buscar, obtener y actualizar una información determinada.

Muchos de los LIMS que se utilizan actualmente a nivel mundial, cuentan con una base de datos, entre ellos podemos citar:

La Suite Pharma LIMS está destinada a la industria farmacéutica y cuenta con un inventario construido sobre bases de datos de moléculas. Para el almacenamiento de la información en las aplicaciones farmacéuticas se tiene en cuenta la magnitud de la misma. Para aquellas organizaciones de gran tamaño y con un gran número de usuarios, la solución idónea es utilizar Oracle como gestor de BD. En el caso de organizaciones de tamaño mediano, que buscan menores costos en recursos informáticos y de mantenimiento, es recomendable utilizar SQL Server pues ofrece una solución completa y asequible en gestión del conocimiento, bases de datos químicos, informática química y bioinformática aplicada. Las soluciones desktop sobre bases de datos MSDE son idóneas para un número reducido de usuarios que desean organizar, almacenar y buscar entre sus datos directamente desde su PC. [2]

LABWORKS LIMS coordina la gestión de datos de los laboratorios y la velocidad de flujo de la información. Los datos que se almacenan en este LIMS, están protegidos contra sobre-escritura accidental, fallos del sistema o manipulaciones intencionadas; debido a que LABWORKS LIMS registra datos, métodos y resultados brutos en bases de datos seguras con protocolos de acceso definidos por el administrador. Además tiene sistemas de copia de seguridad y de recuperación en caso de desastre, asegurando que los datos del sistema estén protegidos y se puedan recuperar.

Matrix LIMS es compatible con una variedad de bases de datos comerciales, incluyendo Oracle y Microsoft SQL Server, para adaptarse al estándar corporativo o a las preferencias del usuario, con posibilidad incluso de configuración con múltiples grupos de bases de datos.

1.3 Surgimiento y desarrollo de las Bases de Datos.

El término *Base de Datos* fue escuchado por primera vez en 1963, en un simposio celebrado en California. [3] Al principio, los lenguajes y las instrucciones de máquina eran muy similares, lo que producía un modelo de programación orientado por procesos. Inicialmente, los programas ejecutaban las tareas y nunca las escribían en un dispositivo de almacenamiento. En esta etapa, uno de los pocos elementos que se almacenaban era el propio programa. Sin embargo, los programadores se dieron cuenta del valor de registrar los resultados. La grabación de los resultados del programa aumentó con el advenimiento del almacenamiento en discos magnéticos rotatorios, lo que ofreció la posibilidad del acceso aleatorio a grandes cantidades de datos almacenados.

Dada la gran complejidad que iban alcanzando los programas y el volumen de datos que generaban, se requería de un almacenamiento que garantizara un cierto número de condiciones y que permitiera operaciones complejas. Además, cada usuario que accediera a los datos debía tener su trabajo protegido de las operaciones que hicieran el resto de los usuarios.

En aras de solucionar esta problemática, surgen las Bases de Datos, que son un conjunto exhaustivo de datos estructurados, organizados independientemente de su utilización. Una BD se diseña, construye y puebla con datos para un propósito específico y está dirigida a un grupo determinado de usuarios. Debe dar la posibilidad de buscar, obtener y actualizar los datos siempre que sea necesario.

Las bases de datos han evolucionado desde sistemas de archivos rudimentarios hasta sistemas gestores de complejas estructuras de datos que ofrecen un gran número de posibilidades.

1.4 Base de Datos y sus elementos.

Las BD y sus tecnologías tienen un impacto decisivo con el creciente desarrollo de las comunicaciones. Estas desempeñan un papel crucial en casi todas las áreas como los negocios, la ingeniería, la medicina, el derecho, la educación y la bibliotecología.

Las bases de datos tienen muchos usos: facilitan el almacenamiento de grandes cantidades de información; permiten la recuperación rápida y flexible de los datos; se puede organizar y reorganizar la información, así como imprimirla o distribuirla en formas diversas.

Los sistemas de bases de datos presentan numerosas ventajas que hacen más factible su uso, entre las que se pueden citar:

- Control sobre la redundancia de datos: Al estar todos los ficheros integrados, no se almacenan varias copias de los mismos datos. en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos
- Consistencia de datos: Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente.
- Compartición de datos: La base de datos pertenece a una empresa determinada y puede ser compartida por todos los usuarios que estén autorizados.
- Mejora en la integridad de datos: La integridad de la BD se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones.
- Mejora en la seguridad: Se manifiesta a través de la protección de la base de datos frente a usuarios no autorizados.
- Mejora en la accesibilidad a los datos: Muchos gestores de bases de datos proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

Una base de datos está formada por:

Datos: Están almacenados de acuerdo a la estructura externa y van a ser procesados para convertirse en información. Deben ser integrados (posibilitando que en la unión de los archivos que forman el sistema no exista redundancia de datos) y compartidos (disímiles usuarios puedan acceder a los mismos datos con fines diferentes).

Hardware: Es el soporte físico que permite almacenar la información de la base de datos, así como los dispositivos periféricos (unidad de control, canales de comunicación, entre otros).

Software: Es el que permite trabajar y gestionar la BD de la forma más eficiente. El Sistema Gestor de Bases de Datos (SGBD) es el encargado de gestionar la BD, por lo tanto todas las operaciones que se realicen sobre las mismas han de pasar por el SGBD.

Usuarios: Una base de datos típica conlleva a la existencia de tres tipos de usuario con relación a su diseño, desarrollo y uso:

- El administrador de bases de datos: Es el usuario más importante, pues se encarga de diseñar y modificar la estructura de la BD.
- El desarrollador de aplicaciones (programador): Se encarga de diseñar y programar las aplicaciones necesarias para la utilización de la BD, realizando las peticiones pertinentes al SGBD.
- Los usuarios finales: Son las personas que se dedican a trabajar sobre los datos almacenados en la BD. Consultan y editan dichos datos mediante un lenguaje de consulta de alto nivel. [3]

1.5 Modelos de Bases de Datos.

Existen distintos modos de organizar la información y representar las relaciones entre los datos en una BD. Los sistemas administradores de bases de datos convencionales usan uno de los cinco modelos lógicos para hacer seguimiento de las entidades, atributos y relaciones. Estos no son unidades físicas sino abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos y conceptos matemáticos. Los principales modelos lógicos son el jerárquico, de redes, el relacional, el orientado a objeto y el objeto-relacional. Cada uno de ellos tiene ciertas ventajas de procesamiento y de negocio.

Modelo Jerárquico:

En este modelo los datos se organizan en una forma similar a un árbol, en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padre es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos, por lo que actualmente no es muy utilizado.

Modelo Red:

Este es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es que se permite que un mismo nodo tenga varios padres.

Es una gran mejora con respecto al modelo jerárquico, pues ofrece una solución eficiente al problema de redundancia de datos; pero, aún así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales; dificultando su uso.

Modelo Relacional:

El modelo relacional se centra en el uso de relaciones, las cuales podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia. La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

Las bases de datos relacionales están estructuradas por una o más tablas relacionadas que contienen la información de una forma organizada. Generalmente contendrán muchas tablas y una tabla sólo contiene un número fijo de campos, en dependencia de la información almacenada. El orden de los campos de una tabla no esta determinado y para cada campo existe un conjunto de valores posible.

El modelo relacional incluye aspectos concernientes a los datos, ellos son:

- La estructura, que permite representar la información que nos interesa del mundo real.
- La manipulación, que engloba las operaciones de actualización y consultas de los datos.
- La integridad, que es posible mediante el establecimiento de reglas de integridad, o sea, condiciones que los datos deben cumplir.

Modelo Orientado a Objetos:

Este modelo trata de almacenar en la BD los objetos completos, o sea, estado y comportamiento. Una base de datos orientada a objetos incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulamiento**: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia**: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo**: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En el modelo orientado a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la BD. Una operación se especifica en dos partes:

- La interfaz: Incluye el nombre de la operación y los tipos de datos de sus argumentos.
- La implementación: Se especifica separadamente y puede modificarse sin afectar la interfaz.

Modelo Objeto-Relacional

Este modelo surge como una respuesta a la incorporación de la tecnología de objetos en las bases de datos relacionales permitiendo el tratamiento de datos y relaciones complejas. Es compatible con la tecnología relacional y tiene un mejor soporte para aplicaciones voluminosas. Se combinan las nociones de extensibilidad, orientación por objetos, y en algunos casos hasta tablas anidadas. Soporta datos y consultas complejas.

El modelo de base de datos seleccionado es el Relacional, porque es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente, garantizando un alto grado de independencia de los datos. Además, brinda una serie de herramientas para evitar la duplicidad de registros, a través de campos claves o llaves; garantiza la integridad referencial, así al eliminar un registro elimina todos los registros relacionados dependientes y favorece la normalización de la BD. Los lenguajes matemáticos sobre los que se asienta este modelo, el álgebra y el cálculo relacionales, aportan un sistema de acceso y consultas orientado al conjunto.

1.6 Arquitectura en los Sistemas de BD.

Hay tres características importantes inherentes a los sistemas de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos. El comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. En esta arquitectura, el esquema de una base de datos se define en tres niveles de abstracción distintos:

1. En el nivel interno se describe la estructura física de la BD mediante un *esquema interno,* en el que se describen todos los detalles para el almacenamiento de la BD así como los métodos de acceso.
2. En el nivel conceptual se describe la estructura de toda la BD para una comunidad de usuarios, mediante un *esquema conceptual*. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de

los usuarios y restricciones. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar el esquema.

3. En el nivel externo se describen varios *esquemas externos o vistas de usuario*. Cada esquema externo describe la parte de la BD que interesa a un grupo de usuarios determinados y oculta a ese grupo el resto de la BD. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas. (Ver Anexo No 2)

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que se puede definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se pueden definir dos tipos de independencia de datos:

- La independencia lógica es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla.
- La independencia física es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de alcanzar que la independencia lógica.

La arquitectura de tres niveles puede facilitar la obtención de la verdadera independencia de datos, tanto física como lógica. Sin embargo, los dos niveles de correspondencia implican un gasto extra durante la ejecución de una consulta o de un programa, lo cual reduce la eficiencia del Sistema Gestor de Bases de Datos (SGBD). Es por esto que muy pocos SGBD han implementado esta arquitectura completa.

1.7 Sistemas Gestores de Base de Datos.

Los sistemas de gestión de base de datos están dedicados a servir de interfaz entre la BD, el usuario y las aplicaciones que la utilizan. Se componen de un lenguaje de definición de datos (DDL), de un lenguaje de manipulación de datos (DML) y de un lenguaje de consulta (SQL).

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, posibilitando un buen manejo de datos.

Los SGBD brindan:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad y protección de la información, de modificaciones, inclusiones y consultas.
- Comprobación de información en el momento de introducir la misma.
- Integridad referencial el terminar los registros.

Aunque presentan algunos inconvenientes:

- El costo de actualización del hardware y software son muy elevados.
- Generan campos vacíos en exceso.
- Requieren una gran cantidad de espacio en disco y de memoria para trabajar de forma eficiente.
- Vulnerable a los fallos. Al estar todo centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse.

1.7.1 MySQL.

MySQL es un sistema de gestión de base de datos multihilo, multiusuario y relacional, lo que permite velocidad y flexibilidad. Se compone de una gran variedad de interfaces de programación (APIs), varios programas clientes y bibliotecas, herramientas administrativas, y un servidor SQL. MySQL es muy utilizado en aplicaciones web, su popularidad está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

Figura 1.1 SGBD MySQL.

Es un software de fuente abierta, lo que significa que es posible para cualquier persona usar y modificar el código fuente sin pagar, en dependencia de sus necesidades. MySQL usa el GPL (GNU General Public License) para definir que puede hacerse y que no con el software en diferentes situaciones.

Las características implementadas por MySQL son:

- Múltiples motores de almacenamiento, permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, con el objetivo de incrementar el número de transacciones por segundo.

Las ventajas que ofrece MySQL, como gestor de base de datos son:

- Mayor rendimiento: mayor velocidad al conectar con el servidor.
- Mejores utilidades de administración (backup, recuperación de errores, etc.).
- Integración perfecta con PHP.
- Sin límites en los tamaños de los registros
- Mejor control de acceso de usuarios, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos.
- Es muy rápido, confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños.
- Tiene un conjunto muy práctico de características desarrolladas en cooperación muy cercana con los usuarios.
- Ofrece un rico y muy útil conjunto de funciones. La conectividad, velocidad y seguridad hace de MySQL altamente conveniente para acceder a bases de datos en Internet.
- Consume muy pocos recursos, tanto de CPU como de memoria. Puede trabajar en distintas plataformas y sistemas operativos distintos. Además es multiproceso, es decir, puede usar varias CPU si estas están disponibles.
- Recuperación automática ante fallas. Si MySQL se cuelga o se da de baja de una forma anormal, no suele perder información ni corromper los datos, además InnoDB automáticamente completará las transacciones que quedaron incompletas.
- Integridad referencial: se pueden definir llaves foráneas entre tablas InnoDB relacionadas para asegurarse de que un registro no puede ser eliminado de una tabla si aún está siendo referenciado por otra tabla.

Inconvenientes:

- No soporta roll-backs ni subconsultas.

1.7.2 Oracle.

Oracle es un sistema de gestión de base de datos relacional, fabricado por Oracle Corporation. Se basa en la tecnología cliente/servidor. Se considera como uno de los SGBD más completos, destacando su soporte de transacciones, estabilidad, escalabilidad y el ser multiplataforma.

Ha incorporado en su sistema el modelo objeto-relacional, garantizando al mismo tiempo la compatibilidad con el tradicional modelo relacional de datos; ofreciendo así un servidor de bases de datos híbrido.

Un SGBD Oracle está compuesto por tres partes principales, que son:

- El kernel de Oracle.
- Las instancias del Sistema de Base de Datos.
- Los archivos relacionados al sistema de Base de Datos.

Oracle corre en computadoras personales (PC), microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo. Soporta 26 idiomas, corre automáticamente en más de 80 arquitecturas de hardware y software distintos, sin tener la necesidad de cambiar una sola línea de código. Esto se debe a que más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas operativos.

Ventajas:

- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Incluye mejoras de rendimiento y de utilización de recursos.
- Bloqueo y concurrencia: el lector verá los datos tal y como estaban antes de que el que escribe comience a cambiarlos (hasta que éste haga commit).
- Disponibilidad: soporta ambientes de clúster en modo activo-pasivo, es decir, que un solo nodo utiliza la base de datos mientras el(los) otro(s) nodo(s) está(n) pendiente(s) de entrar en funcionamiento en el momento que el servidor primario tenga una falla. Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.

Desventajas

- Los costos de soporte técnico y mantenimiento son elevados, por lo que es un producto que por lo general se utiliza en empresas muy grandes y multinacionales.
- Vulnerabilidades en la seguridad de la plataforma, por lo que se hace necesario aplicar parches de seguridad.

1.7.3 PostgreSQL.

PostgreSQL es un SGBD de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase. Es un servidor de BD relacional orientada a objetos de software libre, liberado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo.

Algunas de sus principales características son:

- Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión) 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.

- Disparadores (triggers).

Un disparador o trigger se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

- Vistas.
- Integridad transaccional.
- Herencia de tablas.

Ventajas:

- Extensible.

El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales.

- Diseñado para ambientes de alto volumen.

PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes.

- Herramientas gráficas de diseño y administración de bases de datos.

Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect).

- Multiplataforma.
- Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.
- Soporta transacciones y desde la versión 7.0, llaves foráneas con comprobaciones de integridad referencial.
- Tiene mejor soporte para triggers y procedimientos en el servidor.
- Soporta un subconjunto de SQL92 mayor que el que soporta MySQL. Además, tiene ciertas características orientadas a objetos.
- Usa una arquitectura cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectarse a PostgreSQL.
- Tiene soporte para lenguajes procedurales internos.
- Su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

Inconvenientes:

- Consume más recursos y carga más el sistema.
- Límite del tamaño de cada fila de las tablas a 8k. Se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento.

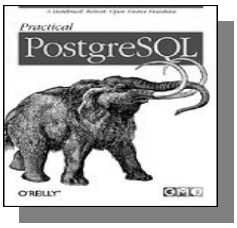


Figura 1.2 SGBD PostgreSQL.

Para el desarrollo del trabajo se utilizará PostgreSQL 8.2, pues es la tecnología definida por el proyecto, debido a que es un motor de base de datos avanzado y de código abierto, es decir, sus

potencialidades están en constante perfeccionamiento, permitiendo su uso y distribución sin costo. Además, soporta consultas complejas incluyendo subconsultas, integridad referencial, triggers, vista, integridad transaccional y control de versiones concurrentes. Es uno de los SGBD más potentes y multiplataforma, que cuenta con una amplia gama de características, por lo que está calificado como uno de los gestores de base de datos de alto nivel, más usado en la actualidad.

1.8 Mapeo de Objetos a Bases de Datos.

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. También facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Para la realización del LIMS de la Dirección de Calidad del CIGB se utilizará Symfony 1.0.11 que es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web. Symfony ofrece un sinnúmero de ventajas, por ejemplo: separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP 5. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft y se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows.

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo Vista Controlador (MVC), que está formado por tres niveles: (Ver Anexo No 3)

▪ El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

▪ La vista transforma el modelo en una página web que permite al usuario interactuar con ella.

▪ El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. [5]

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos utilizado por la aplicación.

La capa del modelo que es la que se encarga de la manipulación de los datos, puede dividirse en una capa de acceso a los datos y otra capa de abstracción de la BD, de manera que las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de un gestor de BD en específico, sino que utilizan otras funciones, implementadas en la capa de abstracción para realizar las consultas, incrementando la portabilidad del sistema.

Las bases de datos siguen una estructura relacional. Symfony, por el contrario accede a las BD desde un contexto orientado a objetos. Por este motivo, es necesaria una interfaz que traduzca la lógica de los objetos a la lógica relacional. Esta interfaz se denomina “mapeo de objetos a bases de datos” (ORM, de sus siglas en inglés “object-relational mapping”). Un ORM consiste en una serie de objetos que permiten acceder a los datos y que contienen en su interior cierta lógica de negocio.

La capa de abstracción ORM evita utilizar una sintaxis específica de un sistema de BD concreto, permite encapsular la lógica de los datos y añadir métodos accesores en los objetos que no tienen relación directa con una tabla.

Para traducir el modelo relacional de las base de datos a un modelo de objetos de datos se necesita una descripción del modelo relacional, que se llama “esquema” (schema). La sintaxis que utiliza Symfony para definir los esquemas hace uso del formato YAML, aunque también puede trabajar con el formato nativo de los esquemas en Propel, que está basado en XML. Este esquema permite construir las clases del modelo que necesita la capa del ORM.

Symfony utiliza también la capa de abstracción Creole que exige utilizar una sintaxis específica para las consultas y a cambio optimiza y adapta el lenguaje SQL a la base de datos concreta que se está utilizando, evitando de esta forma reescribir parte de las consultas en caso de que se necesite cambiar el gestor de bases de datos.

1.9 Proceso Unificado de Desarrollo (Rational Unified Process, RUP).

El Proceso Unificado de Desarrollo es un proceso bien definido, estructurado y adaptable a las características y necesidades de cada proyecto específico. La definición de este proceso está dada por tres características fundamentales: dirigido por casos de uso, iterativo e incremental y centrado en la arquitectura.

Que RUP esté **dirigido por casos de uso** significa que el proceso de desarrollo sigue una trayectoria a través de flujos de trabajos generados por casos de uso. Los casos de uso describen la funcionalidad del sistema, en términos de su importancia para el usuario.

Que RUP sea un **proceso iterativo e incremental** posibilita, que se pueda dividir el trabajo en partes más pequeñas o en mini proyectos, permitiendo el equilibrio entre casos de uso y arquitectura durante cada mini proyecto. Cada mini proyecto se puede ver como una iteración de la cual se obtiene un incremento, provocando un aumento del producto.

Que RUP esté **centrado en la arquitectura** facilita la visión del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo. La arquitectura describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

1.9.1 Lenguaje Unificado de Modelado.

El Proceso Unificado de Desarrollo utiliza el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés Unified Modeling Language) como leguaje de notación.

"**UML** es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software." [6]

Es válido subrayar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Un lenguaje de modelado contiene vistas, diagramas, símbolos o elementos de modelo y un conjunto de mecanismos generales o reglas que indican cómo utilizar los elementos. (Ver Anexo No. 4)

Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo.
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

1.9.2 Actividades y artefactos del rol Diseñador de BD.

"Un proceso de desarrollo de software define quién hace qué, cómo y cuándo. RUP define cuatro elementos: los roles, que responden a la pregunta ¿Quien?, las actividades que responden a la

pregunta ¿Cómo?, los productos, que responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas que responden a la pregunta ¿Cuándo?" [7]

Los trabajadores definen el comportamiento y las responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Una actividad es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

El diseñador de base de datos es el responsable de la definición de los detalles del diseño de la base de datos a partir de las clases del diseño y las realizaciones de casos de uso del diseño, incluyendo para la realización de esta actividad: tablas, índices, vistas, restricciones, disparadores y procedimientos almacenados.

Para la realización de esta actividad el diseñador de base de datos debe realizar los siguientes pasos:

- Desarrollar el modelo lógico de la BD.
- Desarrollar el diseño físico de la BD.
    - Crear los elementos iniciales del diseño físico de la BD.
    - Definir las tablas de referencia.
    - Crear clave primaria y restricciones de integridad.
    - Definir las reglas de integridad referencial y de la información.
    - Normalizar el diseño de la BD para su optimización.
    - Optimizar el acceso a los datos.
    - Definir las características de almacenamiento.
    - Diseño de procedimientos almacenados.
- Evaluar los resultados.

Desarrollar el modelo lógico de la BD: El modelo lógico del diseño de la BD, tiene como propósito proveer una vista lógica de los datos llaves de las entidades y sus relaciones, que son independientes de cualquier software o implementación de la base de datos. Para desarrollar el modelo lógico se necesita primeramente identificar y realizar el diagrama de clases persistentes.

Desarrollar el diseño físico de la BD: El diseño físico de la base de datos incluye elementos del modelo, o sea, tablas, vistas y procedimientos almacenados que representan el diseño subyacente de almacenamiento de datos. Los elementos del modelo constituyen el Modelo de Datos Físico de la base de datos y está contenido en el artefacto Modelo de Datos.

