Universidad de las Ciencias Informáticas
Facultad 6

Título: "Solución informática para la gestión de la información del sistema alasARBOGEN 2.0"

Trabajo de Diploma para optar por el título de

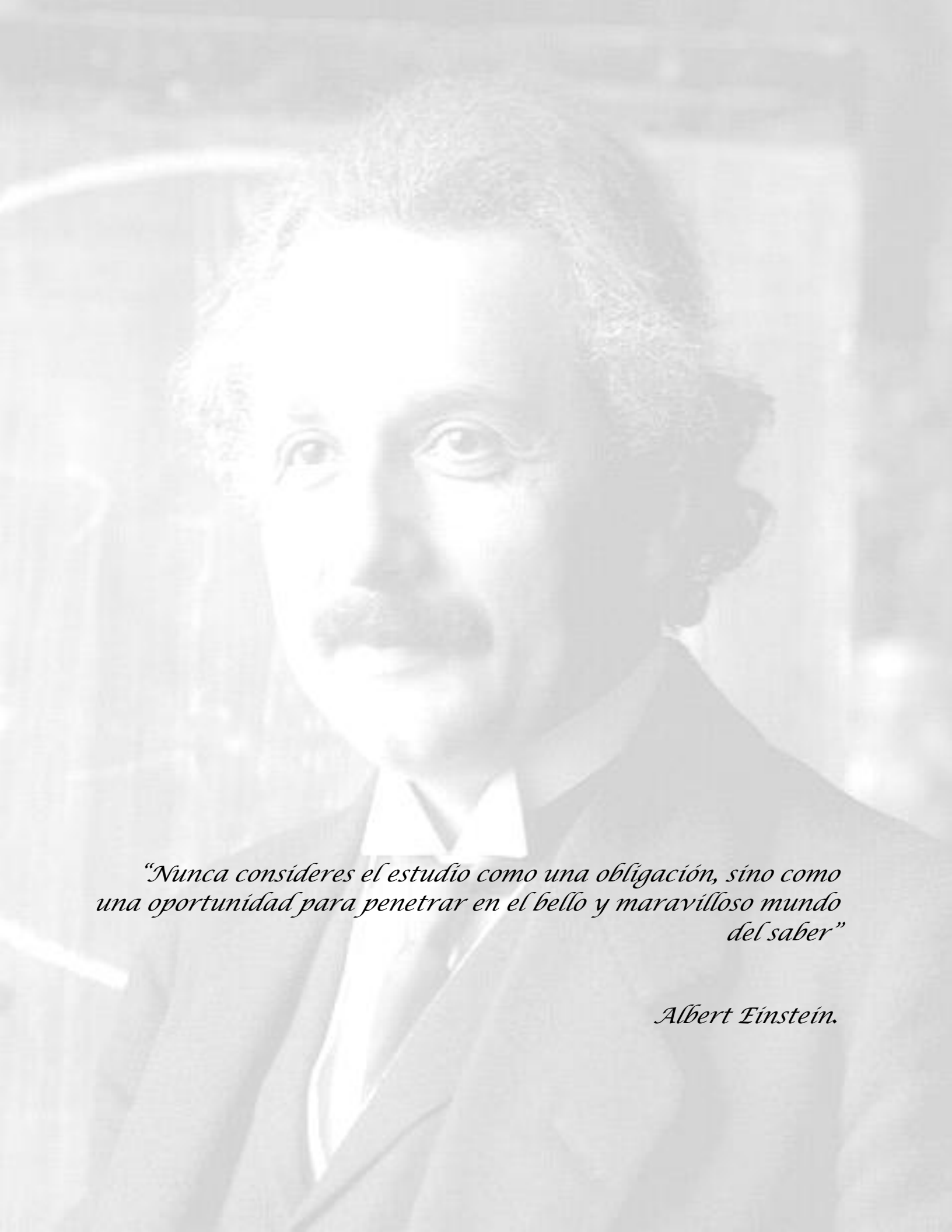
Ingeniero en Ciencias Informáticas

Autores: Yisel de Lisy Sánchez Gallardo
Yadira Rodríguez Tillán

Tutores: Ing. Reynaldo Alvarez Luna
Ing. Yunier Santana Aldana

Ciudad de La Habana, Cuba

Junio, 2010



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber"

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo titulado: Solución informática para la gestión de la información del sistema alasARBOGEN 2.0 y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso de este en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Yisel de Lisy Sánchez Gallardo **Yadira Rodríguez Tillán**

_____ _____

.

AGRADECIMIENTOS

A nuestros padres en primer lugar, por ser las personas más importantes en nuestras vidas, capaces de dar todo por sus niñas.

A nuestras hermanitas, por ser nuestras mejores amigas, compañeras y confidentes.

A nuestros abuelos, por malcriarnos y querernos tanto como a sus propios hijos.

A nuestros novios, por tanto apoyo y amor que nos han dado a través de todos estos años.

A nuestros suegros, por su comprensión, ayuda y cariño incondicional.

A toda nuestra familia, por toda la confianza y el amor depositado en nosotras.

A nuestros tutores, por ser guías y ejemplos de nuestro trabajo.

A todos nuestros amigos que siempre han estado en las buenas y en las malas.

A todos aquellos que, de una forma u otra, han contribuido en nuestra formación como personas y profesionales, muchas gracias.

Yisel y Yadira

DEDICATORIA

De Yisel

A Yadira, por ser mi amiga y compañera incondicional durante toda la carrera.

A mi mamá por ser la persona que más amor ha sabido darme desde el día en que nací. Te adoro mami.

A mi papá por ser mi fuente de inspiración, mi guía, mi consejero, por apoyar mis decisiones y comprenderme como nadie.

A mi hermanita linda por quererme tanto y ser tan especial para mí.

A mi novio por todo el amor, paciencia y dedicación que ha tenido conmigo.

A toda mi familia y mi gente, como diría mi sobrinita, por tanto cariño y comprensión; y a mis verdaderos amigos porque sé, estarán siempre que los necesite, en especial a Annia por ser mi amiga del alma.

De Yadira

A Yisi, por ser mi amiga y compañera incondicional durante toda la carrera.

A mi mamita por darme tanto amor y fuerza desde el día en que nací para seguir adelante. Por ser lo más grande que tengo en la vida.

A mi papito por amarme, apoyarme y guiarme en cada paso que doy. Por ser mi fuente de inspiración.

A mi hermanita, por se mi amiga y compartir conmigo todas las etapas de mi vida. Por ser mi ejemplo a seguir y darme dos sobrinos bien lindos. A mi abuelita por malcriarme y quererme tanto.

A mi novio David, por ayudarme tanto en estos largos años de universidad, por tanta paciencia que ha tenido conmigo. Por ser el gran amor de mi vida.

A toda mi familia y amigos por brindarme tanto amor y cariño, porque sé, estarán siempre que los necesite.

RESUMEN

Con el avance de la informática y la genética médica en nuestro país, surgieron varios proyectos entre la Universidad de las Ciencias Informáticas y el Centro Nacional de Genética Médica. Dada la necesidad de representar los árboles genealógicos de los pacientes con enfermedades genéticas y sus familiares, se desarrolló el sistema alasARBOGEN 1.0, que permite guardar en un fichero el árbol genealógico formado y los datos correspondientes.

Con el fin de proveer al sistema de una fuente centralizada de información que almacene los árboles creados, se requiere desarrollar una segunda versión de alasARBOGEN, que posea una base de datos central, con la cual se puedan comunicar otras aplicaciones. Esto permitirá llevar a cabo investigaciones más profundas y acumular grandes volúmenes de datos, así como garantizar la confidencialidad, disponibilidad e integridad de la información dentro del sistema; es decir, desarrollar de forma segura todo el proceso de gestión de la información.

PALABRAS CLAVE

Base de datos, gestión de la información, alasARBOGEN.

ÍNDICE

INTRODUCCIÓN

El desarrollo convergente de la informática y las telecomunicaciones ha dado lugar a lo que de modo general, se ha denominado las nuevas tecnologías de la información y las comunicaciones (TIC). Cada día, estas desempeñan un papel muy importante en la vida del hombre y están vinculadas a los procesos económicos, políticos y sociales de nuestra época.

Son innumerables los beneficios que brinda el mundo de la informática. La rapidez en la gestión y obtención de resultados, el almacenamiento de grandes volúmenes de datos y la facilidad para encontrar información actualizada, son ejemplo de esto.

Cuba no ha quedado exenta de estos avances, es por ello que en enero de 2000 surge el Ministerio de la Informática y las Comunicaciones (MIC), organismo encargado, entre otras actividades, del desarrollo de la industria del software, el comercio electrónico, la auditoría informática y las telecomunicaciones (GARCÍA, ING. RENIER PÉREZ, 2006).

Durante los últimos años un grupo de instituciones cubanas pertenecientes al MIC junto al Ministerio de Salud Pública (MINSAP), han desarrollado sistemas encaminados a lograr la informatización de los procesos inherentes a la gestión de salud. En todos los casos, el objetivo ha sido proveer al Sistema Nacional de Salud de información confiable, consistente y oportuna para la toma de decisiones y el mejoramiento de los procesos médicos asistenciales, y de esta manera garantizar el incremento en la calidad y seguridad de la atención médica a la población (RAMOS, 2008).

Entre las instituciones del MIC convocadas a contribuir con el proceso de informatización de la sociedad cubana, se encuentra la Universidad de las Ciencias Informáticas (UCI). La UCI en unión a otras entidades, entre las que se encuentra el Centro Nacional de Genética Médica (CNGM), desarrolla disímiles proyectos que requieren sistemas informáticos para la automatización de sus principales actividades. El CNGM realiza un profundo estudio dentro de la población cubana, con el fin de determinar las principales causas de enfermedades genéticas. Entre sus proyecciones figura la excelencia en la atención a los discapacitados, así como cumplir las misiones para preservar el proyecto social cubano.

Con el objetivo de facilitar el trabajo de los genetistas se ha desarrollado en la UCI, en conjunto con el CNGM, varios productos como alasMEDIGEN, alasEPIGEN y alasARBOGEN. Este último es una

aplicación informática para representar árboles genealógicos, primera de su tipo en el país, a través de la cual se pueden representar gráficamente, de una forma organizada y sistemática, los datos genealógicos de un individuo.

La primera versión de alasARBOGEN permite guardar en un fichero el árbol genealógico formado y los datos correspondientes. Esta aplicación se encuentra actualmente desplegada en todos los centros de genética del país, lo que facilita el trabajo investigativo de los genetistas cubanos. Sin embargo, presenta algunas limitaciones, por ejemplo, no cuenta con una fuente centralizada de información donde almacenar los árboles que son creados, ni posee una base de datos con la cual se puedan comunicar otras aplicaciones; esto impide realizar investigaciones más profundas y acumular grandes volúmenes de datos. Tampoco garantiza la seguridad de la información confidencial que se maneja sobre personas con determinadas características significativas desde el punto de vista genético y sobre sus familiares. Debido a los problemas planteados se decidió desarrollar una nueva versión de alasARBOGEN, en la cual se eliminen las ineficiencias existentes.