Revisar los resultados: Verifica la calidad e integridad del Modelo de Datos. El diseñador de la base de datos regularmente debe revisar la estructura implementada para asegurar que el Modelo de Datos es consistente con cualquier cambio que se ha hecho directamente en la base de datos.

El alcance de las actividades realizadas por el diseñador de base de datos varía dependiendo del tamaño y la complejidad del esfuerzo de desarrollo de la aplicación y el tipo de mecanismo de almacenamiento de datos persistentes usados para el proyecto.

El diseñador de base de datos debe tener los conocimientos básicos sólidos sobre:

- Modelado de datos.
- Diseño de base de datos.
- Técnicas de análisis y diseño orientado a objetos.
- Arquitectura de sistema.
- Administración de base de datos.
- Comprensión del entorno y lenguaje de implementación.

Una particularidad del RUP es que, en cada ciclo de iteración, se hace exigente el uso de artefactos para alcanzar un grado de certificación en el desarrollo del software.

Un artefacto, es un fragmento de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los artefactos son los resultados tangibles del proyecto que se van creando y usando hasta obtener el producto final.

El artefacto resultante de diseñar una BD es el Modelo de Datos (Ver Anexo No 5)**,** que describe la representación lógica y física de los datos persistentes. Este se puede obtener de varias formas: a partir de un modelo de objetos en el que se incluirán las clases que se marquen como persistentes, de las clases del diseño, o sino, mediante ingeniería inversa a partir de una BD existente.

En caso de que la aplicación utilice un sistema gestor de base de datos relacional, el modelo de datos también puede incluir elementos para restricciones, disparadores y procedimientos almacenados; que definen la interacción de la aplicación con los componentes del sistema gestor de base de datos.

1.10 Herramientas CASE.

Las Herramientas CASE (por sus siglas en inglés Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.

Estas herramientas ayudan en el ciclo de vida de desarrollo del software en tareas como: el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores.

El empleo de las herramientas CASE es factible ya que permiten:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y costo de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.10.1 Visual Paradigm.

Como herramienta CASE, para la realización del diagrama de clases persistentes y el modelo de datos, se utilizará Visual Paradigm for UML 6.1 Enterprise Edition, que emplea UML como lenguaje de modelado. Esta herramienta soporta el ciclo de vida completo del desarrollo de software. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Figura 1.3 Visual Paradigm for UML.

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0.
- 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 directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

1.11 Herramientas de desarrollo.

1.11.1 SQL Manager 2005 for PostgreSQL.

Se utiliza EMS SQL Manager 2005 for PostgreSQL ya que es una aplicación de alto desempeño para la administración y desarrollo de los servidores de BD para PostgreSQL. El programa trabaja con cualquier versión de PostgreSQL superior a la 8.0 y soporta las últimas características de PostgreSQL, incluyendo espacios de tablas así como nombres de argumentos en funciones. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo. [8]

Características:

- Soporte completo para PostgreSQL hasta la versión 8.1.
- Nueva interfaz de usuario gráfica de tecnología avanzada.
- Rápida gestión de datos y navegación.
- Administración fácil de todos los objetos de PostgreSQL.
- Herramientas de manipulación para datos anticipados.
- Eficaz dirección de seguridad.
- Acceso al Servidor PostgreSQL por protocolo http.
- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento.

1.11.2 EMS Data Generator 2005 for PostgreSQL.

EMS Data Generator 2005 for PostgreSQL es una poderosa herramienta para generar datos de prueba a las tablas de las bases de datos PostgreSQL. La aplicación asistente permite definir tablas y campos

para generar datos, configurar valores de rangos, generar campos de tipo carácter por máscara, cargar valores desde archivos para campos de tipo objetos binarios, obtener listas de valores desde consultas SQL y muchas otras características para generar datos de prueba de manera simple y directa. También provee una aplicación consola, que permite generar datos de un clic usando plantillas de generación. [9]

Características:

- Amigable interfaz asistente de usuario.
- Generación de datos a varias tablas desde diferentes bases de datos en un solo ordenador anfitrión (host).
- Soporte para todos los tipos de datos de PostgreSQL, incluyendo los tipos: vector, dirección de red y geométricos.
- Capacidad para usar resultados de consultas SQL como lista de valores para la generación de datos.
- Control automático sobre integridad referencial para la generación de datos a tablas vinculadas.
- Amplia variedad de generación de parámetros para cada tipo de campo.
- Capacidad para configurar valores nulos para ciertos casos.
- Genera datos de diferentes formas para cada campo, ya sea desde una lista de valores predefinida, valores aleatorios haciendo uso de caracteres previamente definidos o valores incrementales.

Conclusiones.

En este capítulo se analizó el proceso de desarrollo de los Sistemas de Gestión de la Información de los Laboratorios, así como los distintos gestores de base de datos. Además se presentaron las actividades y artefactos que realiza el diseñador de base de datos en el flujo de trabajo de análisis y diseño, según la metodología de desarrollo de software definida, así como las distintas herramientas y tecnologías a utilizar para la obtención de una BD que cumpla con todos los requerimientos exigidos, además de ser flexible a cambios.

Al finalizar el capítulo se arriba a las siguientes conclusiones:

- Utilizar el modelo de base de datos relacional.
- Utilizar el framework Symfony para la implementación de la capa de acceso a datos.

- Utilizar la metodología RUP para el desarrollo de la BD del módulo Sección de Mejoramiento de la Calidad. Por consiguiente el lenguaje de modelado será UML, porque implementa una documentación común para todos los desarrolladores.
- La herramienta CASE a utilizar será Visual Paradigm, para la realización del diagrama de clases persistentes y el modelo de datos.
- Utilizar PostgreSQL como gestor de base de datos, empleando para el desarrollo y administración de la BD: EMS SQL Manager 2005 for PostgreSQL y para la generación de datos de prueba: EMS Data Generador 2005 for PostgreSQL.

Capítulo 2
Descripción y Análisis de la solución propuesta

En este capítulo, para continuar con el desarrollo del trabajo, se describe la solución propuesta basada en los requisitos funcionales y no funcionales del sistema. Se detalla la estrategia de integración de la solución con otros módulos. Se describe y fundamenta la arquitectura que se va a utilizar en el desarrollo de la BD. También se muestra el modelo de objetos de negocio y el diagrama de clases del diseño como punto de partida para elaborar el modelo de datos. Además, se realiza una descripción de todas las clases y de las tablas que conforman la BD.

2.1 Requisitos Funcionales y No Funcionales.

Lograr una comunicación efectiva entre los usuarios y el equipo de proyecto, con el objetivo de llegar a un entendimiento de lo que se necesita, es la clave del éxito en la producción de un software. Muchas aplicaciones han fallado (no se culminaron o no se usaron) porque existieron incongruencias entre lo que el usuario quería, lo que realmente necesitaba, lo que interpretaba cada miembro del equipo de proyecto y lo que realmente se obtiene.

La definición de las necesidades del sistema es un proceso complicado, pues hay que identificar los requisitos del sistema, de modo que cumplan las expectativas del cliente. Los requerimientos deben ser verificados y validados para evitar errores que pueden resultar altamente costosos en una etapa posterior del desarrollo del software.

Los requisitos se pueden clasificar en **funcionales** que son capacidades o condiciones que el sistema debe cumplir y los **no funcionales** que son propiedades o cualidades que el producto debe tener.

2.1.1 Requisitos Funcionales.

En la fase de levantamiento de requisitos los analistas identificaron una serie de funcionalidades con la que debía cumplir el sistema. A continuación se muestran los requisitos que hasta ahora han sido aprobados en la SMC.

CUS Gestionar Registro de Control de Cambios (SIC-0816)

1.1. Crear nuevo Registro de Control de Cambios.

1.2. Registrar datos en el Registro de Control de Cambios.

1.3. Modificar datos del Registro de Control de Cambios.

1.3.1. Registrar traza.

1.4. Generar reporte del Registro de Control de Cambios.

CUS Gestionar Reporte de Límites (SIC-0963)

2.1. Crear nuevo Reporte de Límites.

2.2. Registrar datos en el Reporte de Límites.

2.3. Modificar datos en Reporte de Límites.

2.3.1. Registrar traza.

2.4. Buscar y visualizar Reporte de Límites.

2.5 Generar reporte de Reporte de Límites.

CUS Gestionar Programa de Monitoreo Ambiental (SIC-0098)

3.1. Crear nuevo Programa de Monitoreo Ambiental.

3.2. Modificar datos del Programa de Monitoreo Ambiental.

3.2.1. Registrar traza.

3.3. Buscar y visualizar Programa de Monitoreo Ambiental.

3.4. Generar reporte del Programa de Monitoreo Ambiental.

CUS Gestionar Seguimiento de Acciones Correctivas (SIC-0858)

4.1. Crear nuevo registro para el Seguimiento de Acciones Correctivas.

4.2. Registrar datos en el registro para el Seguimiento de Acciones Correctivas.

4.3. Modificar datos del Seguimiento de Acciones Correctivas.

4.3.1. Registrar traza.

4.4. Buscar y visualizar Seguimiento de Acciones Correctivas.

4.5. Generar reporte del Seguimiento de Acciones Correctivas (en un período de tiempo dado).

CUS Gestionar Programa de Informes de Tendencias (SIC-0919)

5.1. Crear nuevo Programa de Informes de Tendencias.

5.2. Registrar datos en el Programa de Informes de Tendencias.

5.3. Modificar datos del Programa de Informes de Tendencias.

5.3.1. Registrar traza.

5.4. Buscar y visualizar Programa de Informes de Tendencias.

5.5. Generar reporte del Programa de Informes de Tendencias.

CUS Gestionar Registro de Retirada de Productos del Mercado (SIC-0024)

6.1. Crear nuevo Registro de Retirada de Productos del Mercado.

6.2. Modificar datos del Registro de Retirada de Productos del Mercado.

6.2.1. Registrar traza.

6.3. Buscar y visualizar Registro de Retirada de Productos del Mercado.

6.4. Generar un reporte del Registro de Retirada de Productos del Mercado.

CUS Gestionar Registro de Devoluciones (SIC-0023)

7.1. Crear nuevo Registro de Devoluciones.

7.2. Modificar datos del Registro de Devoluciones.

7.2.1. Registrar traza.

7.3. Buscar y visualizar Registro de Devoluciones.

7.4. Generar un reporte.

CUS Gestionar Registro de Quejas y Reclamaciones (SIC-0042)

8.1. Crear el Registro de Quejas y Reclamaciones.

8.2. Modificar datos del Registro de Quejas y Reclamaciones.

8.2.1. Registrar traza.

8.3. Buscar y Visualizar Registro de Quejas y Reclamaciones.

8.4. Generar Reporte del Registro de Quejas y Reclamaciones.

CUS Gestionar Registro de Reportes de Desviaciones (SIC-0851)

9.1. Crear Registro de Reportes de Desviaciones.

9.2. Modificar datos en el Registro de Reportes de Desviaciones.

9.2.1. Registrar traza.

9.3. Buscar y visualizar Registro de Reportes de Desviaciones.

9.4. Generar reporte del Registro de Reportes de Desviaciones.

CUS Administrar Acciones Correctivas (SIC-0848)

10.1. Crear nueva Acción Correctiva.

10.1.1. Registrar datos en la Acción Correctiva.
10.2. Modificar datos en la Acción Correctiva.
10.2.1. Registrar traza.
10.3. Buscar y visualizar una Acción Correctiva.
10.4. Generar Informe de la Acciones Correctivas (en un período de tiempo dado). [1]

2.1.2 Requisitos No Funcionales.

Los requisitos no funcionales detallados por los analistas del sistema son:

RNF de Usabilidad.

Se debe garantizar un acceso fácil y rápido a los usuarios.

RNF de Rendimiento.

Los tiempos de respuestas deben ser generalmente rápidos al igual que la velocidad de procesamiento de la información.

RNF de Soporte.

Se requiere de la instalación de un servidor Web Apache 1.3 o superior con funcionalidades relacionadas con el manejo de la BD con un gestor PostgreSQL y el servicio de interpretación de códigos con PHP 5 o superior.

RNF de Portabilidad.

El sistema podrá ser usado sobre los sistemas operativos Windows y Linux.

RNF de Seguridad.

Garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado. Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que esté activo. El sistema debe contar con protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. Contará con un registro de trazas de documentos y planillas modificadas por los usuarios, para garantizar el control de las operaciones de este tipo en el sistema. Se podrá acceder a algunas funcionalidades del sistema desde cualquier computadora personal que esté fuera del CIGB.

RNF de Confiabilidad.

El sistema será usado y administrado solamente por trabajadores del área de Calidad del CIGB, por lo tanto la información que fluirá en el mismo, será la real emitida por cada uno de los grupos o

departamentos. Podrán acceder a visualizar ciertas informaciones, directivos de otras áreas, previa consulta con la dirección del proyecto y los desarrolladores de la aplicación. El sistema validará la captación de datos para evitar entradas inadecuadas.

RNF de Software.

Se deberá disponer para instalar la aplicación desarrollada, con Sistema Operativo Windows 98 o superior o cualquier distribución de Linux. Igual sistema operativo en las computadoras personales clientes de los usuarios, las cuales además accederán al sistema usando un navegador compatible o superior con Internet Explorer 5.5+, Netscape, Mozilla 1.7 o superior o FireFox 0.9.3 o superior. En el servidor de la aplicación el sistema operativo recomendado es Windows Server 2003 o superior.

RNF de Hardware.

PC servidor: Pentium IV de capacidad de disco duro superior a 80.0 GB, microprocesadores superior a 2.0 GHz, 1.0 GB mínimo de RAM.

RNF de Restricciones en el diseño y la implementación.

Debe utilizarse como SGBD PostgreSQL. Se utilizarán herramientas de desarrollo que garanticen la calidad de todo el ciclo de desarrollo del producto. Se usará Visual Paradigm como herramienta CASE para el modelado de los artefactos que se elaboran en cada uno de los flujos de trabajo. [1]

2.2 Estrategia de integración de la solución con otros módulos.

Actualmente la Dirección de Calidad del CIGB está divida en dos departamentos, los cuales están constituidos por laboratorios y grupos de trabajo. La recogida y almacenamiento de información en cada uno de estos módulos se realiza de forma independiente. Con el objetivo de obtener una BD única para el centro, que les posibilite a cada uno de los grupos almacenar y consultar una determinada información en dependencia del proceso que se realice, se presenta una serie de actividades que posibilitan la integración de los módulos de la BD. (Ver Figura 2.1)

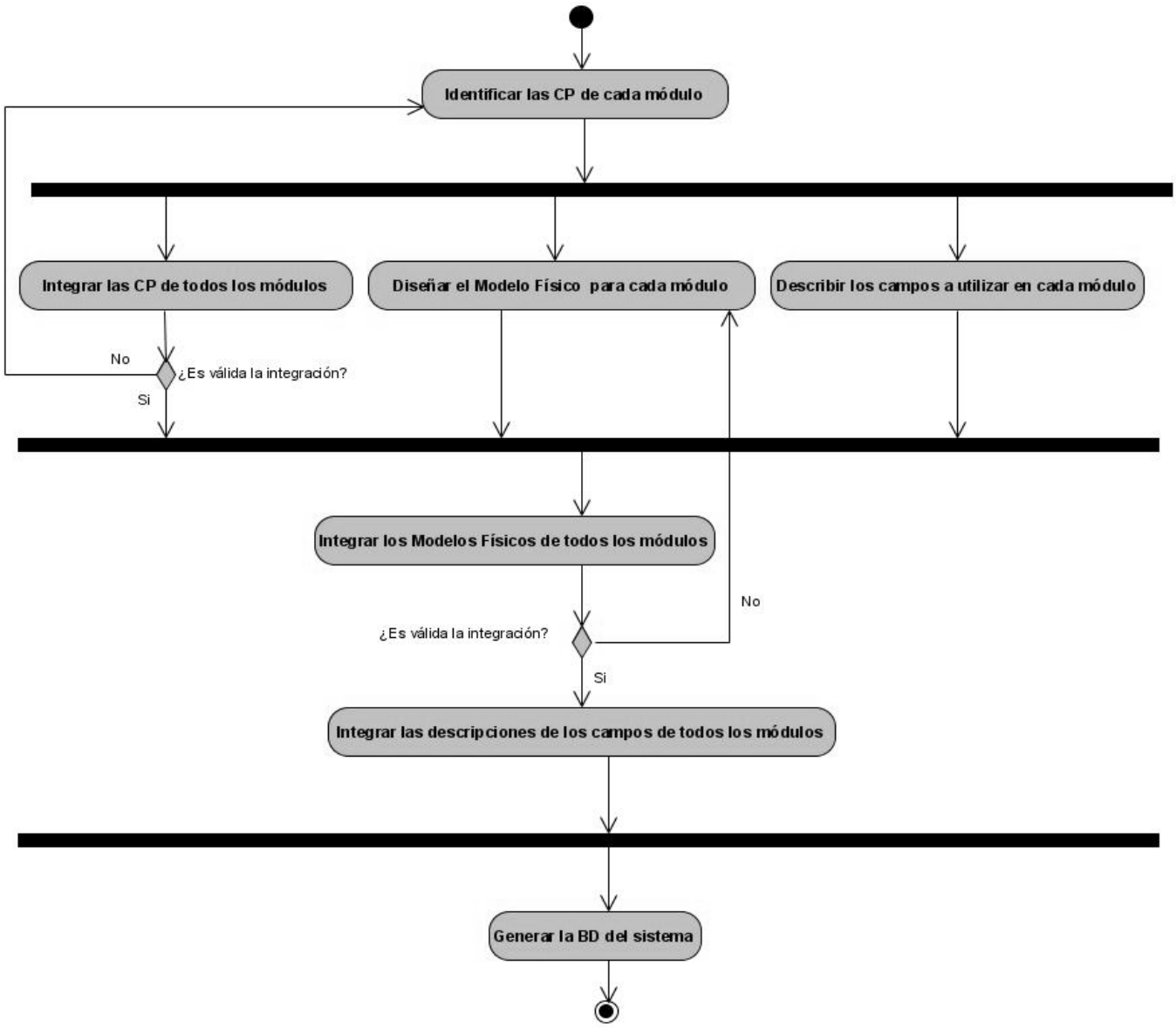


Figura 2.1 Estrategia de Integración.

Identificar las clases persistentes (CP) de cada módulo: Se identifican las clases persistentes de cada módulo a partir del diagrama de clases del diseño.

Integrar las CP de todos los módulos: Se integran las clases persistentes de todos los módulos en un único diagrama de clases que incluya las relaciones entre clases de varios módulos, en caso de que existan. La integración debe validarse, y si no constituye una correcta representación de las clases persistentes del sistema en su totalidad, deben revisarse los diagramas de clases persistentes de cada módulo nuevamente.

Diseñar el Modelo Físico de datos para cada módulo: Se realiza un modelo físico de datos para cada módulo con el objetivo de representar la estructura física de la BD, mediante entidades y las relaciones que existen entre ellas.

Describir los campos a utilizar en cada módulo: Cada módulo debe realizar una descripción de los campos que poseen cada una de sus tablas.

Integrar los Modelos Físicos de todos los módulos: Se integran los modelos físicos realizados por todos los módulos en un único diagrama. La integración debe validarse, y si no constituye una correcta representación de la estructura de la BD del sistema, deben confeccionarse nuevamente los modelos de cada módulo.

Integrar las descripciones de los campos de todos los módulos: Con las descripciones de todos los campos de las tablas de los diferentes módulos debe conformarse un diccionario de datos, para que todos los diseñadores de BD y desarrolladores de la aplicación utilicen la misma nomenclatura.

2.3 Descripción y fundamentación de la arquitectura de la BD.

Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán.

Actualmente se está trabajando con los requerimientos funcionales que están aprobados, por lo que la BD contiene 27 tablas en el modelo físico, las que serán utilizadas por el módulo Sección de Mejoramiento de la Calidad, empleando principalmente las tablas *NotificacionQuejas, RetiradaProductosMercado y RegistroDevoluciones* en las cuales se almacena la mayor cantidad de información de este módulo.

El siguiente diagrama muestra la arquitectura del sistema en tiempo de ejecución, formado por las PCs clientes conectadas por HTTPS al servidor de aplicaciones y éste conectado al servidor de base de datos por una conexión TCP/IP, incluye los dispositivos externos impresora y scanner conectados ambos a las computadoras clientes por el puerto USB.

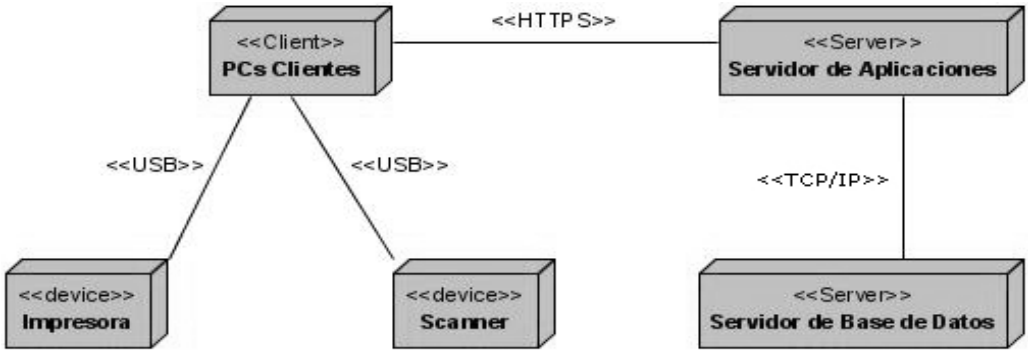


Figura 2.2 Diagrama de despliegue.

La implementación de la capa de acceso a datos se realizará en la capa del Modelo, de acuerdo a la arquitectura del patrón Modelo-Vista-Controlador que implementa Symfony. Para obtener la capa de Mapeo Objeto-Relacional se necesita el esquema, donde se describe el modelo relacional, se definen las tablas, sus relaciones y las características de sus columnas. El archivo schema.yml se generó mediante la tarea de línea de comandos < symfony propel-build-schema > y se guardan en el directorio config/ del proyecto que se definió. Las opciones de conexión con la base de datos se especifican en el archivo databases.yml.

Las clases del modelo se generaron a partir del esquema mediante la tarea <symfony propel-build-model>. Al ejecutar ese comando, se analiza el esquema y se generan las clases base del modelo, que se almacenan en el directorio lib/model/om/ del proyecto. Estas clases no se modifican porque se sobrescriben cada vez que se genera el modelo.