De ahí la necesidad de investigar cómo garantizar la gestión de la información del sistema alasARBOGEN 2.0, para lo cual se define como objeto de estudio, la gestión de la información en sistemas de representación de árboles genealógicos, y como campo de acción relacionado, la gestión de la información del sistema alasARBOGEN2.0.

Para solucionar el problema científico planteado se ha trazado como objetivo general: desarrollar una solución informática para la gestión de los datos, que garantice la seguridad, disponibilidad e integridad de la información del sistema alasARBOGEN 2.0.

Las **tareas de la investigación** son las siguientes:

- Estudio de las aplicaciones existentes en el mundo para la representación de árboles genealógicos.
- Estudio de los modelos de base de datos existentes.
- Definición de las herramientas, tecnologías y tendencias actuales propuestas para el desarrollo de bases de datos.

- Definición de roles a desempeñar y artefactos correspondientes en el desarrollo de la solución.
- Análisis de la arquitectura definida para desarrollar alasARBOGEN 2.0.
- Definición de los requerimientos relacionados con la gestión de la información del sistema.
- Definición de las funcionalidades de la solución informática a desarrollar.
- Diseño de la base de datos.
- Implementación de la base de datos.
- Implementación de la capa de acceso a datos.
- Implementación de un servicio de persistencia y carga entre la base de datos y las aplicaciones web y de escritorio.

Este trabajo está estructurado en tres capítulos, en los que se describen los métodos y procedimientos a seguir para complementar los objetivos trazados. A continuación una breve descripción de cada uno de ellos:

Capítulo 1: Fundamentación teórica

Expone el estudio del estado del arte en función de los elementos esenciales para la solución del problema científico. Se fundamenta además, la selección de las herramientas y metodologías que se usarán en el desarrollo de la solución.

Capítulo 2: Elaboración de la solución propuesta

Se realiza un análisis crítico del negocio del sistema alasARBOGEN 1.0. Se describe la arquitectura, funcionalidades y requerimientos necesarios para el desarrollo de la solución informática a implementar. Se representa, a través de diagramas, el diseño la base de datos y las clases necesarias para solucionar el problema planteado.

Capítulo 3: Validación de la solución propuesta

Se realiza la validación teórica del diseño y la validación funcional, a través de las cuales se especifican conceptos que se tuvieron en cuenta a la hora de desarrollar la solución, se normaliza y realizan pruebas de volumen y de carga a la base de datos, con el objetivo de lograr el buen funcionamiento de esta. Se propone, a través del estudio de la técnica del balanceo de carga para un servidor de base de datos, la aplicación de esta con el fin de mejorar el rendimiento y el tiempo de respuesta de las consultas, en caso de ser desplegado el sistema en un entorno más complejo, donde se incremente la concurrencia de usuarios a la base de datos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

En el capítulo se ha trazado como objetivo realizar el estudio del estado del arte correspondiente al proceso de gestión de la información. Se incluyen definiciones de términos significativos como: proceso de gestión de la información, base de datos y sistemas gestores de base de datos. Se pretende, también, mencionar y describir las herramientas seleccionadas para desarrollar la solución propuesta, con el principal objetivo de obtener, a través de estas, un óptimo resultado.

1.1 Proceso de gestión de la información

Este proceso abarca diversas operaciones, entre ellas: extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida a través de diferentes fuentes. Gestiona, además, el acceso y los derechos de los usuarios (CURTO, 2006).

En el ámbito específico de la gestión de la información genética, y a la hora de representar y almacenar información acerca de árboles genealógicos, aspectos como la seguridad y confidencialidad de la información procesada, deben ser tenidos en cuenta. Es necesario señalar que en este proceso, las bases de datos juegan un papel fundamental porque son las encargadas de almacenar grandes volúmenes de información; es decir, representan uno de los principales mecanismos a través de los cuales se lleva a cabo el proceso de gestión de la información.

1.2 Aplicaciones existentes para la representación de árboles genealógicos

Existen numerosas aplicaciones a través de las cuales se pueden representar los datos de un individuo y su familia en un árbol genealógico. Entre ellas se encuentran:

- “Family Tree Pilot es un editor de árboles genealógicos que integra una base de datos en la que rellenar los nombres de los familiares, las relaciones de parentesco y plantillas para crear un dibujo del árbol que pueda imprimirse” (LEON).

- Cyrillic: Posee todas las características necesarias para dibujar líneas genéticas. Es un software diseñado para el manejo de datos genéticos en el sistema operativo Windows. Cyrillic permite

asignar enfermedades y fenotipos marcados a individuos, así como incluir descripciones detalladas de estos. Existen dos posibilidades para mostrar los datos, directamente en el dibujo o en forma de tabla en una hoja de cálculo (ING ALFONSO CLARO ARCEO).

- alasARBOGEN 1.0: Esta aplicación es usada para representar en un árbol genealógico, los datos de una persona y su familia. Estos datos están dados por la información significativa, desde el punto de vista genético, de un paciente y sus familiares, e incluye enfermedades y características. La persistencia de los datos se desarrolla a través de ficheros binarios de manera local.