Para una tabla específica de la BD, en este caso el SIC0098 se generaron las clases base:

- BaseSIC0098.php
- BaseSIC0098Peer.php

También se crean las clases objeto y las clases peer en el directorio lib/model/, que heredan de las clases base y son a las que se les añaden los métodos propios, ya que no se modifican cuando se vuelve a construir el modelo.

Las clases objeto representan un registro de la BD y permiten acceder a las columnas de un registro y a los registros relacionados. Por su parte las clases peer permiten obtener los registros de las tablas de la BD. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada.

Para la tabla SIC0098 se generó la clase objeto:

- SIC0098.php

Y la clase peer:

- SIC0098Peer.php

De forma general, para todas las tablas de la BD se generaron las clases del modelo como se muestra en la Figura 2.3

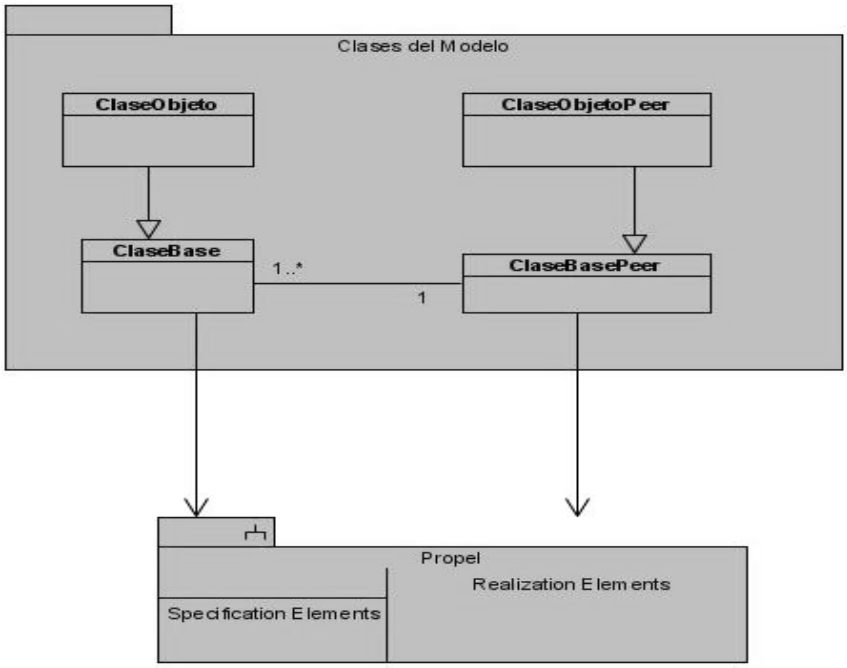


Figura 2.3 Estructura de Mapeo- Objeto Relacional implementada por Symfony.

2.4 Modelo de objetos del negocio.

El modelo de objetos es uno de los artefactos que se generan en el flujo de trabajo Modelamiento del Negocio, en el que se representan las relaciones entre los trabajadores y las entidades del negocio. Para la elaboración del modelo de datos, se consultaron los modelos de objetos realizados por los diseñadores. A continuación se refleja el modelo de objetos para el caso de uso del negocio Realizar Programa (Ver Figura 2.4). Para un análisis de los restantes modelos de objetos remitirse al Expediente del Proyecto.

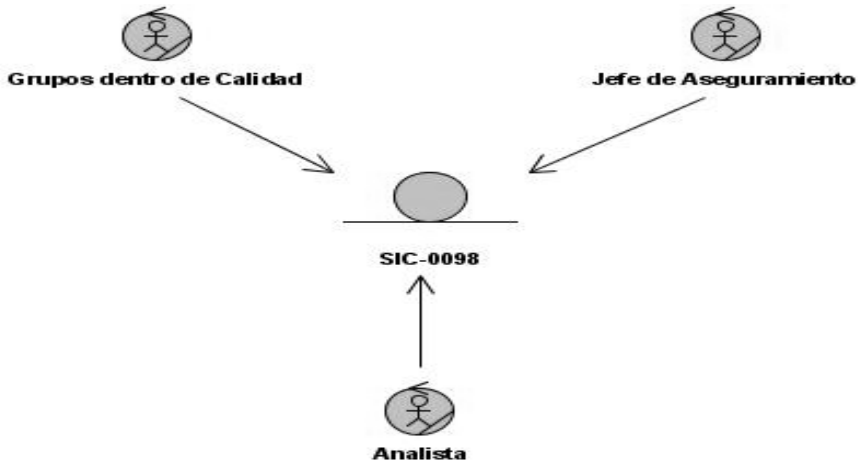


Figura 2.4 Modelo de Objetos del CU Realizar Programa.

2.5 Diagrama de clases del diseño.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar.

Una clase de diseño es una construcción similar en la implementación del sistema. Esta describe un conjunto de objetos que comparten las mismas responsabilidades, las relaciones, las operaciones, atributos y la semántica.

El siguiente diagrama de clases de diseño corresponde al caso de uso del negocio representado anteriormente Realizar Programa (Ver Figura 2.5). Para ver los diagramas más detallados consultar el Expediente de Proyecto.

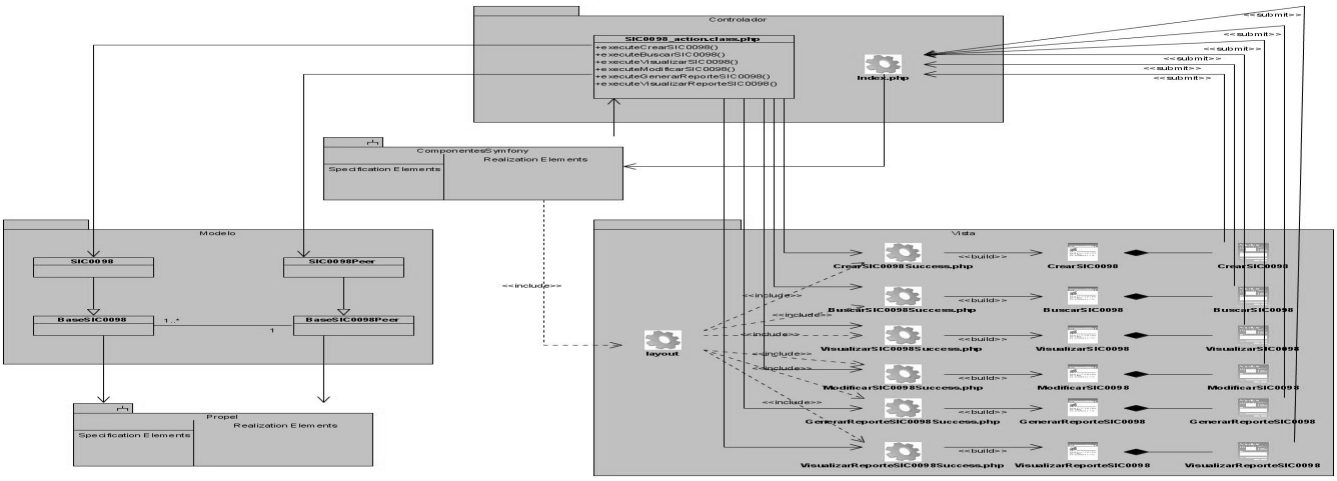


Figura 2.5 Diagrama de Clase del Diseño para el CU Programa de Monitoreo Ambiental.

2.6 Clases Persistentes. Diagrama de clases persistentes.

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes se definen para conocer la información real representada en las tablas de la BD. Generalmente las clases entidades definidas en el negocio dan origen a las clases persistentes, pues modelan la información del sistema así como el comportamiento asociado a un objeto del mundo real o un suceso. No todas las clases identificadas en el dominio del análisis son persistentes, porque pueden contener información que no sea necesario almacenar en la BD.

El diagrama de clase se utiliza para modelar la estructura lógica de la BD, con clases representando tablas, y atributos de clases representando columnas. El diagrama de clases persistentes para la SMC teniendo en cuenta las planillas con las que se trabaja es el que se muestra en la figura 2.6. Para ver con más detalles el diagrama (Ver Anexo No 6).

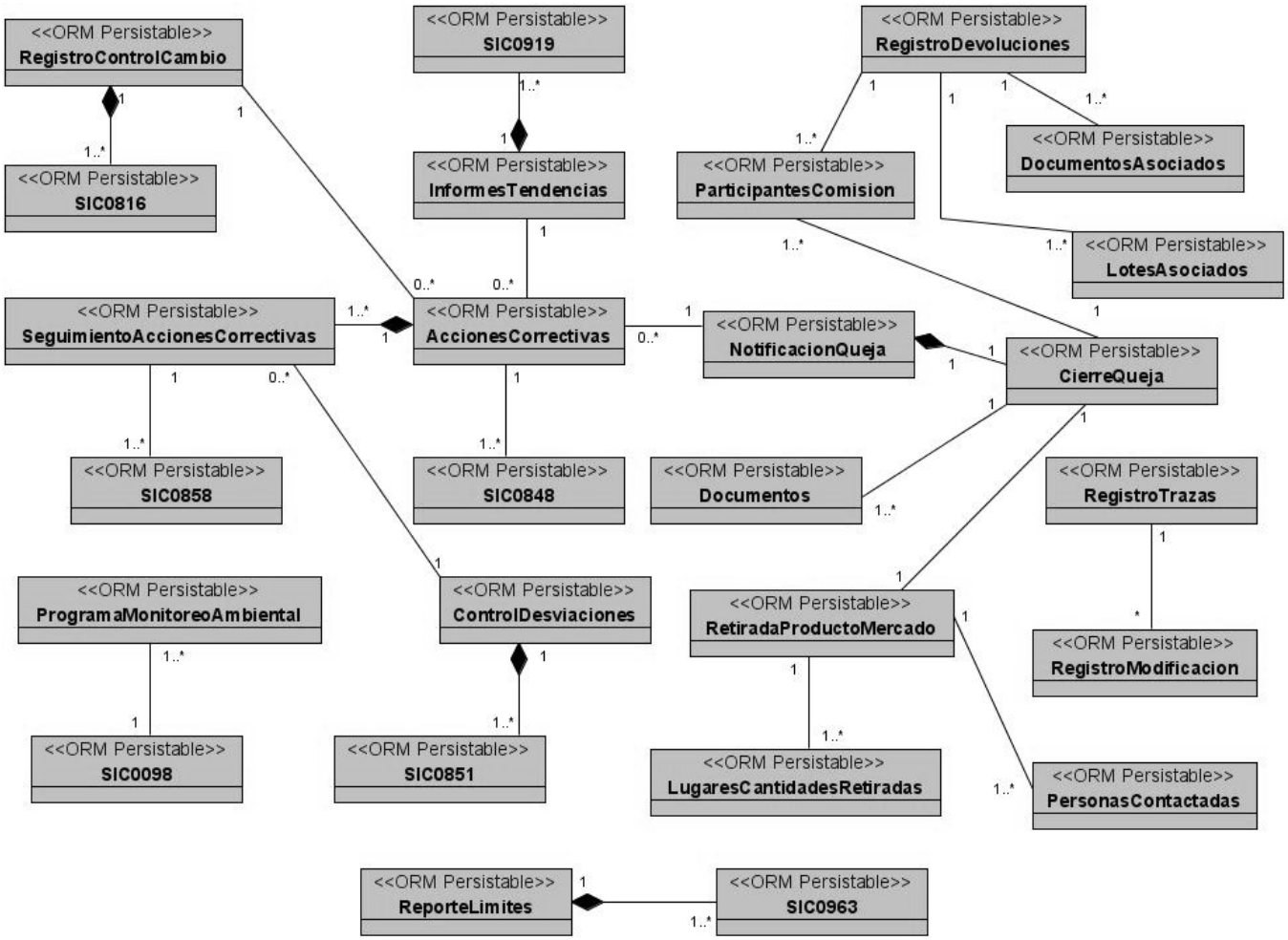


Figura 2.6 Diagrama de clases persistentes.

2.6.1 Descripción de las clases persistentes.

La descripción de las clases persistentes muestran los atributos con el tipo de dato que presentan cada uno para una mejor comprensión. Todas las clases persistentes que se describen a continuación tienen implementadas las propiedades get y set de forma tal que se puedan acceder y modificar los atributos de las mismas.

Tabla 2.1 Descripción de la clase RegistroControlCambios.

Nombre: RegistroControlCambios	
Tipo de clase: Entidad	
Atributo	**Tipo**
folio_cc	Integer
observaciones	String
Responsabilidad:	
Nombre:	RegistroControlCambios()
Descripción:	Constructor

Tabla 2.2 Descripción de la clase SIC0816.

Nombre: SIC0816	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sic0816	Integer
codigo	String
dia_recepcion	Integer
mes_recepcion	Integer
año_recepcion	Integer
menor	Boolean
mayor	Boolean
en_revision	Boolean
pendiente_documentacion	Boolean
en_reg_reg	Boolean
decision_final	String
dia_conclusion	Integer
mes_conclusion	Integer
año_conclusion	Integer
Responsabilidad:	

Nombre:	SIC0816()
Descripción:	Constructor

Tabla 2.3 Descripción de la clase ReporteLimites.

Nombre: ReporteLimites	
Tipo de clase: Entidad	
Atributo	**Tipo**
folio_rl	Integer
periodo_revisado	String
observaciones	String
terminado	Boolean
Responsabilidad:	
Nombre:	ReporteLimites()
Descripción:	Constructor

Tabla 2.4 Descripción de la clase SIC0963.

Nombre: SIC0963	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sic0963	Integer
planta	String
dia_m	Integer
mes_m	Integer
ano_m	Integer
area	String
local	String
caracteristicas	String
pm	String
nombre	String
lal	Integer
lac	Integer
realizado_por	String
revisado_por	String
dia_realizacion	Integer
mes_realizacion	Integer

año_realizacion	Integer
dia_revision	Integer
mes_revision	Integer
año_revision	Integer
Responsabilidad:	
Nombre:	SIC0963()
Descripción:	Constructor

Tabla 2.5 Descripción de la clase RegistroTrazas.

Nombre: RegistroTrazas	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_traza	Integer
usuario	String
tabla_modificada	String
fecha_modificacion	String
id_tupla_modificada	String
Responsabilidad:	
Nombre:	RegistroTrazas()
Descripción:	Constructor

Tabla 2.6 Descripción de la clase RegistroModificación.

Nombre: RegistroModificacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_mod	Integer
valor_nuevo	String
valor_viejo	String
campo_modificado	String
Responsabilidad:	
Nombre:	RegistroModificacion()
Descripción:	Constructor

Tabla 2.7 Descripción de la clase SeguimientoAccionesCorrectivas.

Nombre: SeguimientoAccionesCorrectivas	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sac	Integer
codigo_nc	String
Responsabilidad:	
Nombre:	SeguimientoAccionesCorrectivas()
Descripción:	Constructor

Tabla 2.8 Descripción de la clase SIC0858.

Nombre: SIC0858	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sic0858	Integer
numero_acciones	Integer
estado_cumplimiento	String
dia_seguimiento	Integer
mes_seguimiento	Integer
ano_seguimiento	Integer
accion_cerrada	Boolean
Responsabilidad:	
Nombre:	SIC0858()
Descripción:	Constructor

Tabla 2.9 Descripción de la clase SIC0919.

Nombre: SIC0919	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sic0919	Integer
mes	Integer
desviacion	String
Responsabilidad:	
Nombre:	SIC0919()

Descripción:	Constructor

Tabla 2.10 Descripción de la clase InformeTendencias.

Nombre: InformesTendencias	
Tipo de clase: Entidad	
Atributo	**Tipo**
folio_it	Integer
año	Integer
semestre	String
Responsabilidad:	
Nombre:	InformesTendencias()
Descripción:	Constructor

Tabla 2.11 Descripción de la clase SIC0098.

Nombre: SIC0098	
Tipo de clase: Entidad	
Atributo	**Tipo**
Idsic0098	Integer
folio0098	Integer
produccion	String
area	String
np	String
local	String
clase	String
zona_trabajo	String
terminado	Boolean
realizado_por	String
revisado_por	String
aprobado_por	String
dia_realizacion	Integer
mes_realizacion	Integer
año_realizacion	Integer
dia_revision	Integer
mes_revision	Integer
año_revision	Integer

dia_aprobacion	Integer
mes_aprobacion	Integer
año_aprobacion	Integer
Responsabilidad:	
Nombre:	SIC0098()
Descripción:	Constructor

Tabla 2.12 Descripción de la clase ProgramaMonitoreoAmbiental.

Nombre: ProgramaMonitoreoAmbiental	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_pma	Integer
caracteristicas	String
metodo	String
limite_alerta	String
limite_accion	String
frecuencia	String
instrumento_ppo	String
punto_muestreo	String
responsable	String
Responsabilidad:	
Nombre:	ProgramaMonitoreoAmbiental()
Descripción:	Constructor

Tabla 2.13 Descripción de la clase AccionesCorrectivas.

Nombre: AccionesCorrectivas	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_accion	Integer
areaa	String
codigo_inspecciones	String
codigo_auditoria	String
realizado_por	String
aprobado_por	String
dia_realizacion	Integer

mes_realizacion	Integer
año_realizacion	Integer
dia_aprobacion	Integer
mes_aprobacion	Integer
año_aprobacion	Integer
Responsabilidad:	
Nombre:	AccionesCorrectivas()
Descripción:	Constructor

Tabla 2.14 Descripción de la clase SIC0848.

Nombre: SIC0848	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sic0848	Integer
numero_acciones	Integer
acciones_correctivas	String
responsable	String
dia_cumplimiento	Integer
mes_cumplimiento	Integer
año_cumplimiento	Integer
Responsabilidad:	
Nombre:	SIC0848()
Descripción:	Constructor

Tabla 2.15 Descripción de la clase NotificacionQueja.

Nombre: NotificacionQueja	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_notificacion_queja	Integer
nombre_producto	String
forma_farmaceutica	String
dia_notificacion	Integer
mes_notificacion	Integer
año_notificacion	Integer
presentacion	String

lotes_involucrados	String
via_comunicacion_recibio	String
persona_recibe_comunicacion	String
cargo_persona_recibe	String
persona_comunica_queja	String
cargo_persona_comunica	String
institucion	String
direccion	String
via_contacto	String
descripcion_problema	String
cargo_realizador	String
cargo_revisor	String
realizado_por	String
revisado_por	String
dia_realizacion	Integer
mes_realizacion	Integer
año_realizacion	Integer
dia_revision	Integer
mes_revision	Integer
año_revision	Integer
Responsabilidad:	
Nombre:	NotificacionQueja()
Descripción:	Constructor

Tabla 2.16 Descripción de la clase CierreQueja.

Nombre: CierreQueja	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_cierre_queja	Integer
codigo_queja	String
nombre_producto	String
reportado_anteriormente	Boolean
codigo_queja_anterior	String
conclusiones_investigacion	String
procede_queja	Boolean

procede_reposicion_producto	Boolean
cargo_realizador	String
cargo_revisor	String
cargo_aprobador	String
realizado_por	String
revisado_por	String
aprobado_por	String
dia_realizacion	Integer
mes_realizacion	Integer
año_realizacion	Integer
dia_revision	Integer
mes_revision	Integer
año_revision	Integer
dia_aprobacion	Integer
mes_aprobacion	Integer
año_aprobacion	Integer
Responsabilidad:	
Nombre:	CierreQueja ()
Descripción:	Constructor

Tabla 2.17 Descripción de la clase Documentos.

Nombre: Documentos	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_documentos	Integer
codigo	String
descripcion	String
dia	Integer
mes	Integer
año	Integer
Responsabilidad:	
Nombre:	Documentos()
Descripción:	Constructor

Tabla 2.18 Descripción de la clase RegistroDevoluciones.

Nombre: RegistroDevoluciones	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_registro_devolucion	Integer
codigo_devolucion	String
folio	Integer
dia_notificacion	Integer
mes_notificacion	Integer
año_notificacion	Integer
nombre_producto	String
cantidad_producto	Integer
concentracion_dosis	Integer
cantidad_dosis_bulbo	Integer
numero_lotes	String
forma_farmaceutica	String
descripcion_problema	String
conclusiones_investigacion	String
decision	String
cargo_realizador	String
cargo_revisor	String
realizado_por	String
revisado_por	String
dia_realizacion	Integer
mes_realizacion	Integer
año_realizacion	Integer
dia_revision	Integer
mes_revision	Integer
año_revision	Integer
terminado	Boolean
Responsabilidad:	
Nombre:	RegistroDevoluciones ()
Descripción:	Constructor

Tabla 2.19 Descripción de la clase LotesAsociados.

Nombre: LotesAsociados	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_lotes_asociados	Integer
lotes	String
cantidad_bulbos	Integer
Responsabilidad:	
Nombre:	LotesAsociados()
Descripción:	Constructor

Tabla 2.20 Descripción de la clase DocumentosAsociados.

Nombre: DocumentosAsociados	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_documentos_a	Integer
documentoss	String
Responsabilidad:	
Nombre:	DocumentosAsociados()
Descripción:	Constructor

Tabla 2. 21 Descripción de la clase ParticipantesComision.

Nombre: ParticipantesComision	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_participantes_comision	Integer
nombre	String
cargo	String
Responsabilidad:	
Nombre:	ParticipantesComision()
Descripción:	Constructor

Tabla 2.22 Descripción de la clase PersonasContactadas.

Nombre: PersonasContactadas	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_personas_contactadas	Integer
nombre_apellidos	String
cargo	String
institucion	String
Responsabilidad:	
Nombre:	PersonasContactadas()
Descripción:	Constructor

Tabla 2.23 Descripción de la clase RetiradaProductosMercado.

Nombre: RetiradaProductosMercado	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_retirada_productos_mercado	Integer
producto	String
folio	Integer
np	Integer
dia_inicio_retirada	Integer
mes_inicio_retirada	Integer
año_inicio_retirada	Integer
dia_conclusion_retirada	Integer
mes_conclusion_retirada	Integer
año_conclusion_retirada	Integer
lotes_involucrados	String
total_bulbos_liberados	Integer
naturaleza_defecto	String
accion_tomada	String
urgencia	String
observaciones	String
destino_producto_retirado	String
dia_destruccion	Integer

mes_destruccion	Integer
año_destruccion	Integer
cargo_realizador	String
cargo_aprobador	String
realizado_por	String
aprobado_por	String
dia_realizacion	Integer
mes_realizacion	Integer
año_realizacion	Integer
dia_aprobacion	Integer
mes_aprobacion	Integer
año_aprobacion	Integer
Responsabilidad:	
Nombre:	RetiradaProductoMercado()
Descripción:	Constructor

Tabla 2.24 Descripción de la clase LugaresCantidadesRetiradas.

Nombre: LugaresCantidadesRetiradas	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_lugares_cantidades_retiradas	Integer
lotes	String
lugar	String
cantidad_distribuida	Integer
cantidad_retirada	Integer
Responsabilidad:	
Nombre:	LugaresCantidadesRetiradas()
Descripción:	Constructor
Nombre:	Diferencia (Integer CantidadDistribuida, Integer CantidadRetirada)
Descripción:	Método que calcula la diferencia entre la cantidad distribuida y la cantidad retirada.

Tabla 2.25 Descripción de la clase ControlDesviaciones.

Nombre: ControlDesviaciones	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_control_desviaciones	Integer
producto	String
folio	Integer
Responsabilidad:	
Nombre:	ControlDesviaciones()
Descripción:	Constructor

Tabla 2.26 Descripción de la clase SIC0851.

Nombre: SIC0851	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sic0851	Integer
codigo	String
dia_notificacion_realizacion	Integer
mes_notificacion_realizacion	Integer
año_notificacion_realizacion	Integer
etapa	String
numero_lotes	String
identificacion_desviacion	String
Responsabilidad:	
Nombre:	SIC0851()
Descripción:	Constructor

2.7 Diseño de la BD.

El diseño de la BD tiene que realizarse de forma tal que los datos persistentes sean almacenados consistente y eficientemente. La BD para la Sección de Mejoramiento de la Calidad es el medio para el almacenamiento de todos los datos relacionados con el mejoramiento de la calidad de las producciones del centro.

2.7.1 Modelo de Datos.

El modelo de datos es usado para describir la representación lógica y física de la información persistente manejada por el sistema. El modelo de datos correspondiente a la SMC, teniendo en cuenta las clases persistentes del diseño y las relaciones que se establecen entre las mismas es el que se muestra en la figura 2.7. La vista del modelo de datos, con los atributos de las clases, los tipos de datos y las llaves que se exportan debido a las relaciones existentes se muestra en el Anexo No 7.

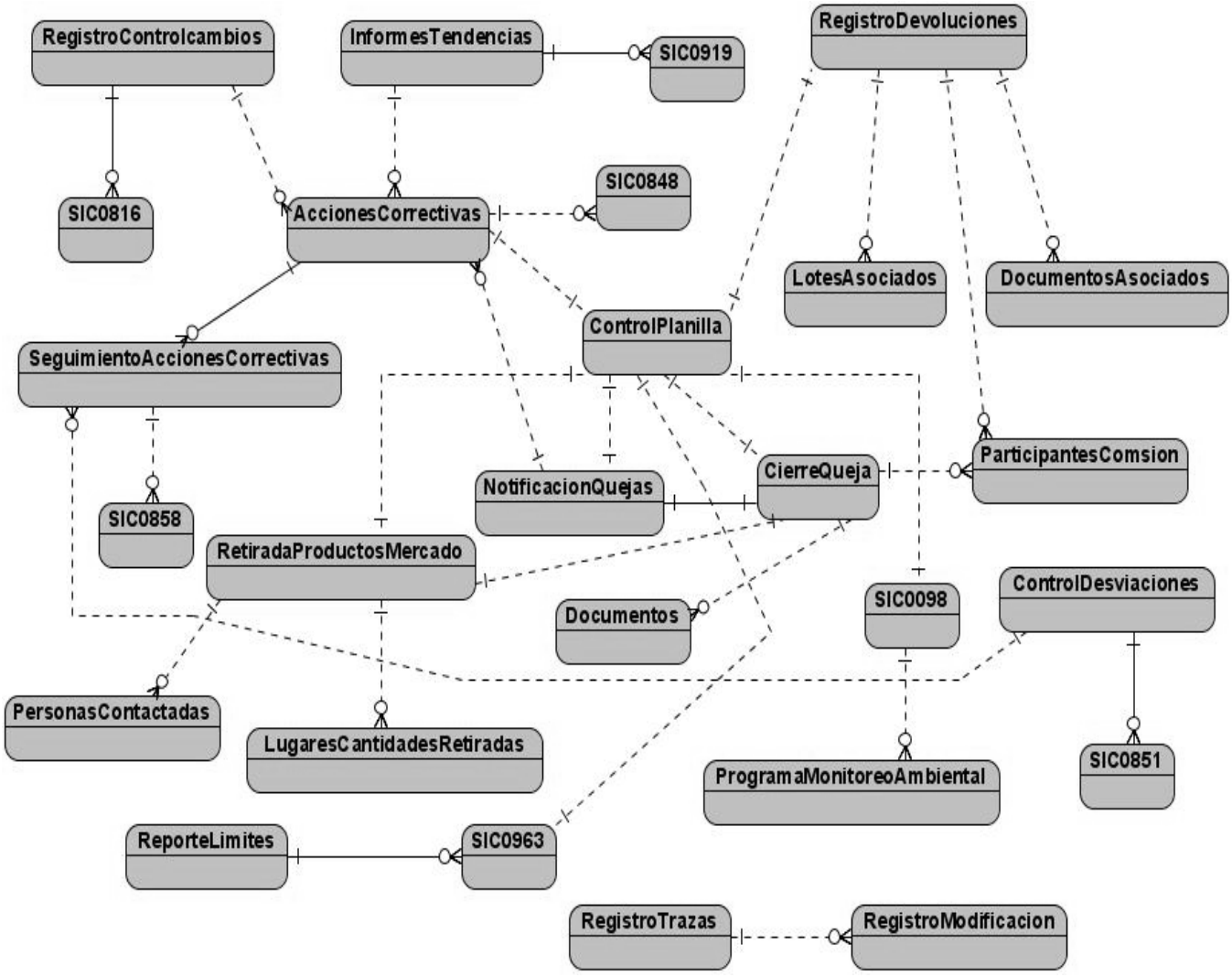


Figura 2.7 Modelo de Datos.

2.7.2 Descripción de las tablas de la BD.

Tabla 2.27 Descripción de la tabla RegistroControlCambios.

Nombre: RegistroControlCambios		
Descripción: En esta tabla se crean los folios del SIC0816		
Atributo	**Tipo**	**Descripción**
FolioCC	Bigint	Es un valor entero de dos dígitos, cada control de cambio tendrá un folio único, el cual se iría incrementando a medida que se necesite realizar un nuevo registro. Es la llave primaria.
Observaciones	Varchar(255)	Información detallada sobre datos específicos de la planilla.

Tabla 2.28 Descripción de la tabla SIC0816.

Nombre: SIC0816		
Descripción: Representa la entidad donde se establece la política de cambio en los procesos y técnicas analíticas, para garantizar el cumplimiento de las buenas prácticas de fabricación y laboratorio. Cualquier variación debe ser justificada y registrada.		
Atributo	**Tipo**	**Descripción**
IdSic0816	Serial	Es el identificador del SIC0816, es un valor entero auto incremental, llave primaria.
Codigo	Varchar(10)	Secuencia de caracteres que va a tener un valor único, permitiendo diferenciar cada planilla.
DiaRecepcion	Smallint	Dia en que se comienza el trámite del cambio.
MesRecepcion	Smallint	Mes en que se comienza el trámite del cambio
AnoRecepcion	Smallint	Año en que se comienza el trámite del cambio
Mayor	Bit	Valor que puede tomar la clasificación, en dependencia de la potencialidad de incidir negativamente sobre la identidad, fortaleza, potencia, calidad y pureza del producto.
Menor	Bit	Valor que puede tomar la clasificación, en dependencia de la potencialidad de incidir negativamente sobre la identidad, fortaleza, potencia, calidad y pureza del producto.
EnRevision	Bit	Estado en el que pueden estar un cambio.
PendienteDocumentacion	Bit	Estado en el que pueden estar un cambio.
EnRegReg	Bit	Estado en el que pueden estar un cambio.
DecisionFinal	Varchar(255)	Se refiere a la decisión de aprobación, rechazo, o si procede el cambio.
DiaConclusion	Smallint	Se llenará una vez concluido el plan de acciones por parte del solicitante, se especifica el día.
MesConclusion	Smallint	Mes en que culmina el proceso de cambio.
AnoConclusion	Smallint	Año en que culmina el proceso de cambio.
FolioCC	Bigint	Llave primaria foránea que la toma de la entidad RegistroControlCambios.

Tabla 2.29 Descripción de la tabla RegistroTrazas.

Nombre: RegistroTrazas		
Descripción: En esta tabla se registran todas las acciones realizadas sobre las entidades.		
Atributo	**Tipo**	**Descripción**
IdTraza	Serial	Es el identificador de la tabla RegistroTrazas, es un valor entero auto incremental, llave primaria.
Usuario	Varchar(15)	Nombre del usuario que accedió a la BD.
TablaModificada	Varchar(50)	Tabla a la que se le realizó la modificación.
FechaModificacion	Timestamp	Fecha en que se realizó la modificación.
IdTuplaModificada	Varchar(50)	El identificador de la tupla que se modificó.

Tabla 2.30 Descripción de la tabla RegistroModificaciones.

Nombre: RegistroModificaciones		
Descripción: En esta tabla se registran todas las acciones realizadas sobre las entidades, centrándonos en los valores modificados.		
Atributo	**Tipo**	**Descripción**
IdMod	Serial	Es el identificador de la tabla RegistroModificaciones, es un valor entero auto incremental, llave primaria.
ValorNuevo	Varchar(255)	Valor nuevo que toma el campo modificado.
ValorViejo	Varchar(255)	Valor que tenía el campo antes de ser modificado.
CampoModificado	Timestamp	Campo al que se le realizó la modificación.
IdTraza	Bigint	Llave foránea que la toma de la entidad RegistroTrazas.

Tabla 2.31 Descripción de la tabla SeguimientoAccionesCorrectivas.

Nombre: SeguimientoAccionesCorrectivas		
Descripción: En esta tabla se especifica el área y el código del proceso al que se le realizará el seguimiento de la acción correctiva.		
Atributo	**Tipo**	**Descripción**
IdSAC	Serial	Es el identificador de la tabla, un valor entero, llave primaria.
CodigoDesviaciones	Varchar(10)	Código de la desviación a la que se le realiza el seguimiento de acciones correctivas.
CodigoNc	Varchar(10)	Código de la no conformidad a la que se le realiza el seguimiento de acciones correctivas.
IdAccion	Bigint	Llave primaria foránea que la toma de la entidad AccionesCorrectivas.

Tabla 2.32 Descripción de la tabla InformesTendencias.

Nombre: InformesTendencias		
Descripción: En esta tabla se crean los campos necesarios para representar la información el SIC0919.		
Atributo	**Tipo**	**Descripción**
FolioIT	Smallint	Es el identificador de la tabla, es un valor entero de dos dígitos único que representa el folio de la planilla.

Ano	Smallint	Año en que se realiza el informe de tendencia.
Semestre	Varchar(30)	Semestre en que se realiza el informe de tendencia.

Tabla 2.33 Descripción de la tabla SIC0919.

Nombre: SIC0919		
Descripción: Representa la entidad que analiza el comportamiento de las características de los diferentes productos en un período determinado, para tomar medidas e investigar las cusas reales de los problemas. También sirve para mantener bajo control los diferentes procesos.		
Atributo	**Tipo**	**Descripción**
IdSic0919	Serial	Es el identificador del SIC0919, es un valor entero auto incremental, llave primaria.
Mes	Smallint	Mes en que se realiza el informe de tendencia.
Desviacion	Varchar(255)	Razones por la que no se ejecutó el informe de tendencias.
FolioIT	Smallint	Llave primaria foránea que la toma de la entidad InformesTendencias.

Tabla 2.34 Descripción de la tabla SIC0858.

Nombre: SIC0858		
Descripción: Representa la entidad donde se establecen los pasos para realizar el seguimiento a las acciones correctivas y preventivas en un área determinada, para eliminar las causas de no conformidades detectadas al sistema de calidad, a los productos y a los procesos.		
Atributo	**Tipo**	**Descripción**
IDSIC0858	Serial	Es el identificador del SIC0858, es un valor entero auto incremental, llave primaria.
NumeroAcciones	Smallint	Número de acciones correctivas que se realizaron.
EstadoCumplimiento	Varchar(15)	El estado en que está la acción correctiva (completado o no).
DiaSeguimiento	Smallint	Día en que se verifica la acción correctiva.
MesSeguimiento	Smallint	Mes en que se verifica la acción correctiva.
AnoSeguimiento	Smallint	Año en que se verifica la acción correctiva.
AccionCerrada	Bit	Se realiza cuando se comprueba que se cumplieron las medidas correctivas y de no ser posible se hace un nuevo análisis y se modifica la medida o la fecha de cumplimiento de la misma.
IdAccion	Bigint	Llave foránea que la toma de la entidad SeguimientoAccionesCorrectivas.
IdSAC	Bigint	Llave foránea que la toma de la entidad SeguimientoAccionesCorrectivas.

Tabla 2.35 Descripción de la tabla ReporteLimites.

Nombre: ReporteLimites		
Descripción: En esta tabla se crean los folios y el período de revisado del SIC0963.		
Atributo	**Tipo**	**Descripción**
FolioRL	Bigint	Cadena de caracteres, llave primaria.
PeriodoRevisado	Varchar(255)	Período en que va a ser revisado el reporte de límites.

Observaciones	Varchar(255)	Información detallada sobre datos específicos de la planilla.
Terminado	Bit	Si el proceso se terminó o no.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.

Tabla 2.36 Descripción de la tabla SIC0098.

Nombre: SIC0098		
Descripción: Representa la entidad que establece los procedimientos necesarios para la elaboración y coordinación de los programas de monitoreo ambiental de las áreas limpias de los diferentes productos del CIGB.		
Atributo	**Tipo**	**Descripción**
IdSIC0098	Serial	Es un valor entero auto incremental, llave primaria.
Folio0098	Smallint	Es un valor entero de dos dígitos único que representa el folio de la planilla.
Produccion	Varchar(30)	Se aclara la producción para la que se desarrolla el programa, tanto para sistemas críticos como para plantas multiproductos.
Areaa	Varchar(30)	Se especifica la planta de producción y/o el área o etapa de producción.
Np	Smallint	Número de parte donde se especifica el área de trabajo.
Locall	Varchar(30)	Nombre o código del local. En el caso de los PMA de sistemas críticos se invalida este espacio.
Clase	Varchar(30)	Se mencionarán las clases que le corresponde al local. En el caso de los PMA de Sistemas críticos se invalida este espacio.
ZonaTrabajo	Varchar(30)	Se especifica dentro de un local la zona de trabajo. En el caso de los PMA de Sistemas críticos se invalida este espacio.
Terminado	Boolean	Si se terminó o no el programa.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.

Tabla 2.37 Descripción de la tabla SIC0963.

Nombre: SIC0963		
Descripción: Representa la entidad que se encarga de la detención de valores fuera de los límites de alerta y de acción e incumplimiento de la frecuencia de medición.		
Atributo	**Tipo**	**Descripción**
IdSic0963	Serial	Es el identificador del SIC0963, es un valor entero auto incremental, llave primaria.
Planta	Varchar(30)	Se especifica la planta de producción.
DiaM	Smallint	Dia en que se realiza el muestreo.
MesM	Smallint	Mes en que se realiza el muestreo.
AnoM	Smallint	Año en que se realiza el muestreo.
Areaa	Varchar(30)	Se especifica la planta de producción y/o el área o etapa de producción.
Locall	Varchar(30)	Nombre o código del local.
Caracteristicas	Varchar(255)	Características que van a tener los reportes de

		límites.
Pm	Varchar(255)	Punto de muestreo.
Nombre	Varchar(255)	Nombre del producto que se le realiza el muestreo.
Lal	Smallint	Límite de alerta.
Lac	Smallint	Límite de acción.
FolioRL	Bigint	Llave primaria foránea que la toma de la entidad ReporteLimites.

Tabla 2.38 Descripción de la tabla ProgramaMonitoreoAmbiental.

Nombre: ProgramaMonitoreoAmbiental		
Descripción: En esta tabla se especifican una serie de datos que se almacenan en un Programa de Monitoreo Ambiental, en dependencia de la cantidad de información que se necesite registrar.		
Atributo	**Tipo**	**Descripción**
IdPMA	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Caracteristicas	Varchar(255)	Se establecerá el nombre de la característica objeto de control.
Metodo	Varchar(255)	Nombre del método por el que se miden las características.
LimiteAlerta	Varchar(30)	Se establecerán para cada una de las características. Se definirá preferentemente en forma de intervalos, máximo o mínimo y se aclarará la unidad de medida correspondiente. Puede que no exista.
LimiteAccion	Varchar(30)	Se establecerán para cada una de las características. Se definirá preferentemente en forma de intervalos, máximo o mínimo y se aclarará la unidad de medida correspondiente. Puede que no exista.
Frecuencia	Varchar(50)	Se hará referencia al intervalo de tiempo entre las diferentes muestras de la característica.
InstrumentoPpo	Varchar(30)	Define como se va a proceder para realizar el monitoreo.
PuntoMuestreo	Varchar(30)	Se establecerán los lugares donde se tomarán las muestras para realizar la determinación de la característica, o se hará referencia al título del documento donde se encuentre descrita.
Responsable	Varchar(30)	Es el laboratorio o grupo encargado de realizar el ensayo. Cuando la característica sea controlada por una entidad externa al CIGB, se establecerá claramente el nombre de dicha entidad.
IdSIC0098	Bigint	Llave foránea que la toma de la entidad SIC0098.

Tabla 2.39 Descripción de la tabla AccionesCorrectivas.

Nombre: AccionesCorrectivas		
Descripción: En esta tabla se especifica el área y el código del proceso al que se le realizará la acción correctiva.		
Atributo	**Tipo**	**Descripción**

IdAccion	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Areaa	Varchar(30)	Área en que se tiene que llevar a cabo la acción correctiva.
CodigoInspecciones	Varchar(10)	Código de la inspección a la que se le realiza el seguimiento de acciones correctivas.
CodigoAuditoria	Varchar(10)	Código de la auditoria a la que se le realiza el seguimiento de acciones correctivas.
FolioIT	Smallint	Llave foránea que la toma de la entidad InformesTendencias.
IdNotificacionQueja	Bigint	Llave foránea que la toma de la entidad NotificacionQueja.
FolioCC	Bigint	Llave foránea que la toma de la entidad RegistroControlCambios.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.
IdRegistroDevoluciones	Bigint	Llave foránea que la toma de la entidad RegistroDevoluciones.

Tabla 2.40 Descripción de la tabla SIC0848.

Nombre: SIC0848		
Descripción: Representa la entidad donde se establecen los pasos para realizar las acciones correctivas y preventivas en un área determinada, para eliminar las causas de no conformidades detectadas al sistema de calidad, a los productos y a los procesos.		
Atributo	**Tipo**	**Descripción**
IdSic0848	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
NumeroAcciones	Smallint	Número de acciones correctivas que se realizarán.
AccionesCorrectivas	Varchar(255)	Acciones correctivas que se llevarán acabo.
Responsable	Varchar(30)	Nombre del grupo al cual pertenece la persona que realiza la acción correctiva.
DiaCumplimiento	Smallint	Dia en el que se le debe dar cumplimiento a la acción correctiva tomada.
MesCumplimiento	Smallint	Mes en el que se le debe dar cumplimiento a la acción correctiva tomada.
AnoCumplimiento	Smallint	Año en el que se le debe dar cumplimiento a la acción correctiva tomada.
IdAccion	Bigint	Llave foránea que la toma de la entidad AccionesCorrectivas.

Tabla 2.41 Descripción de la tabla NotificacionQueja.

Nombre: NotificacionQuejas		
Descripción: En esta tabla se registra la información relacionada con cualquier inconformidad del cliente con algún producto del CIGB.		
Atributo	**Tipo**	**Descripción**
IdNotificacionQuejas	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
NombreProducto	Varchar(30)	Nombre del producto sobre el que se origina la queja.
FormaFarmaceutica	Varchar(20)	Forma en que se presenta el producto puede

		ser inyectable, oral de uso tópico, entre otras.
DiaNotificacion	Smallint	Dia en que se comunica la queja.
MesNotificacion	Smallint	Mes en que se comunica la queja.
AnoNotificacion	Smallint	Ano en que se comunica la queja.
Presentacion	Varchar(255)	Forma en que se presenta el producto.
LotesInvolucrados	Varchar(50)	Lotes que contienen al producto sobre el que se origina la queja.
ViaComunicacionRecibio	Varchar(50)	Vía por la que se recibió la queja.
PersonaRecibeComunicacion	Varchar(30)	Nombre de la persona que recibe la queja.
CargoPersonaRecibe	Varchar(20)	Cargo de la persona que recibe la queja.
PersonaComunicaQueja	Varchar(30)	Nombre de la persona que se queja de algún producto del CIGB.
CargoPersonaComunica	Varchar(20)	Cargo de la persona que se queja de algún producto del CIGB.
Institucion	Varchar(30)	Institución a la que pertenece la persona que se queja.
Direccion	Varchar(50)	Dirección de la persona o centro que se queja.
ViaContacto	Varchar(50)	Vía por la que se contactó con el CIGB para informar de la queja.
DescripcionProblema	Varchar(255)	Descripción de la queja.
CargoRealizador	Varchar(20)	Cargo de la persona que realizó la notificación de la queja.
CargoRevisor	Varchar(20)	Cargo de la persona que revisó la notificación de la queja.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.

Tabla 2.42 Descripción de la tabla CierreQueja.

Nombre: CierreQueja		
Descripción: En esta tabla se cierra el análisis comenzado por cualquier inconformidad del cliente con algún producto del CIGB.		
Atributo	**Tipo**	**Descripción**
IdCierreQueja	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria
CodigoQueja	Varchar(10)	Cadena de caracteres única, donde se especifica el codigo de la queja que se desea cerrar.
NombreProducto	Varchar(30)	Nombre del producto al que se le está haciendo el cierre de la queja.
ReportadoAnteriormente	Boolean	Se especifica si el producto fue reportado o no anteriormente.
CodigoQuejaAnterior	Varchar(10)	En caso de haber sido reportado anteriormente, se especifica el código de la queja anterior.
ConclusionesInvestigacion	Varchar(255)	Resultado final después de realizada la investigación.
ProcedeQueja	Boolean	Se decide si se procede o no con la queja.
ProcedeReposicionProducto	Boolean	Se decide si se procede o no con la reposición del producto.
CargoAprobador	Varchar(20)	Cargo de la persona que aprobó el cierre de la queja.
IdNotificacionQuejas	Bigint	Llave primaria foránea que la toma de la

		entidad NotificacionQuejas.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.