Los sistemas antes mencionados no cuentan con una base de datos centralizada que permita realizar un profundo proceso de gestión de la información. A pesar de ser el Cyrillic el software más desarrollado de su tipo, solo cuenta con una base de datos local en la cual almacenar los datos procesados.

1.3 Mecanismos para realizar el proceso de gestión de la información

Incluye técnicas que se encuentran relacionadas al proceso de gestión de la información, así como las tecnologías existentes para el desempeño de este proceso.

1.3.1 Base de Datos

Existen varios criterios o conceptos de base de datos (BD), definidos desde diferentes puntos de vista, entre los que se encuentran:

- “Una base de datos es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización. En una base de datos, además de los datos, también se almacena su descripción” (ANDRÉS, 2001).

- “Colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones” (MORATALLA, 2001).

- “Una base de datos (cuya abreviatura es BD) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Diferentes programas y diferentes usuarios deben poder utilizar estos datos” (*Kioskea.net*, 2008).

En resumen, se define una BD como una serie de datos organizados de forma estructurada y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una organización determinada.

Además, las bases de datos deben caracterizarse por poseer independencia lógica y física de los datos, igual que mínima redundancia de estos. También debe garantizar acceso seguro y concurrente por parte de múltiples usuarios y permitirles a la vez, realizar consultas complejas de manera optimizada. Un sistema de base de datos debe responder por la integridad y seguridad de la información.

Se definen, básicamente, como modelos de base de datos:

- Bases de datos jerárquicas

“Son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz y a los nodos que no tienen hijos se los conoce como hojas. (Ver figura 1).

”Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos que permiten crear estructuras estables y de gran rendimiento.

”Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos” (MINAY).

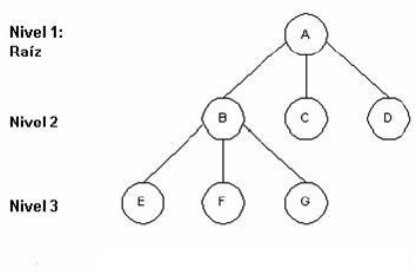


Fig. 1 Estructura jerárquica o en árbol.

- Base de datos de red

"Es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). (Ver figura 2).

"Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía 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" (MINAY).

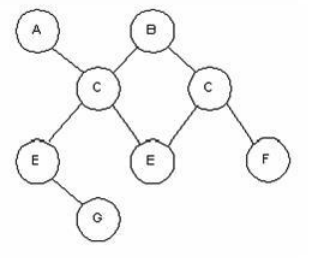


Fig. 2 Estructura de datos de red.

- Bases de datos relacionales

"Es utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que esta es la teoría de las bases de datos relacionales, la

mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla). (Ver figura 3).

”El lenguaje más habitual para construir las consultas a bases de datos relacionales es el *Lenguaje Estructurado de Consultas* (SQL por sus siglas en inglés, *Structured Query Language*), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

”Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos” (MINAY).

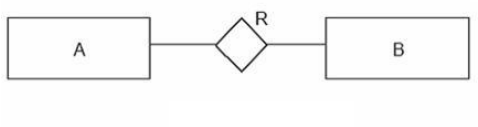


Fig. 3 Bases de datos relacionales.

Debido a lo antes expuesto, el modelo de datos a utilizar en el desarrollo de la solución es el modelo de datos relacional, porque es el que más se ajusta al problema planteado. Esto se debe a que la información procede de problemas de la vida real y se necesita administrar de forma dinámica. Siguiendo este modelo se evita la ineficiencia de la redundancia de datos que caracteriza a las bases de datos jerárquicas, y es a la vez más sencillo de administrar que las bases de datos de red.

1.3.2 Sistema Gestor de Base de Datos (SGBD)

El concepto de Sistema Gestor de Base de Datos (SGBD) ha sido definido de múltiples maneras:

- “Los sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta” (ARCEL LABRADA BATISTA, 2007).

- “El sistema de gestión de la base de datos es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma” (ANDRÉS, 2001).

En resumen, un SGBD permite almacenar y a la vez acceder a los datos, de forma rápida y estructurada. Estos sistemas admiten definir usuarios y restricciones de acceso para cada uno de ellos, también logran manipular los datos siguiendo las órdenes de los usuarios. Los SGBD son capaces de controlar la concurrencia y las operaciones asociadas a la recuperación de los fallos y cuidan que se respete la seguridad e integridad de la información.

1.3.3 Tecnologías de comunicación entre aplicaciones

Algunas de las tecnologías de comunicación entre aplicaciones informáticas son:

- CORBA

Son las siglas en inglés de Common Object Request Broker Arquitecture y designan a un conjunto de estándares que forman un marco de referencia para establecer la interacción de los componentes. Las especificaciones se escriben en un lenguaje de definición neutral, *Interface Definition Language* (IDL), que define la frontera de un componente. CORBA fue diseñado para resolver el problema de intercomunicar máquinas, es una arquitectura para manejar componentes remotos o distribuidos. Se ha instrumentado en un rango amplio de plataformas y vendedores (mainframe, Unix, Windows). (RODRÍGUEZ)

- RMI

“Remote Method Invocation es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales de Java, así comparten recursos y carga de procesamiento a través de varios sistemas” (MONTENEGRO, 2003).