Tabla 2.43 Descripción de la tabla PersonasContactadas.

Nombre: PersonasContactadas		
Descripción: En esta tabla se especifican los datos de las personas que intervienen en todo el proceso de la retirada del producto.		
Atributo	**Tipo**	**Descripción**
IdPersonasContactadas	Integer	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
NombreApellidos	Varchar(30)	Nombre y apellidos de las personas contactadas durante la investigación.
Cargo	Varchar(20)	Cargo de cada una de las personas contactadas.
Institucion	Varchar(20)	Nombre de la institución o centro de trabajo a la que pertenece dicha persona.
IdRetiradaProductosMercado	Bigint	Llave foránea que la toma de la entidad RetiradaProductosMercado.

Tabla 2.44 Descripción de la tabla SIC0851.

Nombre: SIC0851		
Descripción: Cuando los límites de alerta y de acción dan por encima de lo esperado, se genera una no conformidad, las cuales a medida que se van aprobando se van registrando en esta tabla.		
Atributo	**Tipo**	**Descripción**
IdSic0851	Serial	Es el identificador de la tabla, es un valor entero auto incremento, llave primaria.
Codigo	Varchar(10)	Código de la no conformidad.
DiaNotificacionRealizacion	Smallint	Día en que se notificó y se realizó la no conformidad.
MesNotificacionRealizacion	Smallint	Mes en que se notificó y se realizó la no conformidad.
AnoNotificacionRealizacion	Smallint	Año en que se notificó y se realizó la no conformidad.
Etapa	Varchar(50)	Etapa del proceso en que se encontraba el producto cuando se identificó la no conformidad.
NumeroLotes	Varchar(30)	Lote al que pertenece el producto que se le identificó la no conformidad.
IdentificacionDesviacion	Varchar(30)	Método por el cual se realizó el monitoreo y se identificó la no conformidad.
IdControlDesviaciones	Bigint	Llave foránea que la toma de la entidad ControlDesviaciones.

Tabla 2.45 Descripción de la tabla RetiradaProductosMercado.

Nombre: RetiradaProductosMercado		
Descripción: En esta tabla se registran los datos de los productos que se encuentran en la etapa de distribución y en la red comercial y que se ha descubierto o se sospecha que tiene defectos que pueden provocar consecuencias adversas.		
Atributo	**Tipo**	**Descripción**

IdRetiradaProductosMercado	Serial	Es el identificador de la tabla, es un valor entero auto incremento, llave primaria de la tabla.
Producto	Varchar(30)	Es una cadena de caracteres que va a almacenar el nombre del producto. Un producto puede tener más de una retirada del mercado, diferenciándose por el folio que tenga cada una.
Folio	Smallint	Es un valor entero de 2 dígitos, que puede tomar el mismo valor para productos diferentes
Np	Smallint	Número de parte donde se especifica el área de trabajo.
DiaInicioRetirada	Smallint	Día en que se notifica y se comienza a retirar el producto de la red comercial.
MesInicioRetirada	Smallint	Mes en que se notifica y se comienza a retirar el producto de la red comercial.
AnoInicioRetirada	Smallint	Año en que se notifica y se comienza a retirar el producto de la red comercial.
DiaConclusionRetirada	Smallint	Día en que se termina de retirar el producto de la red comercial.
MesConclusionRetirada	Smallint	Mes en que se termina de retirar el producto de la red comercial.
AnoConclusionRetirada	Smallint	Año en que se termina de retirar el producto de la red comercial.
LotesInvolucrados	Varchar(30)	Códigos de los lotes involucrados en la retirada de la red comercial.
TotalBulbosLiberados	Integer	Total de bulbos que fueron liberados al mercado de ese producto.
NaturalezaDefecto	Varchar(255)	Se da brevemente una explicación de las causas por las que se retira el producto.
AccionTomada	Varchar(40)	Representa la acción que se toma después de haber analizado el producto en cuestión.
Urgencia	Varchar(20)	Es el nivel de la retirada, en dependencia de la urgencia que demande.
Observaciones	Varchar(255)	Se detalla cualquier información válida referente al producto que se desea retirar.
DestinoProductoRetirado	Varchar(15)	Se especifica el destino del producto que se desea retirar, este campo siempre va a tomar el mismo valor, la destrucción del producto.
DiaDestruccion	Smallint	Se especifica el día en que va a ser destruido dicho producto.
MesDestruccion	Smallint	Se especifica el mes en que va a ser destruido dicho producto.
AnoDestruccion	Smallint	Se especifica el año en que va a ser destruido dicho producto.
CargoRealizador	Varchar(20)	Cargo de la persona que realizó el registro de retirada.
CargoAprobador	Varchar(20)	Cargo de la persona que aprueba el registro de retirada.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.
IdCierreQueja	Bigint	Llave foránea que la toma de la entidad CierreQueja.

IdNotificacionQueja	Bigint	Llave foránea que la toma de la entidad NotificacionQueja.

Tabla 2.46 Descripción de la tabla ControlDesviaciones.

Nombre: ControlDesviaciones		
Descripción: En esta tabla se especifican los datos necesarios para crear inicialmente un Control de Desviaciones.		
Atributo	**Tipo**	**Descripción**
IdControlDesviaciones	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Producto	Varchar(30)	Nombre del producto al cual pertenece la no conformidad. Un producto puede tener más de un control de desviaciones, diferenciándose por el folio que tenga cada una.
Folio	Smallint	Es un valor entero de 3 dígitos, que puede tomar el mismo valor para productos diferentes.

Tabla 2.47 Descripción de la tabla ParticipantesComision.

Nombre: ParticipantesComision		
Descripción: En esta tabla se almacenan los nombres y cargos de todas las personas que participan en el análisis de la queja.		
Atributo	**Tipo**	**Descripción**
IdParticipantesComision	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Nombre	Varchar(30)	Nombre de las personas que participan en la investigación.
Cargo	Varchar(20)	Cargo de las personas que participan en la investigación.
IdCierreQueja	Bigint	Llave foránea que la toma de la entidad CierreQueja.
IdRegistroDevoluciones	Bigint	Llave foránea que la toma de la entidad RegistroDevoluciones.
IdNotificacionQuejas	Bigint	Llave foránea que la toma de la entidad NotificacionQuejas.

Tabla 2.48 Descripción de la tabla Documentos.

Nombre: Documentos		
Descripción: En esta tabla se almacenan los documentos que se relacionan con la queja durante la investigación.		
Atributo	**Tipo**	**Descripción**
IdDocumentos	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Codigo	Varchar(10)	Código de los documentos que se relacionan con la queja.
Descripcion	Varchar(255)	Descripción que se relata en los documentos sobre la queja.
Dia	Smallint	Día en que se consultó dicho documento.
Mes	Smallint	Mes en que se consultó dicho documento.
Ano	Smallint	Año en que se consultó dicho documento.

IdCierreQueja	Bigint	Llave foránea que la toma de la entidad CierreQueja.
IdNotificacionQuejas	Bigint	Llave foránea que la toma de la entidad NotificacionQuejas.

Tabla 2.49 Descripción de la tabla LugaresCantidadesRetiradas.

Nombre: LugaresCantidadesRetiradas		
Descripción: En esta tabla se especifican los datos de las personas que intervienen en todo el proceso de la retirada del producto.		
Atributo	**Tipo**	**Descripción**
IdLugaresCantidadesRetiradas	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Lotes	Varchar(10)	Se especifica el código de los lotes asociados al producto que se desea retirar.
Lugar	Varchar(40)	Nombre del lugar de donde se va a llevar a cabo la retirada del producto.
CantidadDistribuida	Bigint	Cantidad de unidades distribuida de ese producto en dependencia del lugar.
CantidadRetirada	Bigint	Cantidad de unidades retiradas de ese producto.
IdRetiradaProductosMercado	Bigint	Llave foránea que la toma de la entidad RetiradaProductosMercado.

Tabla 2.50 Descripción de la tabla LotesAsociados.

Nombre: LotesAsociados		
Descripción: En esta tabla se especifican todos los códigos de los lotes que tenga asociado un registro de devolución de un determinado producto.		
Atributo	**Tipo**	**Descripción**
IdLotesAsociados	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Lotes	Varchar(10)	Cadena de caracteres que representa los códigos de los lotes asociados a una devolución.
CantidadBulbos	Integer	Cantidad de bulbos en dependencia del lote al que pertenezca.
IdRegistroDevoluciones	Bigint	Llave foránea que la toma de la entidad RegistroDevoluciones.

Tabla 2.51 Descripción de la tabla RegistroDevoluciones.

Nombre: RegistroDevoluciones		
Descripción: En esta tabla se registran los datos del producto liberado o comercializado, involucrado en la devolución y las conclusiones de la investigación realizada.		
Atributo	**Tipo**	**Descripción**
IdRegistroDevoluciones	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
CodigoDevolucion	Varchar(10)	Es una cadena de caracteres única para cada registro de devolución que se realice.
Folio	Smallint	Es un valor entero de 2 dígitos, que puede tomar el mismo valor para códigos de devoluciones diferentes

DiaNotificacion	Smallint	Dia en que se notifica la devolución.
MesNotificacion	Smallint	Mes en que se notifica la devolución.
AnoNotificacion	Smallint	Ano en que se notifica la devolución.
NombreProducto	Varchar(30)	Nombre del producto involucrado en la devolución.
CantidadProducto	Integer	Se especifica la cantidad de producto de los lotes involucrados en la devolución.
ConcentracionDosis	Integer	Se especifica la concentración por dosis del producto, este campo puede tomar valores nulos.
CantidadDosisBulbo	Integer	Cantidad de dosis por bulbos suministrada.
NumeroLotes	Varchar(30)	Cantidad de lotes involucrados a ese producto.
FormaFarmaceutica	Varchar(30)	Forma en que se presenta el producto puede ser inyectable, oral de uso tópico, entre otras
DescripcionProblema	Varchar(255)	Se describe brevemente el motivo por el cual se originó la devolución.
ConclusionesInvestigacion	Varchar(255)	Se especifican las conclusiones de la investigación.
Decision	Varchar(50)	Se puede destruir, utilizarlo en otras investigaciones o autorizar la comercialización del mismo.
CargoRealizador	Varchar(20)	Cargo de la persona que realizó el registro de retirada.
CargoRevisor	Varchar(20)	Cargo de la persona que revisó el registro de retirada.
Terminado	Boolean	Se especifica si se terminó o no el registro de devolución para un determinado producto.
IdControlPlanilla	Bigint	Llave foránea que la toma de la entidad ControlPlanilla.

Tabla 2.52 Descripción de la tabla ControlPlanilla.

Nombre: ControlPlanilla		
Descripción: En esta tabla se especifican los datos que aprueban todos los procesos que se desarrollan en las diferentes planillas.		
Atributo	**Tipo**	**Descripción**
IdControlPlanilla	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
RealizadoPor	Varchar(30)	Se especifica el nombre de la persona que realiza el proceso.
RevisadoPor	Varchar(30)	Se especifica el nombre de la persona que revisa el proceso.
AprobadoPor	Varchar(30)	Se especifica el nombre de la persona que aprueba el proceso.
DiaRealizacion	Smallint	Día en que se realiza el proceso según la planilla descrita.
MesRealizacion	Smallint	Mes en que se realiza el proceso según la planilla descrita.
AnoRealizacion	Smallint	Año en que se realiza el proceso según la planilla descrita.
DiaRevision	Smallint	Día en que se revisa el proceso según la planilla descrita.
MesRevision	Smallint	Mes en que se revisa el proceso según la

		planilla descrita.
AnoRevision	Smallint	Año en que se revisa el proceso según la planilla descrita.
DiaAprobacion	Smallint	Día en que se aprueba el proceso según la planilla descrita.
MesAprobacion	Smallint	Mes en que se aprueba el proceso según la planilla descrita.
AnoAprobacion	Smallint	Año en que se aprueba el proceso según la planilla descrita.

Tabla 2.53 Descripción de la tabla DocumentosAsociados.

Nombre: DocumentosAsociados		
Descripción: En esta tabla se especifican los documentos que son consultados en la investigación durante todo el proceso de la retirada del producto.		
Atributo	**Tipo**	**Descripción**
IdDocumentosA	Serial	Es el identificador de la tabla, es un valor entero auto incremental, llave primaria.
Documentoss	Varchar(255)	Se especifica el código, descripción y ubicación de todos los documentos que son consultados durante la investigación.
IdRegistroDevoluciones	Bigint	Llave foránea que la toma de la entidad RegistroDevoluciones

Conclusiones

En este capítulo se definió la estrategia a seguir para la integración de todos los módulos, con el fin de obtener la BD del sistema. También se hizo una descripción y fundamentación de la arquitectura de la BD, teniendo en cuenta las necesidades del centro.

Además, se mostró el diagrama de clases persistentes y el modelo de datos; definiendo un total de 26 clases y 27 entidades respectivamente. Se realizó una breve descripción de cada una de ellas, de sus atributos y tipos de datos, facilitando así el entendimiento de los diagramas y de cada una de sus partes.

Capítulo 3
Validación del diseño realizado

En este capítulo se valida el diseño realizado teórica y funcionalmente, utilizando para ello varios parámetros, con el fin de obtener un diseño óptimo. Además, se realiza un análisis de optimización de consultas utilizando el framework Symfony.

3.1 Validación teórica del diseño.

Al modelar una BD, es necesario evitar duplicación de la información, y por ende un mal funcionamiento y exploración de la información, por lo que se deben tener en cuenta varios aspectos como son: la integridad de los datos, el análisis de redundancia de la información, la normalización del diseño, el análisis de la seguridad de la BD y la trazabilidad de las acciones. Esto permitirá controlar los accesos a los datos y que los mismos no se dañen como resultado de acciones no controladas.

3.1.1 Integridad.

La integridad de los datos consiste en garantizar la no contradicción entre los datos almacenados de modo que, en cualquier momento del tiempo, estos sean correctos, es decir, que no se detecte inconsistencia entre los datos. Está relacionada con la minimización de redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia.

El término integridad de datos se refiere a la corrección y completitud de los datos. Cuando los contenidos de una BD se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Ejemplo de esto es cuando se añaden datos no válidos a la base de datos, se modifican datos existentes tomando un valor incorrecto o los cambios pueden perderse debido a un error del sistema o a un fallo en el suministro de energía.

La integridad de datos presenta varias restricciones que posibilitan la declaración y comprobación de condiciones para expresar la consistencia, corrección y exactitud de datos almacenados, estas son: Integridad de dominio, de entidad, referencial, entre otras. [9]

Integridad de dominio

La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y definiciones NOT NULL.

Para realizar estas validaciones PostgreSQL presenta los "obligadores" (constrains), que se encargan de chequear (**CHECK**) el cumplimiento de una o varias condiciones que deben cumplir los campos de una tabla y disparan excepciones, o sea, mensajes de error, en caso de ser violadas.

Para la BD de la SMC se definió la fecha por los campos: día, mes y año, en el que se restringió el dominio del atributo día a un valor entre 1 y 31, pues para que sea un número válido tiene que ser mayor que 0 y menor que 32, en el caso del mes sucede algo parecido el cual puede tomar valores entre 1 y 12, y en el caso del año su dominio estaría en los enteros de 4 dígitos entre 1000 y 9999, rango que aunque es amplio, valida cualquier fecha de la realidad. Otro ejemplo de integridad de dominio sería la cantidad de caracteres válidos admisibles para una cadena de caracteres, en este caso se utiliza un varchar(n), ejemplo de ello sería el *NombreProducto* que es un tipo de dato varchar y solo acepta cadenas de longitud 30, su dominio sería varchar(n)(30). Cuando el tipo de dato es varchar, admite números, letras o caracteres, pero existen campos que es necesario validar los datos que pueden almacenar, pues toman un valor específico, por ejemplo, en el caso anterior el *NombreProducto* ya sea en la tabla *Cierre Queja*, *RegistroDevoluciones*, o en cualquier otra que se utilice solo acepta letras. Lo mismo pasa con los atributos que sólo toman valores numéricos.

Un ejemplo implementado de los "obligadores", en este caso un **CHECK**, para validar que los valores insertados en la fecha están en los rangos correctos sería:

CONSTRAINT notificacionquejas_dia_notificacion **CHECK** ((dianotificacion>0) **AND** (dianotificacion<32));

CONSTRAINT notificacionquejas_mes_notificacion **CHECK** ((mesnotificacion>0) **AND** (mesnotificacion<13));

CONSTRAINT notificacionquejas_año_notificacion **CHECK** ((anonotificacion>1000) **AND** (anonotificacion<9999));

Otro ejemplo es para validar los campos que toman como valor combinaciones de letras y números, en este caso sería:

CONSTRAINT "validar_codigo" **CHECK** ((codigoinspecciones) **::text ~ similar_escape** ('([a-zA-Z]|[0-9])*' **::text, NULL::text**));

Postgre también cuenta con “disparadores” (triggers), los cuales, pueden validar campos de más de una tabla a la vez; disparando una excepción (error), en caso de no cumplirse algunas de las condiciones que valida. Los disparadores hacen uso de funciones especiales, las cuales validan las condiciones que debe cumplir una tupla para ser aceptada como correcta e insertarla en la BD o retornan valor nulo (**NULL**) en caso de violarse alguna condición. Además pueden ser programados para dispararse antes o después de ocurrir eventos de **INSERT**, **UPDATE**, o **DELETE** de una tupla en la tabla. En la BD propuesta se implementaron los trigger para validar la integridad referencial y para registrar la trazabilidad de las acciones.

Otro tipo de integridad de dominio es definir el conjunto de valores que puede tomar un campo lo más específico que se pueda, ejemplo de ello es el estado de un *control de cambio* en el *SIC0816*, el cual puede ser: en revisión, pendiente de documentación o en regulaciones y registros. Esto posibilita que el usuario de acuerdo al cambio que esté analizando no cometa un error que pueda afectar el sistema y viole las reglas de integridad.

Integridad de entidad

La integridad de entidad pretende que cada entidad que se guarda en la BD sea identificable de un modo único, es decir, que evitemos la información redundante. Exige que cada entidad almacenada deba tener una clave principal, pues cada registro de una tabla, debe tener un campo que identifique de modo exclusivo ese registro respecto al resto de registros de esa tabla. Además establece que ningún valor de llave primaria puede ser nulo.

Integridad referencial

La integridad referencial es un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias y datos perdidos.

La integridad referencial hace que el sistema gestor de la base de datos se asegure de que no haya en las claves foráneas valores que no estén en la tabla principal. Se activa en cuanto creamos una clave foránea y a partir de ese momento se comprueba cada vez que se modifiquen datos que puedan alterarla.

Por ejemplo: La tabla *CierreQueja* es una entidad débil de *NotificacionQueja* por lo que se necesita el Id (llave primaria) de *NotificacionQueja* para poder insertar posteriormente un cierre, ya que el Id de la notificación es una llave primaria foránea de la tabla *CierreQueja*, por lo que se hace necesario validar la existencia de la Notificación de Queja antes de insertar o actualizar alguna tupla de la tabla *CierreQueja*.

Otro ejemplo, en la tabla *AccionesCorrectivas* existen como llaves foráneas los identificadores de los distintos registros al que se le puede desarrollar una acción correctiva, uno de ellos es el folio de un Informe de Tendencias, que es el identificador; por tanto al insertar una acción correctiva específicamente para un Informe de Tendencia, se hace necesario chequear que exista previamente ese informe que se desea corregir.

Aunque PostgreSQL valida que se cumpla la integridad referencial, en la BD se hace uso de "disparadores" (triggers) para el tratamiento de errores, que permiten validar la existencia previa de las llaves foráneas e insertar información válida en la BD.

Datos requeridos

Establece que una columna tenga un valor no nulo (NOT NULL). Se define efectuando la declaración de una columna NOT NULL cuando la tabla que contiene las columnas se crea por primera vez.

Ejemplo de ello es en la tabla *SIC0098* el atributo producción es un dato requerido, por lo que se declara NOT NULL, validando que no existan tuplas con esa columna con valor nulo.

Chequeo de validez

Cuando se crea una tabla cada columna tiene un tipo de dato y el SGBD asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

3.1.2 Normalización de la BD.

La normalización no es más que un conjunto de reglas que sirven para ayudar a los diseñadores a desarrollar un esquema que minimice errores lógicos en la manipulación de datos y facilite agregar nuevas columnas sin romper el esquema actual ni las relaciones. Cada regla está basada en la que le antecede.

Cuando se normaliza se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

Cuando se realizó el diseño de la BD para la SMC, existía redundancia de datos debido a que algunas entidades tenían atributos comunes, los cuales almacenaban el mismo tipo de información, por ejemplo los nombres de las personas que realizan, revisan y aprueban cualquier proceso llevado a cabo en una planilla determinada, así como la fecha en que se desarrollaron los mismos; con el objetivo de eliminar esta inconsistencia se creó una nueva tabla *ControlPlanilla* donde se incluyeron estos atributos.

Existen algunos ejemplos puntuales de entidades que tienen atributos comunes, o sea, existe redundancia en un grado menos significativo, pero no cumple objetivo crear una nueva tabla para cada caso, pues se estaría aumentando el tiempo de acceso y actualización a la BD, siendo conveniente almacenar estos valores en entidades independientes

La normalización incluye varios niveles o formas normales que son condiciones que se establecen sobre las relaciones, las cuales al diseñar un sistema y definir la estructura lógica de las relaciones se emplean para quitar de ellas problemas de redundancia y establecer de forma clara las dependencias funcionales entre los atributos en las relaciones. Estas son:

- Primera Forma Normal.
- Segunda Forma Normal.
- Tercera Forma Normal.
- Forma Normal Boyce-Codd
- Cuarta Forma Normal.
- Quinta Forma Normal o Forma Normal de Proyección-Unión.
- Forma Normal de Proyección-Unión Fuerte.
- Forma Normal de Proyección-Unión Extra Fuerte.
- Forma Normal de Clave de Dominio. [10]

Primera Forma Normal

Una relación está en primera forma normal (1FN) si y sólo si todos los dominios son atómicos. Un dominio es atómico si los elementos del dominio son indivisibles. Es decir, no tenemos grupos de repetición o un conjunto de valores repetidos asociados a una misma tupla.

Segunda Forma Normal

Una relación está en segunda forma normal (2FN) si y sólo si está en 1FN y todos los atributos que no sean llaves (ni primarias, ni candidatas) dependen por completo de la llave primaria. Es decir que no existen dependencias parciales.

Tercera Forma Normal

Una relación está en tercera forma normal (3FN) si y sólo si está en 2FN y todos los atributos no llave dependen de manera no transitiva de la llave primaria.

Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica al insertar o eliminar registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no debe haber datos repetidos. Esto provee un esquema limpio, fácil de trabajar y de expandir.

La BD para la SMC está normalizada hasta Segunda Forma Normal ya que provee suficiente nivel de normalización, es decir, inicialmente se parte del cumplimiento de la Primera Forma Normal, pues todos los valores de los atributos de cada tabla son atómicos y no presenta atributos compuestos, ni multivaluados, por ejemplo en la tabla *RegistroDevoluciones* se almacena la fecha en que se notificó el registro que se desea devolver, la cual se registra de forma atómica: día, mes y año en que se realizó el registro, es decir, fecha de notificación: {DiaNotificacion, MesNotificacion, AnoNotificacion}. Existen otras entidades donde también se almacena información referente a la fecha y se registra de forma similar.

En la tabla *NotificacionQueja* se registra la dirección de la institución que realizó la notificación de la queja, este sería un ejemplo de atributo compuesto, que se debería almacenar de forma atómica, pero como nada más se necesita insertar los datos referentes a la dirección sin realizar consultas para tomar valores específicos como calle o número, es ventajoso analizarlo como un atributo indivisible.

Además, cumple con la Segunda Forma Normal (2NF), ya que todas sus tablas están en 1FN y sus atributos no llave dependen únicamente de la llave de la tabla, permitiendo que la ocurrencia de cada fila en la tabla sea única y quede representada por la llave.

Se ha normalizado hasta la 2FN, debido a que utilizar un nivel de normalización muy elevado conduce a una BD más relacional, pero más compleja, es decir, normalizar demasiado puede conducir a tener una BD ineficiente, ya que se obtendría un esquema demasiado complejo para trabajar y resultaría difícil el acceso a los datos desde el contexto orientado a objetos que implementa Symfony, pues se generarían en la capa ORM clases que no representan entidades significativas del negocio, complicando de esta forma el acceso a los datos. Además normalizando hasta la 2FN se controlan la mayoría de los problemas de lógica y se puede insertar un registro sin exceso de datos en la mayoría de las tablas.

3.1.3 Análisis de la seguridad de la BD.

La información almacenada en una BD, está constantemente en peligro de sufrir múltiples riesgos de seguridad, debido a ataques o el acceso no autorizado para modificar, borrar y propagar información, por ello es de vital importancia cuidar la seguridad de la misma, protegiéndola contra accesos no autorizados o cualquier acción que pueda violar la integridad y disponibilidad de los datos o la confidencialidad de los mismos. Otra forma de proteger los datos que se almacenan en la BD, es realizar salvas, de forma tal, que en caso de que se pierda la información se pueda restaurar.

Aunque evitar el intento de acceso no autorizado, ya sea por error del usuario o por mala intención a la BD se hace difícil, se pueden tomar medidas que permitan mantener la consistencia de la información y que sea tan alto el costo de violar la seguridad que frustre casi cualquier intento de acceso no autorizado a la BD. El administrador de bases de datos es el máximo responsable de controlar y garantizar la seguridad global del sistema de bases de datos, tiene la obligación de otorgar o revocar privilegios a cuentas individuales, usuarios o grupos de usuarios y efectuar diferentes tipos de acciones como la creación de cuentas y asignación de niveles de seguridad.

Los SGBD ofrecen múltiples ventajas que facilitan la seguridad de la BD. Primeramente se deben presentar los datos solo a quien esté autorizado, no todos los usuarios pueden visualizar una determinada información, por tal motivo, para que un sistema de base de datos sea confiable debe mantener un grado de seguridad que garantice la autenticación y protección de los datos. PostgreSQL, permite realizar configuraciones para controlar los permisos de los usuarios, utilizando tres niveles de acceso. En un primer nivel (nivel 0) se configuran los permisos de conexión para los host y los usuarios a la o las BD(s), estos datos se recogen en el archivo pg_hba.conf, en el mismo se define que PCs (dirección o direcciones IP) tendrán acceso, además a cuál o cuales BD(s) y el modo en que podrán conectarse los usuarios, que puede ser: conexión sin contraseña, validando el usuario y la contraseña para conectarse, o que rechace cualquier conexión desde el IP o rangos IP y usuarios seleccionados.

Esta comprobación es realizada cada vez que una nueva conexión es solicitada [11]. En un segundo nivel (nivel 1), PostgreSQL permite configurar a qué BD pueden acceder determinados usuarios, utilizando las opciones del archivo pg_ident.conf. El tercer nivel (nivel 2), permite configurar dentro de las BD los accesos a las tablas, utilizando los comandos GRANT y REVOKE para permitir o denegar los permisos, respectivamente.

En la BD propuesta se establecen controles de seguridad para los datos almacenados, garantizando que sólo los usuarios autorizados puedan realizar operaciones correctas sobre algunas tablas o sobre la BD, es decir, los usuarios registrados tienen los permisos restringidos dentro de la BD y pueden tener acceso a la información según el rol que desempeñan. Por la importancia de conservar los datos por un período ilimitado de tiempo, se denegó el permiso para eliminar sobre las tablas que contienen datos de la SMC, dándole la posibilidad de insertar, modificar y actualizar la información necesaria.

Como se hizo alusión inicialmente, para validar que no exista pérdida de información en caso de corrupción de los datos, PostgreSQL permite la realización de salvas (backups) mediante el volcado (dump) de la BD, los cuales pueden realizarse a elección del cliente o de forma automática. Para el volcado de la BD se utiliza el comando pg_dump, el cual indica al SGBD que se va a realizar la salva de la BD. Además brinda una serie de parámetros adicionales que permiten indicar el nombre de la BD, el usuario y contraseña para conectarse a la misma, el nombre que tendrá el archivo de salva y donde se desea ubicar. El volcado de la BD puede realizarse con el servicio Postgre corriendo, lo cual constituye una ventaja, se puede realizar a una o varias BD(s) y además mantiene compatibilidad entre versiones. Sus ineficiencias están marcadas por la lentitud que alcanza con bases de datos de gran volumen de información y el consumo de recursos del sistema.

En caso de que se deba restaurar una salva para su análisis o uso, se utiliza el comando psql, indicándole la salva que se desea restaurar, además de algún otro parámetro que sea necesario introducir.

3.1.4 Trazabilidad de las acciones.

La trazabilidad es la cualidad que permite que todas las acciones realizadas sobre un sistema de tecnología de la información sean asociadas de modo inequívoco a un individuo o entidad. Además, es la capacidad que tiene una organización o sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos [12].

El framework de desarrollo web Symfony utilizado en el desarrollo de la aplicación para el LIMS, guarda mucha información sobre los eventos realizados en archivos de log que se encuentran en el directorio log del proyecto. Por cada evento realizado se genera una nueva línea en el archivo que incluye la fecha y hora a la que se ha producido, el tipo de evento, el objeto que ha sido procesado y otros detalles que dependen de cada tipo de evento y/o objeto procesado. En todos los entornos de trabajo de Symfony, se guardan por defecto los archivos de log, excepto en el entorno de producción.

Cuando se realizan actualizaciones a la BD los archivos de log de Symfony no registran los valores que tenían los datos antes de ocurrir la actualización, y en el módulo SMC se necesita conservarlos para dar cumplimiento a los requisitos funcionales relacionados con registros de trazas. Es por ello que en la BD se crearon las tablas *RegistroTrazas y RegistroModificacion* para almacenar el historial completo de las operaciones realizadas, es decir, los datos necesarios para conocer qué usuario inició la acción y si realizó un cambio en la BD. Además para auditar el momento, se registra la fecha en que se llevó a cabo la acción, cuál fue el campo modificado, y el nuevo valor del mismo.

Un ejemplo de la implementación del trigger para registrar las trazas dando cumplimiento al requisito funcional Registrar traza, aprobado por los analistas sería:

```
declare
id integer;
BEGIN
LOCK TABLE registrotrazas IN EXCLUSIVE MODE;
IF((OLD.foliorl != NEW.foliorl) OR (OLD.periodorevisado != NEW.periodorevisado) OR (OLD.terminado
!= NEW.terminado) OR (OLD.observaciones != NEW.observaciones)) THEN
INSERT INTO registrotrazas (usuario, tablamodificada, fechamodificacion, idtuplamodificada) VALUES
(CURRENT_USER, TG_RELNAME, CURRENT_TIMESTAMP, OLD.foliorl);
    SELECT INTO id MAX (idtraza) FROM registrotrazas;
IF (OLD.periodorevisado!= NEW.periodorevisado) THEN
     INSERT INTO registromodificacion (idtraza, valornuevo, valorviejo, campomodificado) VALUES
(id,NEW.periodorevisado, OLD.periodorevisado,'periodorevisado');
    END IF;
IF (OLD.observaciones!= NEW.observaciones) THEN
     INSERT INTO registromodificacion (idtraza, valornuevo, valorviejo, campomodificado) VALUES
(id,NEW.observaciones, OLD.observaciones,'observaciones');
    END IF;
```

```
IF ((OLD.terminado!= NEW.terminado) AND (NEW.terminado != TRUE)) THEN
INSERT INTO registromodificacion (idtraza, valornuevo, valorviejo, campomodificado)
VALUES (id,'false', 'true','terminado');
END IF;
    IF ((OLD.terminado!= NEW.terminado) AND (NEW.terminado != FALSE)) THEN
    INSERT INTO registromodificacion (idtraza, valornuevo, valorviejo, campomodificado)
VALUES (id,'true','false','terminado');
   END IF;
   END IF;
  RETURN NEW;
END;
```

La trazabilidad es una forma segura de tener un conocimiento certero de los procedimientos realizados en la BD por cada uno de los usuarios que se registran, de forma tal que si ocurre alguna pérdida o modificación de la información se tenga conocimiento de las causas que la originaron.

3.2 Validación Funcional.

3.2.1 Generación de datos para un llenado voluminoso de la BD.

Con el objetivo de obtener un llenado voluminoso de la Base de Datos, y realizarle pruebas a la misma sin violar la integridad de los datos, se empleó la herramienta EMS Data Generator 2005 for PostgreSQL, que permite seleccionar tablas y columnas para generar datos, definir rangos de valores admisibles, así como la cantidad de tuplas que se desea generar. El empleo de esta herramienta posibilitó chequear de forma automática la integridad referencial de la BD.

Se generaron para las pruebas a la BD un determinado volumen de información en diferentes tablas, de forma tal que simulara el flujo de información en el módulo SMC, en un período de 5 años. Es necesario tener presente que para las tablas que incluyen llaves foráneas el software hace una búsqueda en el catálogo de llaves para realizar las validaciones.

El resultado obtenido se muestra en la siguiente tabla:

Tabla 3.1 Descripción de la cantidad de datos generados para las tablas de la BD.

Tablas	Volumen de dato

LotesAsociados, PersonasContactadas y LugaresCantidadesRetiradas.	30 000
ControlPlanilla, NotificacionQuejas, RetiradaProductosMercado y Documentos.	20 000
SIC0963, ParticipantesComision y DocumentosAsociados.	10 000
Para las restantes tablas.	5000

Con el objetivo de probar el funcionamiento correcto de la BD, se realizaron consultas, teniendo en cuenta los resultados y el tiempo de ejecución de las mismas.

Una de las consultas realizadas a la BD fue a la tabla *RetiradaProductosMercado*, con el objetivo de conocer los productos que se retiraron en una fecha determinada, ordenándolos por la cantidad de bulbos liberados. Es necesario tener presente que la retirada se inicia cuando se recibe una queja, debido a que el producto no es óptimo o no cumple las condiciones requeridas.

```
SELECT
  public.retiradaproductosmercado.idretiradaproductosmercado,
  public.retiradaproductosmercado.producto,
  public.retiradaproductosmercado.diainicioretirada,
  public.controlplanilla.realizadopor,
  public.controlplanilla.revisadopor,
  public.cierrequeja.reportadoanteriormente,
  public.cierrequeja.codigoquejaanterior,
  public.retiradaproductosmercado.totalbulbosliberados,
  public.notificacionquejas.formafarmaceutica,
  public.notificacionquejas.lotesinvolucrados,
  public.notificacionquejas.mesnotificacion

FROM
 public.retiradaproductosmercado
```

```
 INNER JOIN public.controlplanilla ON
(public.retiradaproductosmercado.controlplanillaidcontrolplanilla=public.controlplanilla.idcontrolplanilla)
 INNER JOIN public.cierrequeja ON
(public.retiradaproductosmercado.cierrequejaidcierrequeja=public.cierrequeja.idcierrequeja)
INNER JOIN public.notificacionquejas ON
(public.cierrequeja.notificacionquejasidnotificacionqueja=public.notificacionquejas.idnotificacionqueja)
WHERE
  (diainicioretirada BETWEEN '5' AND '25') AND
  (mesnotificacion >= '3')
ORDER BY
  totalbulbosliberados
```

Para el ejemplo anterior, realizando una búsqueda secuencial, el resultado obtenido fue de 11220 filas en 5.15 segundos. Después de haber realizado los índices de tipo B-tree a las tablas, especificando los criterios de búsqueda definidos en los requerimientos funcionales desarrollados por el analista, se obtuvo el mismo resultado en 2 segundos 157 milisegundos, demostrando que la búsqueda fue más óptima que la realizada inicialmente.

Otro ejemplo sería:

```
SELECT
  public.documentos.iddocumentos,
  public.documentos.codigo,
  public.documentos.descripcion,
  public.cierrequeja.nombreproducto,
  public.cierrequeja.reportadoanteriormente,
  public.notificacionquejas.formafarmaceutica,
  public.controlplanilla.realizadopor,
  public.documentos.dia,
  public.notificacionquejas.lotesinvolucrados
FROM
 public.documentos
 INNER JOIN public.cierrequeja ON
(public.documentos.cierrequejaidcierrequeja=public.cierrequeja.idcierrequeja)
```

INNER JOIN public.notificacionquejas **ON**
(public.cierrequeja.notificacionquejasidnotificacionqueja=public.notificacionquejas.idnotificacionqueja)
INNER JOIN public.controlplanilla **ON**
(public.notificacionquejas.controlplanillaidcontrolplanilla=public.controlplanilla.idcontrolplanilla)
WHERE
(dia <= '20') **AND**
(reportadoanteriormente = true)

El resultado de esta consulta en una búsqueda secuencial fue de 6935 filas, que representa los documentos que son consultados en un proceso de cierre de queja, se realizó en aproximadamente 2.01 segundos. Después de haber realizado los índices de tipo B-tree a las tablas, se obtuvo el mismo resultado en 906 milisegundos, demostrando que la búsqueda fue más óptima que la realizada inicialmente.

Las entidades que según el negocio debían tener como identificador una cadena de caracteres, se les asignó un identificador autoincremental, con el objetivo de optimizar el tiempo de ejecución de las consultas, de forma tal que cuando se realice el acceso a una determinada tabla el tiempo de respuesta sea mínimo. En algunas tablas como *RetiradaProductosMercado,* a cada producto se le asigna un folio determinado; para garantizar que no se repita la combinación de estos dos campos se realizó un índice que permite que la unión de ellos sea única, dando la posibilidad de que se puedan insertar varios folios para un mismo producto y que varios productos tengan un mismo folio. Cumpliendo así el negocio modelado por los analistas.

Ejemplo de esto:

Tabla 3.2 Datos de prueba del índice compuesto (producto-folio).

Combinación de campos (producto-folio)	Válido para la inserción
Penicilina-01	Si
Gentamicina-01	Si
Penicilina-02	Si
Gentamicina-01	No

La generación de datos de pruebas y las consultas realizadas a la BD se ejecutaron en un tiempo óptimo de acuerdo a las características que presenta la computadora utilizada: PC Pentium (R) 4 CPU 3.00 GHz, 512 MB de RAM; cuando se utilice la BD con el flujo real de información diario del módulo,

este tiempo de ejecución debe ser mejor, teniendo en cuenta las propiedades que presenta el servidor de BD utilizado en el centro.

Las pruebas realizadas a una BD, nunca pueden tomarse como resultados reales, aunque dan un aproximado de como sería el funcionamiento de la misma. La implantación y utilización de una BD, está marcada por varios factores: cantidad de usuarios a conectarse y el grado de concurrencia de las solicitudes hechas por los mismos. Además, es necesario velar el cumplimiento de todos los requerimientos no funcionales propuestos para la BD, ya que validan el funcionamiento correcto y el máximo rendimiento de la misma. De aquí que, aunque las pruebas realizadas brinden resultados que puedan parecer alentadores, no pueden ser motivo de confianza, todo lo contrario, la mejor prueba que se le puede realizar a cualquier resultado informático lo constituye la interacción directa con el usuario.

Luego de las pruebas realizadas se obtuvo como resultado el cumplimiento de los requisitos funcionales propuestos por los analistas.

3.2.2 Análisis de optimización de consultas.

Las consultas son solicitudes que se realizan a la BD, para almacenar, visualizar y actualizar los datos. Una consulta puede expresarse de varias maneras, donde cada una propone una forma diferente de hallar el resultado, de ahí que unas sean más óptimas que otras.

El acceso a los datos de la BD del LIMS se implementará mediante objetos, utilizando la capa ORM de Propel que transforma las llamadas a los objetos en consultas SQL optimizadas para el sistema gestor de bases de datos que se está utilizando y la capa de abstracción de la BD Creole que proporciona una interfaz entre el código PHP y el código SQL de la base de datos.

Las clases objeto base que genera Symfony para cada tabla, contienen una serie de constructores y accesores por defecto en función de la definición de cada columna: los métodos new(), getColumna() y setColumna() permiten crear y obtener las propiedades de los objetos. Estos métodos utilizan una variante del estándar de codificación camlCase aplicada al nombre de las columnas. Para guardar los datos en la base de datos, se debe invocar el método save() del objeto, que se traduce en una sentencia INSERT si no existía el objeto en la base de datos y en una sentencia UPDATE si ya existía.

Cuando se quiere obtener más de un registro de la BD, se puede utilizar el método doSelect() de la clase *peer c*orrespondiente a los objetos que se quieren obtener. El primer parámetro de este método es un objeto de la clase Criteria, que se utiliza para definir consultas simples sin utilizar SQL, para conseguir la abstracción de la base de datos.

Invocar el método doSelect() es más potente que ejecutar una consulta SQL, pues optimiza el código SQL, devuelve un arreglo de objetos, todos los valores pasados a Criteria se escapan antes de insertarlos en el código SQL para evitar la inyección SQL. Además la capa ORM crea y rellena automáticamente los objetos en función del result set de la base de datos, llamándose este proceso *hidratación.*

Este proceso requiere una cantidad de tiempo directamente proporcional al número de resultados que devuelve la consulta. Debido a esto, las funciones del modelo deberían optimizarse para devolver solamente los resultados necesarios. Si no se necesitan todos los resultados devueltos, debe refinarse el objeto Critreria mediante los métodos setLimit() y setOffset() o automatizarse esta acción utilizando un paginador. El objeto sfPropelPager gestiona de forma automática los valores offset y limit para una consulta Propel, de forma que solamente se crean los objetos mostrados en cada página.

Para las selecciones más complejas de objetos, se necesitan equivalentes a las sentencias WHERE, ORDER BY, GROUP BY y demás de SQL. El objeto Criteria dispone de métodos y parámetros para indicar todas estas condiciones.

También es importante controlar el número de consultas que se realizan a la base de datos por cada petición para optimizar el código. Si se utiliza el método doSelect() para ejecutar una consulta que involucra campos de dos tablas relacionadas, posterior a la ejecución del método se debe hacer una consulta a la BD por cada objeto del que se necesite buscar datos de la tabla relacionada. Para obtener estos datos utilizando una única consulta se deben utilizar Joins. El método doSelectJoinXXX() realiza una consulta más compleja que el doSelect() donde las columnas adicionales que están presentes en el resultado obtenido permiten a Propel *"hidratar"* los objetos que hacen referencia a las tablas relacionadas. Este método se genera automáticamente cuando se ejecuta la tarea propel-build-model, para cada clave externa en las clases peer.

El resultado devuelto al emplear lo métodos doSelect() y doSelectJoinXXX() es el mismo array de objetos. La diferencia se hace evidente cuando se utiliza un método *getter* asociado con una clave externa. En el caso del método doSelect(), se realiza una consulta a la base de datos y se crea un nuevo objeto con el resultado; en el caso del método doSelectJoinXXX(), el objeto asociado ya existe y no se realiza la consulta con la base de datos, por lo que el proceso es mucho más rápido. [5]

Evidentemente, la llamada al método doSelectJoinXXX() es un poco más lenta que la llamada a un método simple doSelect(), por lo que solamente mejora el rendimiento global de la página si se utilizan los objetos relacionados. [5]

En ocasiones es necesario realizar una consulta que involucre campos de varias tablas, donde la cantidad de registros a recuperar es muy grande, para estos casos es preferible obtener directamente el resultado y no obtener los objetos porque los tiempos de respuestas serían demasiado altos debido al proceso de hidratación. Bajo estas circunstancias resultaría desventajoso utilizar la ORM (Propel), pero es recomendable usar la capa de abstracción de la BD (Creole) pues se encarga de mantener la seguridad de la base de datos y el buen rendimiento y portabilidad de la aplicación, mientras que si se utilizan instrucciones PHP que accedan directamente a la base de datos se corre el riesgo de sufrir ataques de inyección SQL.

La implementación de las consultas a la BD en Symfony la realizarán los programadores del proyecto, a partir de la capa ORM generada en este trabajo. Se propone realizar un análisis de las vías más óptimas de acceso a los datos para obtener mejores tiempos de respuestas a las solicitudes.

Conclusiones

En este capítulo se realizó la validación teórica y funcional del diseño propuesto para la BD.