Las aplicaciones RMI comprenden dos programas separados: un servidor y un cliente. Una aplicación servidor crea varios objetos remotos, hace accesibles unas referencias a esos objetos, y espera a que los clientes realicen llamadas a estos. Una aplicación cliente obtiene una referencia remota de uno o más objetos en el servidor y llama a sus métodos. RMI proporciona el mecanismo por el que se comunican e intercambian información del cliente al servidor (TORRIJOS, 2001).

- Servicios HTTP Invoker

Se encuentra entre los servicios remotos que son soportados por Spring Framework. Emplea el modelo remoto para hacer llamadas sobre HTTP y al mismo tiempo, el paso de objetos de Java usando técnicas de serialización del lenguaje. Esto permite que la implementación de un servicio remoto desde una clase de Java sea más fácil y que el desarrollador se concentre en la interfaz de negocio del servicio, por encima de los detalles de su infraestructura. Utiliza HTTP como protocolo de transporte, sin embargo, se basa en la infraestructura de invocación de RMI y supera a este en cuanto a la capacidad de atravesar firewalls (ÁLVAREZ, 2008).

- Componente para la comunicación entre aplicaciones

Este componente de software para la transferencia de datos en arquitecturas cliente/servidor, tiene como objetivo principal ofrecer los servicios que brindan las tecnologías HTTP Invoker y RMI. Permite la transferencia de datos entre aplicaciones de forma segura a través de la red, sin tener que conocer las particularidades de las tecnologías de comunicación utilizadas en el envío, mediante el uso de certificados digitales y la selección del mecanismo a utilizar, según las características del entorno.

Se seleccionó este componente debido a que se ajusta a las necesidades, en el ámbito de la comunicación entre aplicaciones, y además fue realizado en la UCI.

1.4 Herramientas a utilizar en el proceso de desarrollo de la solución

1.4.1 Herramienta CASE a utilizar para el modelado de base de datos

Entre las herramientas CASE más utilizadas para el diseño de base de datos se encuentran:

Embarcadero ER/Studio

“Es una herramienta de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos, grandes y complejas.

”Funcionalidades:

- Capacidad fuerte en el diseño lógico.

- Sincronización bidireccional de los diseños lógico y físico.
- Construcción automática de Base de Datos.
- Reingeniería inversa de Base de Datos.
- Documentación basada en HTML.
- Un Repositorio para el modelado" (2007, Chango).

Visual Paradigm

Es una herramienta CASE que utiliza UML como lenguaje de modelado. Visual Paradigm, es además, una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software contribuye a una rápida construcción de aplicaciones con calidad y a un menor costo. Permite representar disímiles diagramas de clases, y también generar código inverso, código desde diagramas y desde documentación.

La herramienta Visual Paradigm ofrece sencillez y claridad a la hora de su manejo. Esto se debe a que posee un entorno fácil de manipular y muy flexible, hasta el punto de admitir tipos de datos definidos por el usuario. Permite realizar dos tipos de diagramas fundamentales: el diagrama entidad-relación, que modela la base de datos relacional en el nivel físico, y el diagrama de mapeo relacional de objetos, que muestra el mapeo entre clases y entidades. Tiene la ventaja de ser multiplataforma (Free Dowload Manager, 2007). No es de libre distribución, pero la UCI adquirió la licencia de Visual Paradigm para UML.

Como principales características se encuentran:

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Diagramas de flujo de datos. Soporte ORM.
- Generación de objetos Java desde la base de datos. Generación de bases de datos.

- Disponibilidad de integrarse en los principales IDE.
- Disponibilidad en múltiples plataformas. Está disponible para Win98, WinME, WinNT 4.x, Windows2000, WinXP, Windows2003, Mac OS, GNU/Linux (SIERRA).

Se seleccionó Visual Paradigm para el modelado de la base de datos, a pesar de ser Embarcadero ER/Studio una eficaz herramienta, debido a que soporta el ciclo de vida completo del sistema alasARBOGEN 2.0; a través de este se modelará todo el sistema en su totalidad y no sólo la base de datos.

También brinda una serie de ventajas, entre las que se destaca, la existencia de un Smart Development Environment (SDE) para Eclipse; es decir, la herramienta CASE (Visual Paradigm), de modelado UML, se integra completamente a este entorno de desarrollo (Eclipse). Esto permite representar los diferentes tipos de diagramas UML en Eclipse, realizar ingeniería inversa desde código Java a diagramas de clases y generar código Java y documentación.

1.4.2 Herramienta a utilizar para la gestión de la base de datos

Sistema Gestor de Bases de Datos MySQL

Es un Sistema Gestor de Bases de Datos Relacional muy utilizado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación se debe, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de muchos lenguajes de programación. Soporta gran cantidad de tipos de datos para las columnas; además, tiene gran portabilidad entre sistemas y gestiona usuarios y contraseñas, manteniendo un buen nivel de seguridad en los datos (*Soporte Linux* 2008).

Se encuentran entre sus principales características:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP).
- Integridad referencial: admite la declaración de llaves foráneas en la creación de tablas, aunque internamente no las trate de forma diferente al resto de campos (CORONADO, 2004).

Sistema Gestor de Bases de Datos PostgreSQL

Es un Sistema Gestor de Bases de Datos Objeto-Relacionales. Está considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Esta herramienta cliente-servidor para la gestión de bases de datos se caracteriza por ser un sistema estable, de alto rendimiento y gran flexibilidad, pues trabaja en la mayoría de los sistemas Unix.