Para la validación teórica se tuvieron en cuenta varios aspectos como son: la integridad de los datos, la redundancia de información, la cual se eliminó normalizando la BD, llevándola hasta la segunda forma normal (2FN), pues es la solución más óptima para el framework utilizado. Además se analizó la seguridad de la BD, limitando los permisos de los usuarios en dependencia del rol que desempeñen y realizando copias de seguridad, para recuperar los datos en caso de pérdida. Para lograr el control de las operaciones de los usuarios en la BD en cada momento se analizó la trazabilidad de las acciones, almacenando para ello los datos necesarios en las tablas *RegistroTrazas y RegistroModificacion* de la BD.

Se realizó un llenado voluminoso de la BD con la herramienta EMS Data Generador For PostgreSQL, para la realización de pruebas de carga intensiva y la validación del cumplimiento de los requerimientos funcionales.

Se realizó un estudio sobre la optimización de las consultas a la BD desde el framework de desarrollo de aplicaciones web Symfony.

Conclusiones

El almacenamiento de toda la información de la SMC está en formato duro, únicamente los datos más significativos se transcriben a formato digital, imposibilitando a los trabajadores el acceso y manejo de datos.

Con la realización de este trabajo se cumplieron los objetivos propuestos:

- Se diseñó el Modelo de Datos para la BD, el cual consta de 27 tablas. Para realizar este diseño se seleccionó el modelo relacional, utilizando como herramienta de modelado el Visual Paradigm y como SGBD PostgreSQL, permitiendo almacenar y manejar el gran volumen de información generada en la SMC del CIGB.
- Se implementaron las funcionalidades y los disparadores de la BD que garantizan el buen funcionamiento de la misma.
- Se validó teórica y funcionalmente el diseño de la BD, teniendo en cuenta la integridad, redundancia, normalización y seguridad de la información. También se analizó la trazabilidad de las acciones, registrando las acciones a realizarse en la BD. Además, se realizaron pruebas para el llenado voluminoso de la BD a través de la utilización de la herramienta EMS Data Generator 2005 for PostgreSQL, así como la implementación de consultas, para verificar el funcionamiento de la misma, obteniendo como resultado el cumplimiento de todos los requisitos funcionales planteados por los analistas.

Por todo lo antes expuesto se concluye que todos los objetivos enunciados en la investigación se cumplieron satisfactoriamente.

Recomendaciones

Dado el volumen de información que se genera en la SMC en el CIGB, y que esta no puede ser eliminada porque constantemente es consultada para realizar auditorias y establecer estadísticas; se recomienda:

- La elaboración de un almacén de datos, (del inglés *data warehouse*) que es una colección de datos, integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la empresa u organización en la que se utiliza. El almacenamiento de los datos no debe usarse con datos de uso actual, por lo que se debe realizar cuando la información almacenada en la BD tiene un período mayor de 5 años.

Referencias Bibliográficas

1. **Díaz, Elennis y Nieto, Liusmila.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis de la Sección de Mejoramiento de la Calidad y del Grupo de Desarrollo.* Cuidad Habana : s.n., 2007.

2. Selección de Aplicaciones Software para la industria farmacéutica. [En línea] [Citado el: 2008 de febrero de 12.] http://www.farmaindustrial.com/articulos/articulospdf/software/Addlink.pdf.

3. **Raga, Charlis.** Base de Datos. [En línea] [Citado el: 20 de febrero de 2008.] http://www.monografias.com/trabajos7/bada/bada.shtml

4. **Roso, Jhon, Rivas, Norelis y Ramírez, Mercedes.** Base De Datos. [En línea] 30 de julio de 2003. [Citado el: 2008 de febrero de 8.] http://www.monografias.com/trabajos14/basededatos/basededatos.shtml.

5. **Potencier, Fabien y Zaninotto, Francois.** *Symfony la guía definitiva.* 2007.

6. **Rumbaugh, James, Booch, Grady y Jacobson, Ivar.** *El proceso unificado de desarrollo de software. Volumen 1.* Ciudad Habana, Cuba : Félix Varela, 2004.

7. Teleformación.uci.cu. [En línea] 16 de semptiembre de 2007. [Citado el: 15 de enero de 2008.] http://teleformacion.uci.cu/mod/resource/view.php?id=6655.

8. EMS SQL Manager 2005 for PostgreSQL 3.6. [En línea] 9 de junio de 2006. [Citado el: 2008 de febrero de 8.] http://www.download3000.com/download_5711.html.

9. **Domínguez, Arodys y Miranda, Duniel.** *Sistema de Manejo de Datos de Ensayos Clínicos:Diseño e implementación de la Base de Datos.* Ciudad Habana : s.n., 2007.

10. Normalización de las Bases de datos. [En línea] [Citado el: 20 de mayo de 2008.] http://www.trucostecnicos.com/trucos/ver.php?id_art=278.

11. **Worsley, John y Drake, Joshua**. [En Línea] 2001 [Citado el: 20 de mayo de 2008.] http://www.sobl.org/traducciones/practical-postgres/node40.html.

12. **Cano, Jeimy J.** [En línea] 2005 [Citado el: 20 de mayo de 2008.] http://www.isaca.org/Template.cfm?Section=Home&CONTENTID=24655&TEMPLATE=/ContentManagement/ContentDisplay.cfm.

Bibliografía

1. **Alvarez, Sara.** Desarrollo Web: Modelos de bases de datos. [En línea] [Citado el: 13 de febrero de 2008.] http://www.desarrolloweb.com/articulos/modelos-base-datos.html.

2. Ayuda extendida del Rational Rose Enterprise Edition 2003. [Citado el: 20 de febrero de 2008.]

3. Conceptos básicos de integridad referencial. [En Línea] julio 2000. [Citado el: 21 de mayo de 2008.] http://www.aulaclic.es/sql/b_8_1_1.htm.

4. Curso de Introducción a SQL Server 2005. [En Línea] [Citado el: 21 de mayo de 2008.] http://www.adrformacion.com/cursos/sqlserver/leccion2/tutorial3.html.

5. **Decker, Hendrik.** Tipos de Restricciones de Integridad en Bases de Datos Relacionales. [En Línea] [Citado el: 21 de mayo de 2008.] http://web.iti.upv.es/actualidadtic/2004/06/2004-06-SSQL.pdf.

6. **De Girolami, Alejandro.** Las Bases de datos. [En Línea] 30 de abril de 2007. [Citado el: 20 de mayo de 2008.] http://www.techtear.com/2007/04/30/las-bases-de-datos.

7. **Kuroki, Christian.** PostgreSQL 8. [En línea] [Citado el: 15 de febrero de 2008.] http://www.dbrunas.com.ar/postgres/migrapg.pdf.

8. La herramienta de productividad para cualquier laboratorio. [En línea] [Citado el: 13 de febrero de 2008.] http://www.automatica.es/imprimirpag.asp?pid=472.

9. **Marín Illera, Álvaro.** MySQL. [En línea] [Citado el: 8 de febrero de 2008.] http://www.e-ghost.deusto.es/docs/TutorialMySQL.html.

10. **Marqués Andrés, María Mercedes.** Arquitectura de los sistemas de bases de datos . [En línea] [Citado el: 2008 de abril de 8.] http://www3.uji.es/~mmarques/f47/apun/node33.html.

11. **Mato García, Rosa María.** *Sistemas de Bases de Datos.* Ciudad Habana, Cuba : Pueblo y Educación, 2005.

12. Normalización de bases de datos. [En Línea] [Citado el: 20 de mayo de 2008.] http://www.emagister.com/normalizacion-bases-datos-cursos-328281.htm#programa.

13. **Potencier, Fabien y Zaninotto, Francois.** *Symfony la guía definitiva.* 2007.

14. **Ruiz Tijerina, Martha Esthela.** Bases de Datos . [En línea] [Citado el: 9 de febrero de 2008.] http://www.monografias.com/trabajos5/tipbases/tipbases.shtml.

15. **Rumbaugh, James, Booch, Grady y Jacobson, Ivar.** *El proceso unificado de desarrollo de software. Volumen 1.* Ciudad Habana, Cuba : Félix Varela, 2004.

16. Seguridad en bases de datos. [En Línea] [Citado el: 21 de mayo de 2008.] http://html.rincondelvago.com/seguridad-en-bases-de-datos.html.

17. **Trejo Martinez, Janhil Aurora.** Base de Datos. [En línea] [Citado el: 9 de febrero de 2008.] http://www.monografias.com/trabajos11/basda/basda.shtml.

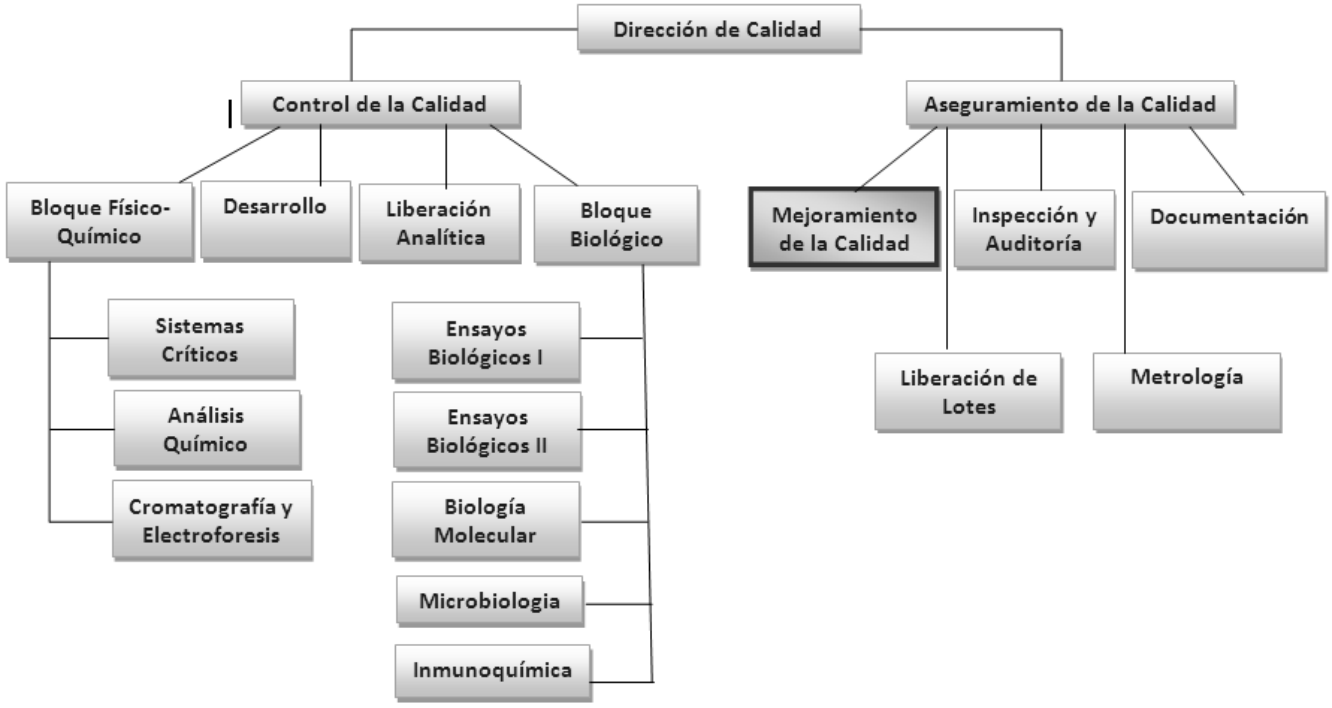
18. **Valera, Isabel.** Programación Orientada a Objetos, Oracle y Sql Server. [En línea] [Citado el: 12 de febrero de 2008.] http://www.monografias.com/trabajos4/basesdatos/basesdatos.shtml.

19. Ventajas de PostgreSQL . [En línea] [Citado el: 13 de febrero de 2008.] http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.

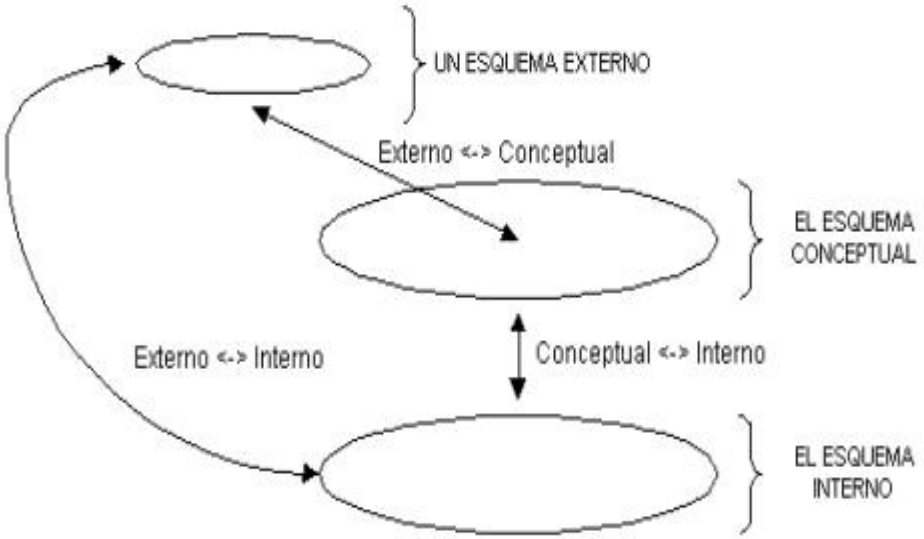
20. **Vizcaíno, Aurora, García, Felix Óscar y Cabal, Ismael.** Una Herramienta CASE para ADOO: Visual Paradigm. [En línea] [Citado el: 15 de febrero de 2008.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

ANEXOS

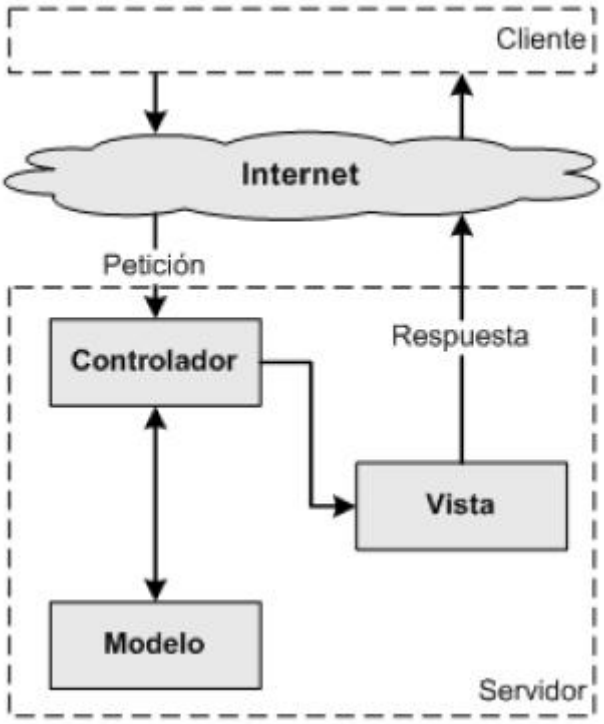
Anexo 1. Organigrama del Departamento de Calidad del CIGB.



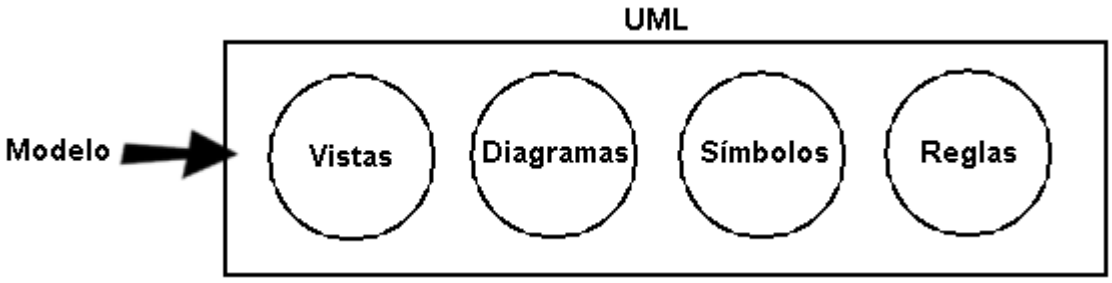
Anexo 2. Arquitectura de tres niveles de la BD.



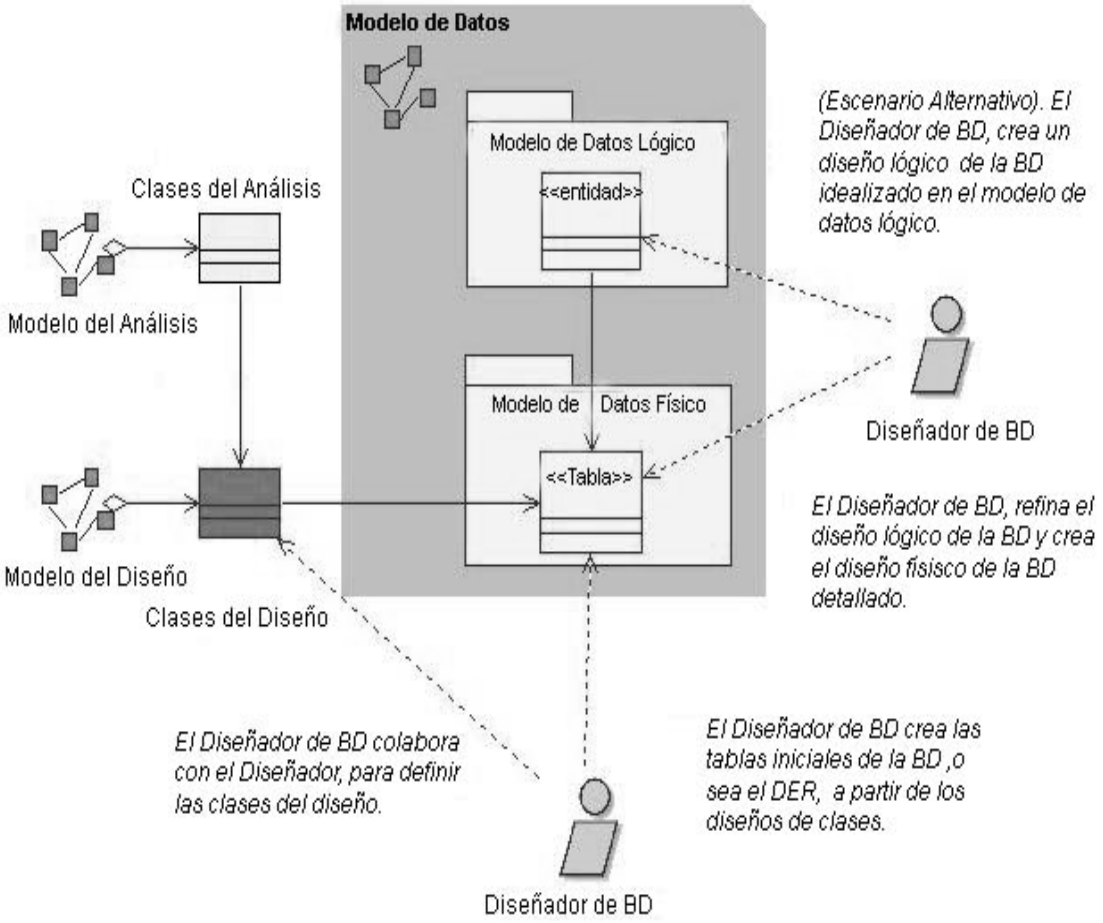
Anexo 3. Patrón de arquitectura Modelo-Vista-Controlador (MVC).



Anexo 4. Representación gráfica del Lenguaje de Modelado.



Anexo 5. Artefacto a realizar.



Anexo 6. Diagrama de clases persistentes.

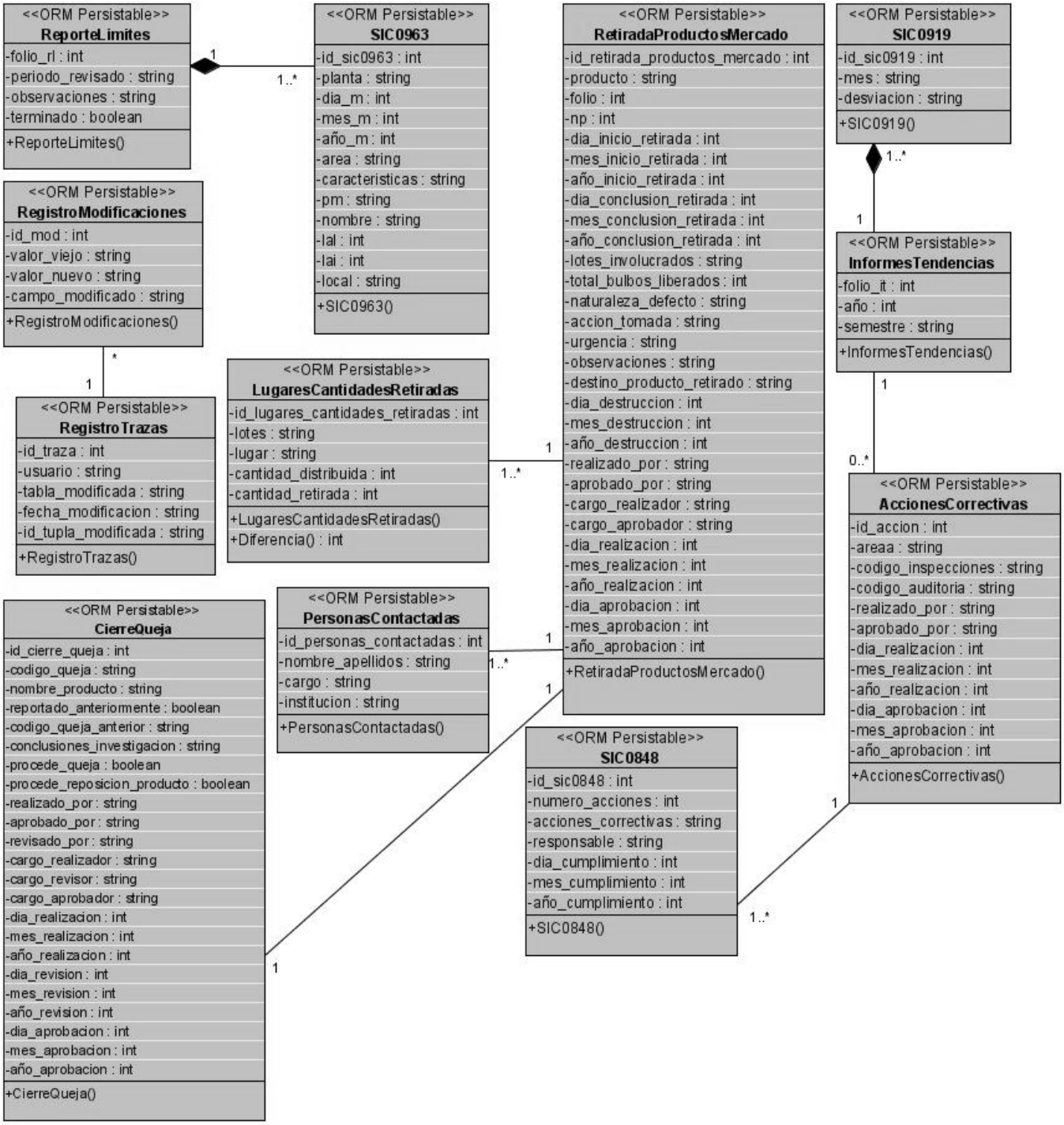


Diagrama de Clases Persistentes (Parte I)

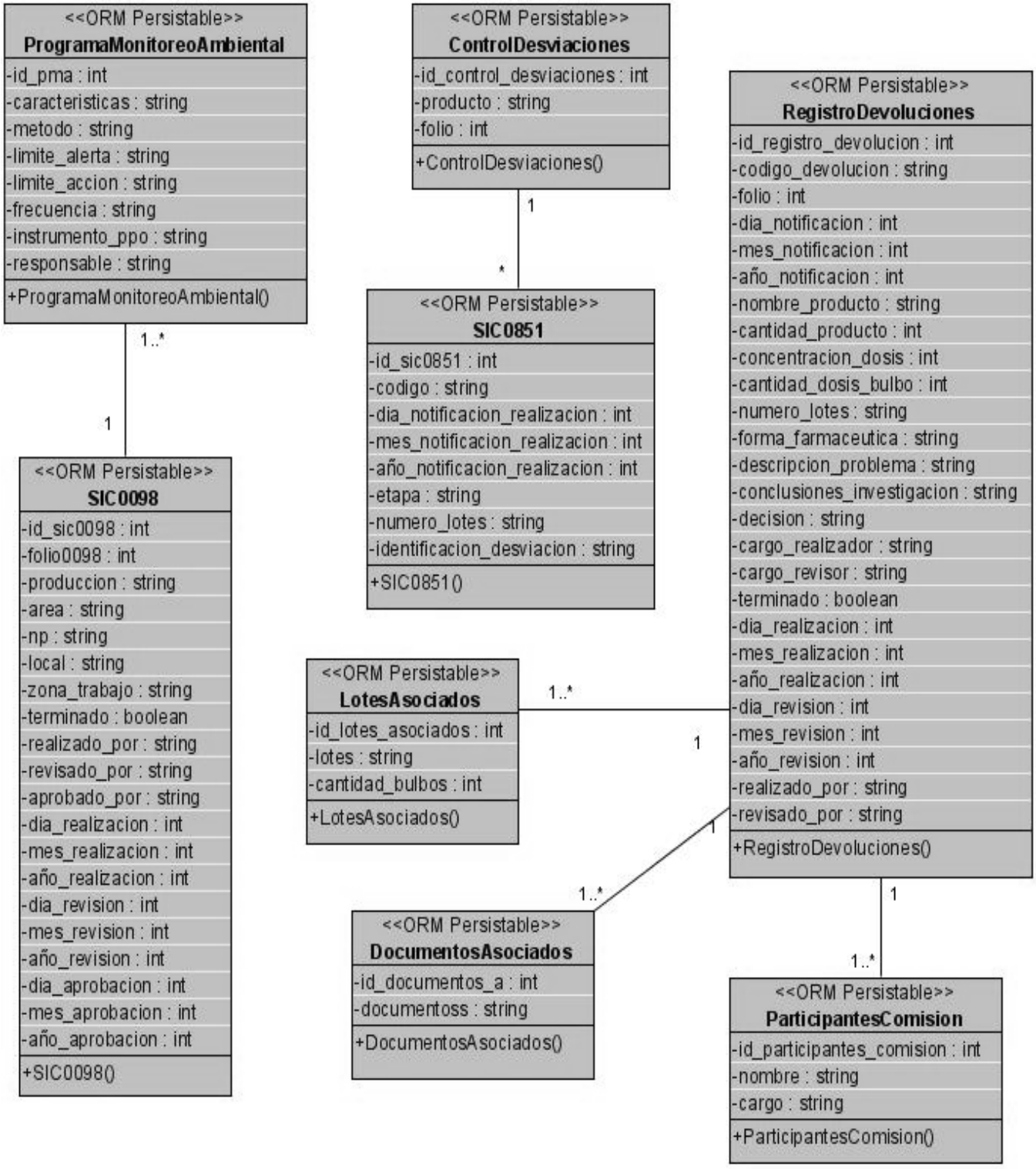


Diagrama de Clases Persistentes (Parte II)

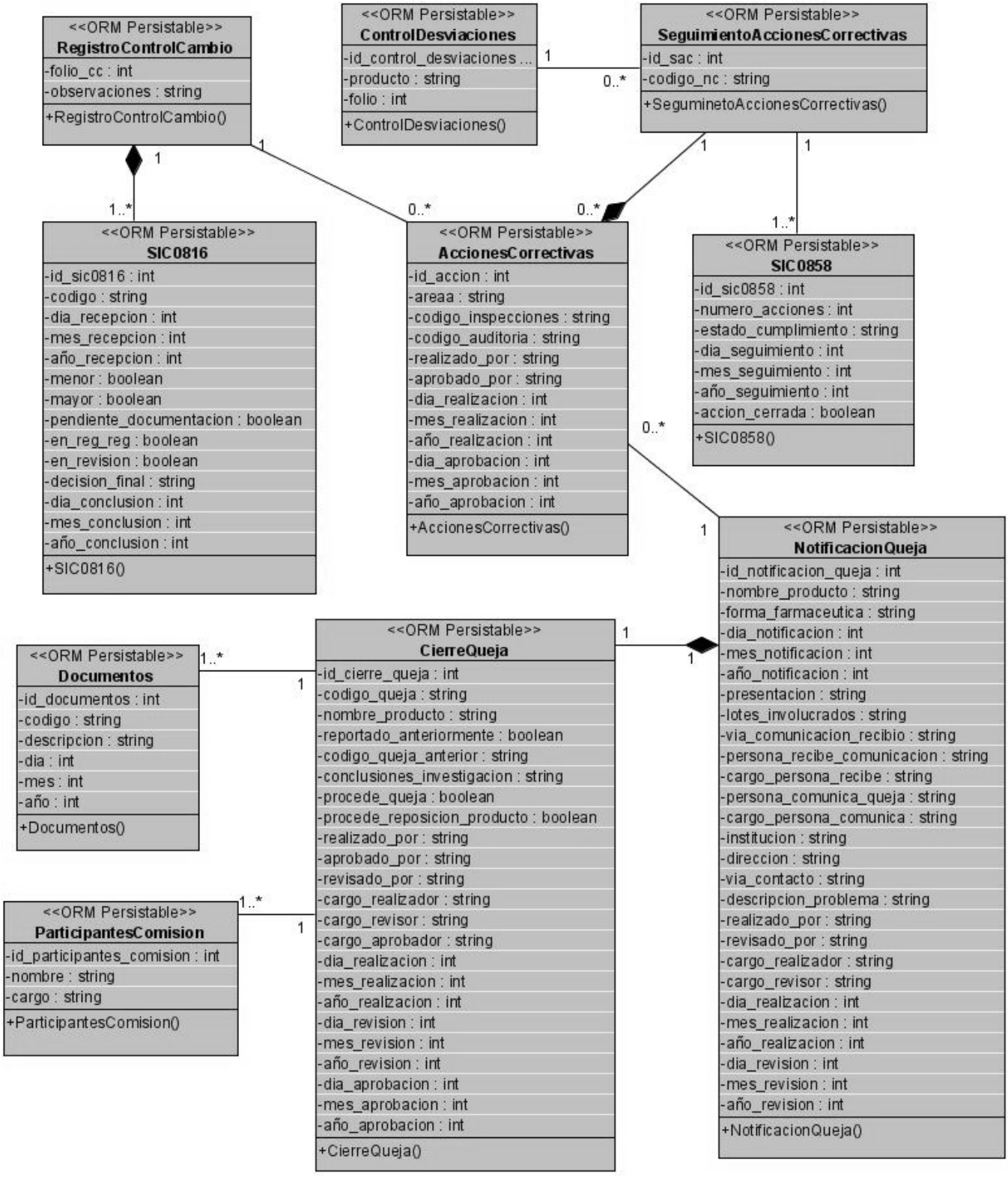
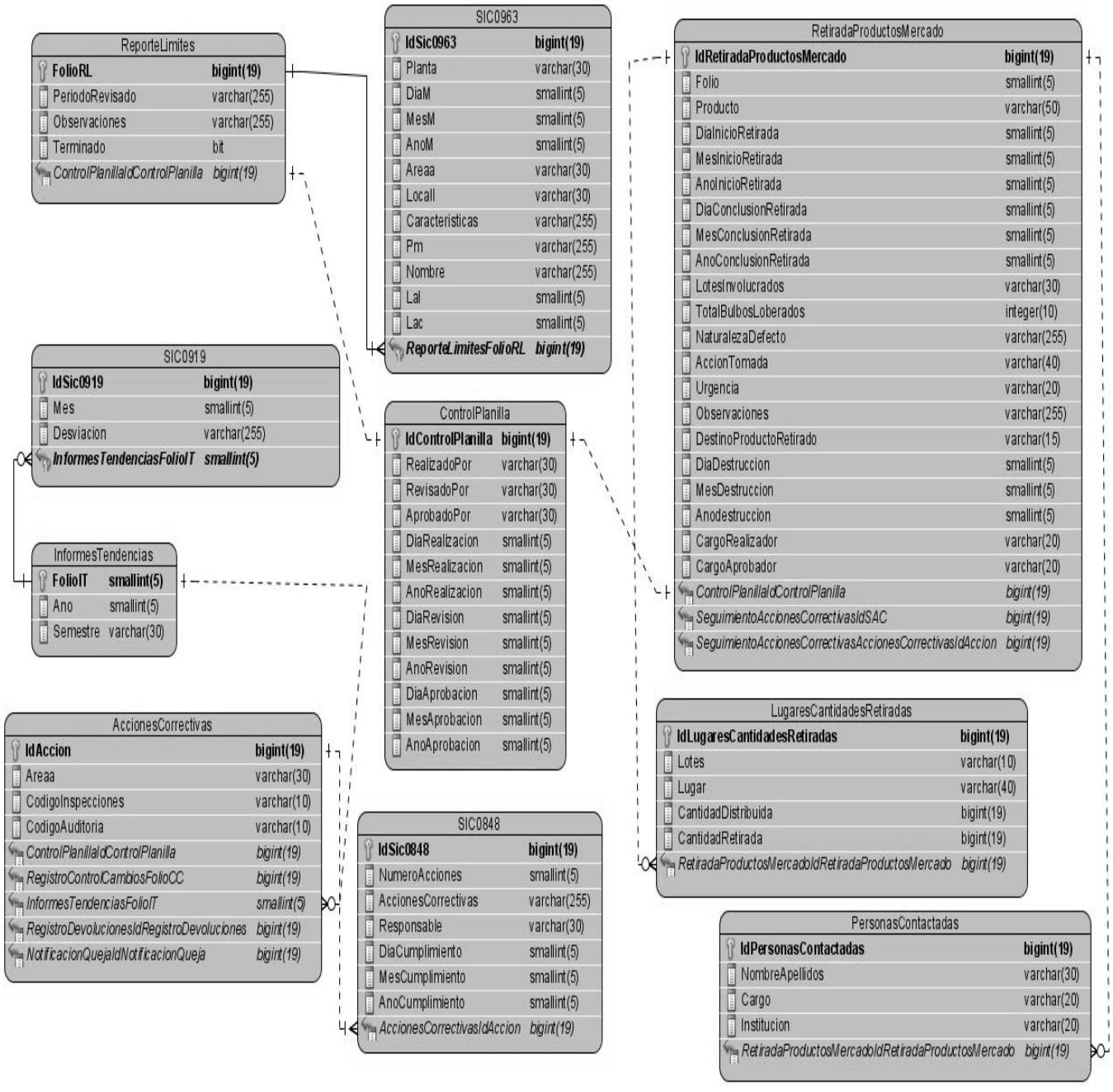
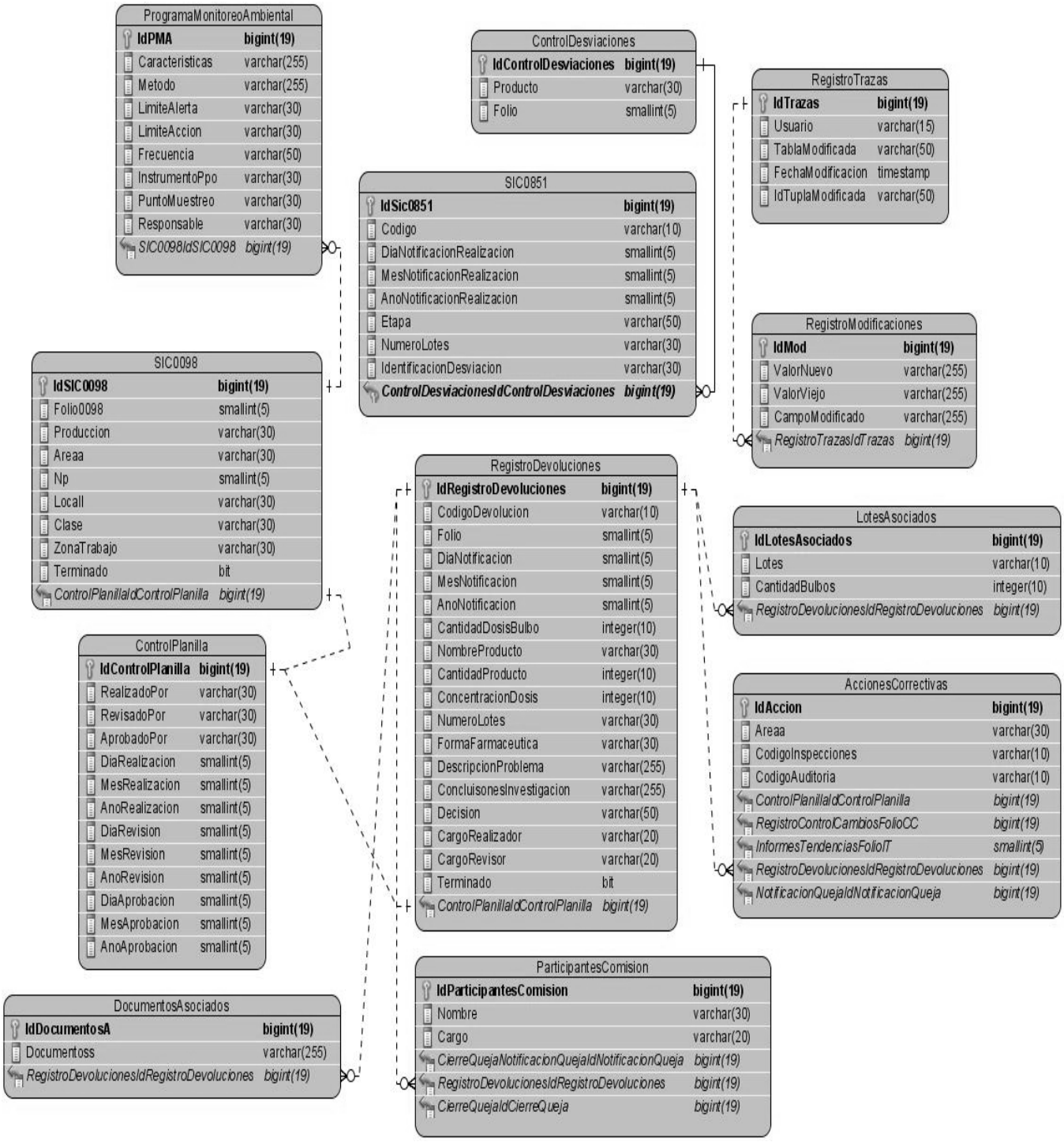


Diagrama de Clases Persistentes (Parte III)

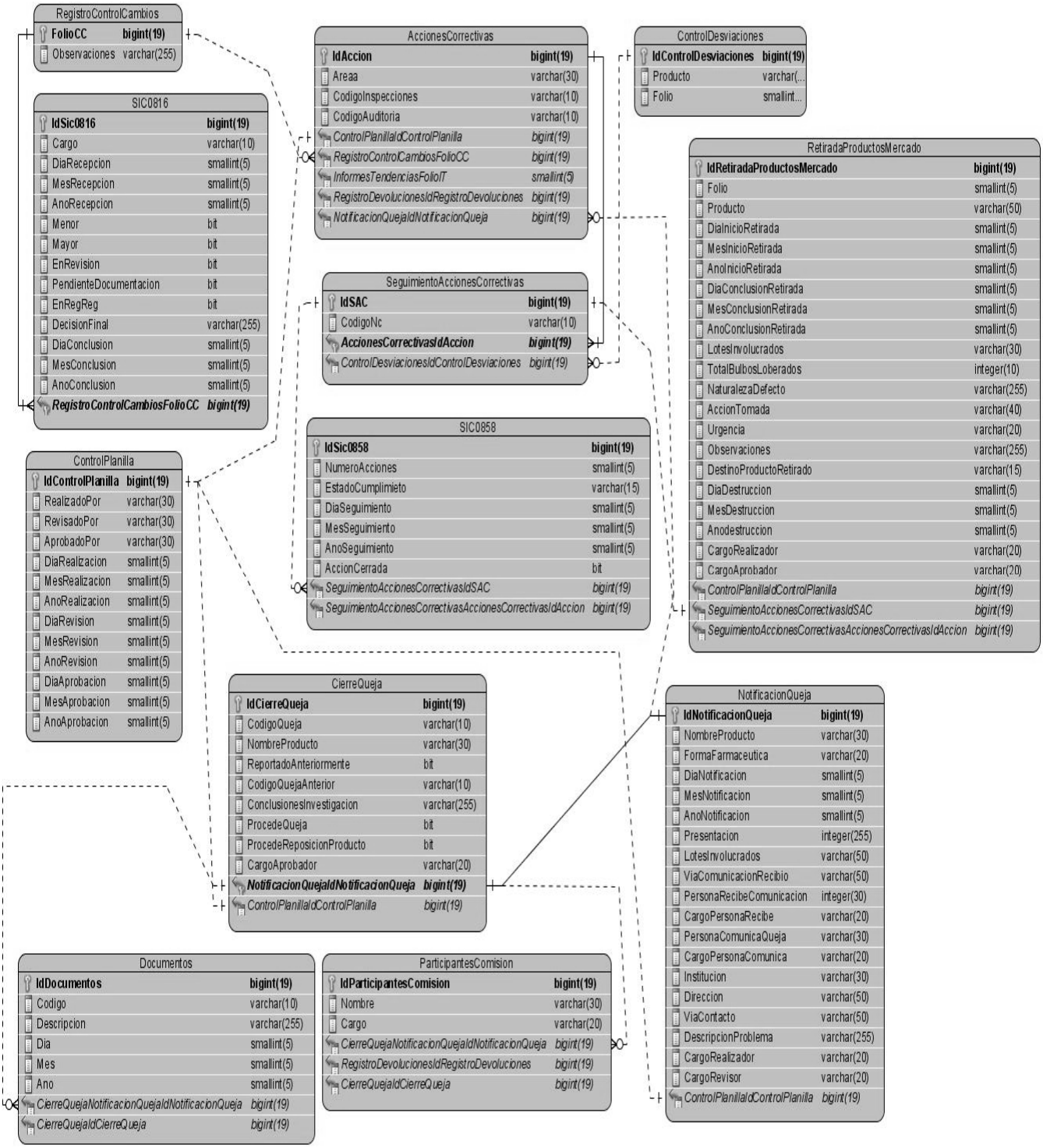
Anexo 7. Modelo de Datos.



Modelo de datos (Parte I)



Modelo de Datos (Parte II)



Modelo de datos (Parte III)

Glosario de términos

MSDE: Microsoft SQL Server Desktop Engine es un motor de base de datos que ofrece a los desarrolladores nuevas oportunidades para crear soluciones que se pueden distribuir libremente, facilitando una migración sencilla hacia la tecnología SQL Server.

Tupla: Se define como una función finita que mapea (asocia unívocamente) los nombres con algunos valores.

API: *(Application Programming Interface).* Interfaz de Programación de Aplicaciones, conjunto de que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

GNU GPL: Licencia Pública General creada por la Free Software Foundation. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Backup: Copia de seguridad o de respaldo (backup en inglés). Copias de datos de tal forma que estas copias adicionales puedan restaurar un sistema después de una pérdida de información.

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.

BD Híbrida: 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.

Clúster: Es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio.

Licencia 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 y permite el uso del código fuente en software no libre.

Precisión arbitraria: Es un método que permite la representación, en un programa informático, de números ya sean enteros ó racionales con tantos dígitos de precisión como cuanto sea deseado y además posibilita la realización de operaciones aritméticas sobre dichos números.

Triggers: Es un evento que se ejecuta en una base de datos cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

MVCC: Acceso concurrente multiversión. Es un sistema que ofrece el SGBD Postgre para controlar la concurrencia.

Lenguajes Procedurales: Los lenguajes procedurales están fundamentados en la utilización de variables para almacenar valores y en la realización de operaciones con los datos almacenados.

Framework: Es una estructura de soporte, utilizada en el desarrollo de software, sobre la cual pueden organizarse y desarrollarse proyectos. Representa una arquitectura de software y provee una estructura y una metodología de trabajo.

Creole: Capa de abstracción de bases de datos para PHP5.

Propel: Es un servicio de objeto persistente y de consulta que provee un sistema para almacenar objetos en una base de datos y un sistema para búsqueda y restauración de objetos desde una base de datos.

IDEs: Un entorno de desarrollo integrado o en inglés Integrated Development Environment es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

EMS (Electronic MicroSystems): Es una Compañía de Tecnología de la Información, centrada en el desarrollo de software. Inicialmente se especializó en el desarrollo de aplicaciones de red, BD corporativas y en la construcción de herramientas de automatización de negocios en arquitectura cliente-servidor multi-capa.

Dominio: Es un conjunto de valores que puede tomar un atributo en una relación.

Log: Registro de eventos durante un periodo de tiempo en particular.