PostgreSQL puede ser integrado al ambiente Windows, y de esta manera, les permite a los desarrolladores generar nuevas aplicaciones o mantener las ya existentes, así como desarrollar o migrar aplicaciones desde Access, Visual Basic, Foxpro, Visual Foxpro, C/C++ Visual C/C++, Delphi. Está caracterizado por ser altamente adaptable a las necesidades del cliente y posee documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios (GONZÁLEZ, 2010).

Entre sus principales características se encuentran:

- Implementación del estándar SQL92/SQL99.
- Incorpora una estructura de datos array.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows.
- Soporte de todas las características de una base de datos profesional (triggers, store procedures funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas).

- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales y minería de datos.
- Utilidades para análisis y optimización de consultas.
- Almacenaje especial para tipos de datos grandes.

Se seleccionó PostgreSQL por encontrarse entre los SGBD más completos. Se destaca su soporte de transacciones y su estabilidad, además es multiplataforma.

Es preciso señalar que PostgreSQL posee una gran escalabilidad. Es capaz de ajustarse, de forma óptima, al número de CPU y a la cantidad de memoria que posee el sistema, por lo que soporta una mayor cantidad de peticiones simultáneas de manera correcta, no sucede así con MySQL.

1.5 Fundamentación de las tecnologías y herramientas a utilizar

1.5.1 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Indican todas las actividades a realizar para lograr el producto informático deseado, establecen qué personas deben participar en el desarrollo de las actividades y qué papel deben desarrollar. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (COMMONS, 2006).

Rational Unified Process (RUP)

Es un proceso de desarrollo de software que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Entre sus características esenciales están que es dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental.

El ciclo de vida de RUP está dividido en cuatro fases: inicio, elaboración, construcción y transición, que corresponden a los cuatro hitos principales de RUP: proyecto, arquitectura, versión β y *release.*

Fases de RUP

- Inicio: Consiste en alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto y generar el ámbito del proyecto, el caso de negocio, síntesis de arquitectura posible y el alcance del proyecto.

- Elaboración: Se realiza el establecimiento de la línea base para la arquitectura del sistema y proporcionar una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, en la que se mitigan la mayoría de los riesgos tecnológicos.

- Construcción: Consiste en completar el desarrollo del sistema basado en la línea base de la arquitectura.

- Transición: Es donde se garantiza que el software está listo para entregarlo a los usuarios.

Además de las principales características, RUP define nueve flujos de trabajo. De estos seis son de ingeniería y los tres restantes sirven de apoyo a los de ingeniería. Los flujos son Modelación de Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Instalación, Administración del proyecto, Administración de configuración y cambios y Ambiente (*Itera*, 2010).

1.5.2 Lenguaje Unificado de Modelado (UML)

Unified Modeling Language (UML, por sus siglas en inglés) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el *Object Management Group* (OMG). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), e incluye aspectos conceptuales (procesos de negocios y funciones del sistema) y aspectos concretos (expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables).

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar (SERRANO, 2009).

1.5.3 Framework de desarrollo

Un framework es una estructura definida por artefactos o módulos de software, mediante el cual son organizados y desarrollados proyectos de software. Incluye soporte de programas, bibliotecas y un lenguaje interpretado con el objetivo de desarrollar y unir los diferentes componentes dentro de un proyecto.

Propone una arquitectura de software que modela las relaciones de las entidades del dominio y proporciona una estructura y una metodología de trabajo que utiliza las aplicaciones del dominio (Framework, 2010).

Existe un framework de mapeo objeto-relacional conocido por su nombre en inglés *Object-Relational mapping* (ORM). Una herramienta ORM define la forma en que se establece la correspondencia entre las clases y las tablas. Entre las herramientas ORM más conocidas se encuentra Hibernate.

Hibernate

Es un framework de consultas y mapeos objeto-relacional muy potente y de alto rendimiento. Tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y obtener información de estas. Facilita el mapeo de atributos entre una BD relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML).

Hibernate proporciona además, un lenguaje para el manejo de consultas a la base de datos similar a SQL y es utilizado para obtener objetos de la base de datos. Es un modelo de programación natural y una capa de persistencia objeto/relacional que permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, así como un gran número de tipos de datos.

Las principales características de Hibernate:

- Facilidades en metadatos: soporta el formato del mapeado de XML, diseñado para ser editado a mano y el mapeado basado en anotaciones. Además de validación basada en anotaciones.

- Puede configurarse y ejecutarse en la mayoría de aplicaciones Java y entornos de desarrollo. Generalmente, se utiliza en aplicaciones cliente-servidor de dos y tres capas, y se despliega únicamente en el servidor.

- Transacciones, caché, asociaciones, polimorfismo, herencia*, lazy loading*, persistencia transitiva, estrategias de *fetching*.

- Gran escalabilidad: es muy eficiente, tiene una arquitectura de caché de doble capa y podría ser usado en un clúster.

- Software libre: está bajo licencia *Lesser GNU Public License* (LGPL).

- Su característica más importante es que implementa varios dialectos.

Los dialectos en Hibernate son los encargados de encapsular todas las diferentes formas en que el framework puede comunicarse con una base de datos, para lograr tareas como obtener un valor de secuencia o estructurar una petición SELECT.

Hibernate reúne un gran rango de dialectos para las bases de datos más populares (PostgreSQL, Microsoft, SQL Server, MySQL, Oracle). Además de manera sencilla se puede escribir uno propio (ANDALUCIA).

1.5.4 Herramientas de administración de bases de datos

En la actualidad, las bases de datos creadas bajo lenguaje SQL o afines se han convertido en parte de toda creación de utilidades y/o sitios de internet. También existen numerosas herramientas que se relacionan directamente con servidores de bases de datos.

pgAdminIII

Es la herramienta *Open Source* de administración por excelencia para bases de datos *PostgreSQL*. Es empaquetado con el instalador de Windows, y puede ser usado como un cliente para administrar un servidor remoto en otro sistema operativo.

Esta herramienta está diseñada para darles respuestas a las necesidades de la mayoría de los usuarios, desde la escritura de consultas simples en SQL hasta el desarrollo de bases de datos complejas. La interfaz gráfica soporta todas las características presentes en PostgreSQL y se puede realizar fácilmente su administración. Está disponible en más de 30 lenguajes y para varios sistemas operativos.

Algunas de sus características son:

- Soporte completo para UNICODE.
- Edición rápida de consultas y datos multihilo.
- Soporte para todos los tipos de objetos de PostgreSQL.
- Incluye una interfaz gráfica de administración.
- Incluye un editor de código de procedimientos y funciones (*GUI de PostgreSQL*, 2009).

1.5.5 Lenguajes de Programación

Es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto es un conjunto de normas y reglas a seguir por las personas, con el objetivo de dar instrucciones a un equipo (VASQUEZ, 1998).

Java

Es un lenguaje de programación orientado a objetos desarrollado por *Sun Microsystems*. Tiene mucho en su sintaxis de C y C++, pero posee un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir muchos errores, como la manipulación directa de punteros a memoria.

Java es un potente y versátil lenguaje de programación que puede trabajar en todo tipo de entornos, desde servidores de aplicaciones *middle-tier* hasta clientes web. Independientemente del tipo de aplicación que se desarrolle y del tipo de máquina en la que se ejecute el código, la aplicación seguramente tendrá que acceder a datos almacenados en algún tipo de base de datos. Las bases de datos relacionales son la elección obvia a la hora de trabajar con Java debido a sus características.

Una de las principales características que favoreció el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que un mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además, es un lenguaje orientado a objetos que resuelve los problemas en la complejidad de los sistemas (ALVAREZ, 2001).

La plataforma Java consta de las siguientes partes:

- La máquina virtual de Java o JRE (Java Runtime Environment) que permite la portabilidad en ejecución.
- El API Java o JDK (Java Development Kit), una biblioteca estándar para el lenguaje.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi todo lo que se puede hacer en otro lenguaje también se puede hacer en Java y muchas veces con grandes ventajas (*WebTaller*, 2008).

1.5.6 Entorno de Desarrollo

Un *Integrated Development Environment* (IDE) 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 GUI. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (JELSOFT, 2010).

Eclipse

"Es un IDE o una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en *plugins.* Además da soporte a todo tipo de proyectos que abarcan todo el ciclo de vida del desarrollo de aplicaciones, se incluye soporte para modelado" (LAURA BERMEJO SANZ).

Eclipse es un IDE mucho más flexible de lo imaginado, pues gracias a una infinidad de plugins permite, editar clases visuales de Java y en varios lenguajes más, conectarse a bases de datos y escribir consultas SQL, así como conservar el registro de las versiones y generar y mantener la documentación de cada etapa del proyecto. Otra ventaja de los plugins es la existencia de herramientas como el SDE para

Eclipse, que está completamente integrada a este entorno de desarrollo, con el objetivo de lograr un eficiente modelado UML (LAURA BERMEJO SANZ).

Las características más importantes de Eclipse son las siguientes:

- "Editor visual con sintaxis coloreada.
- Modifica e inspecciona valores de variables.
- Avisa de los errores cometidos mediante una ventana secundaria.
- Depura código que resida en una máquina remota.
- ECLIPSE es soportado por los principales sistemas operativos Linux, Windows, Solaris 8, Mac OSX-Mac/Carbon.
- En Eclipse existen Control de versiones como CVS y Subclipse" (LAURA BERMEJO SANZ).

1.6 Roles y artefactos

Es una definición abstracta de un conjunto de responsabilidades por actividades a realizar y artefactos a producir. Un rol específico puede ser desempeñado por un individuo o un conjunto de individuos, trabajando juntos como un equipo. Un mismo miembro del equipo dentro de un proyecto puede desempeñar diferentes roles.

Los roles están constituidos por un conjunto coherente de actividades a realizar. Estas actividades están muy relacionadas y funcionalmente acopladas.

Las actividades realizadas por los diferentes roles están muy relacionadas a los artefactos. Los artefactos son los productos iniciales, finales o intermedios de las actividades desarrolladas en un proyecto.

En el presente trabajo de diploma de acuerdo con las necesidades actuales del proyecto se desarrollarán los roles de: diseñador de base de datos e implementador, perteneciente al grupo de Desarrolladores definido por RUP.

1.6.1 Diseñador de base de datos

Este rol dirige el diseño de la estructura de almacenamiento de datos persistentes que se utilizará en el sistema. El diseñador de base de datos es responsable de definir el diseño detallado de la base de datos, e incluye tablas, índices, vistas, restricciones, desencadenantes, procedimientos almacenados y otras construcciones específicas necesarias de la base de datos para almacenar, recuperar y eliminar objetos persistentes.

Los artefactos realizados por el diseñador de base de datos que se obtendrán en el presente trabajo son:

- Especificación de migración de datos: Este artefacto contiene el perfil de datos de las fuentes de datos que se van a migrar y la asignación entre las fuentes de datos y la base de datos de destino. Su objetivo principal es describir las fuentes de datos a ser migrados y especificar la asignación entre los datos de origen y los datos de destino. Será utilizado por el diseñador para diseñar los componentes de software automatizado de migración de datos.

- Modelo de datos: Este artefacto describe las representaciones lógicas y físicas de datos persistentes utilizados por la aplicación. En los casos en que la aplicación utilizara un sistema de gestión de bases de datos relacionales (RDBMS), el modelo de datos también incluye elementos de modelo para procedimientos almacenados, desencadenantes, restricciones y otros, que definen la interacción de los componentes de la aplicación con RDBMS.

1.6.2 Implementador

Este rol desarrolla los componentes de software y efectúa las pruebas de desarrollador para la integración en subsistemas más grandes, de acuerdo con los estándares adoptados de proyecto.

Los artefactos realizados por el implementador que se obtendrán en el presente trabajo son:

- Elemento de implementación: Estos artefactos son los componentes físicos que forman una implementación que cuenta con archivos y directorios. Incluyen los archivos de código de software (origen y binario), los archivos de datos y los archivos de documentación, así como los archivos de ayuda en línea.

- Subsistema de implementación: Este artefacto consta de un conjunto de elementos de implementación. Estructura el modelo de implementación dividiéndolo en componentes más pequeños que se pueden integrar y probar separadamente.

- Prueba de desarrollador: Este artefacto abarca el trabajo tradicionalmente pensado bajo las categorías siguientes: pruebas de unidad, parte de las pruebas de integración, y algunos aspectos de lo que se denomina pruebas del sistema.

- Prueba de fragmento para simulación: Este artefacto es un elemento de implementación especializado que se utiliza para efectuar pruebas, que simula un componente verdadero (*Ayuda de RUP*).

1.7 Patrón DAO

Data Access Object Service Activator (DAO) se encuentra ubicado en la capa de integración de los patrones *Java 2 Enterprise Edition* (J2EE). Utiliza un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos, así logra separar la lógica de negocios de la lógica de acceso a datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos. Estos pueden estar almacenados de modos distintos (EJBs, JDO, LDAP).

Separar la lógica de negocios con el acceso a datos permite crear implementaciones plegables del DAO, con solo seleccionar el tipo de fuente de datos durante la instalación-configuración de una aplicación.

La figura 4 muestra el diagrama de clases que representa las relaciones para el patrón DAO:

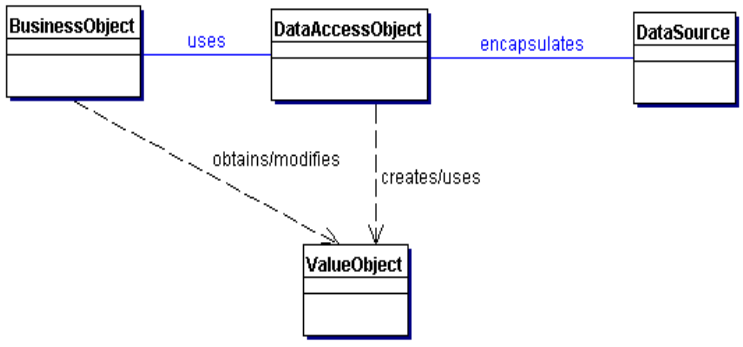


Fig. 4 Diagrama de clases para el patrón DAO (Ciberaula, 2010).

CONCLUSIONES

A través de este capítulo se ha realizado todo el análisis referente al estudio del arte. Se expusieron y argumentaron los conceptos fundamentales relacionados al proceso de gestión de la información, así como las descripciones de varias herramientas que pueden ser utilizadas en el transcurso de ese proceso. Se fundamentó la selección de las herramientas que serán usadas con el fin de lograr un óptimo desarrollo del sistema. Se definieron y especificaron además, los roles a desempeñar y los artefactos que resultan de las tareas desarrolladas en cada rol definido.

CAPÍTULO 2: ELABORACIÓN DE LA SOLUCIÓN PROPUESTA

INTRODUCCIÓN

En el presente capítulo se realizará un análisis crítico del negocio de la primera versión del sistema alasARBOGEN. Se explicarán los procesos que se desean automatizar y el dominio existente dado por el conjunto de entidades que en este intervienen. Se describirá la arquitectura del nuevo sistema, así como los requisitos funcionales y no funcionales a tener en cuenta para el desarrollo de la solución propuesta.

Se realizarán los diagramas y las tablas pertinentes para lograr diseñar la solución y se describirán las clases necesarias para cumplimentar los objetivos trazados.

2.1 Flujo de actividades de los procesos

La primera versión del sistema alasARBOGEN presenta varias dificultades debido a las ineficiencias de sus funcionalidades. No cuenta con una aplicación web a través de la cual se pueda interactuar con otros servicios que debe brindar este tipo de sistema, como por ejemplo, los que brinda la Red Nacional de Salud (INFOMED).

Este sistema no permite realizar una amplia gestión de las muestras de laboratorio. La estructura y organización gráfica del trabajo no es eficiente, debido a que el árbol genealógico no queda organizado por generación y fuerza al genetista a perder tiempo a la hora de analizar la información. Una gran desventaja que presenta es que está desarrollada para software propietario.

Tampoco cuenta con una fuente centralizada de información donde almacenar los árboles que son creados y los datos de las familias representadas en estos. No garantiza la seguridad y confidencialidad de la información que se maneja, sobre personas con determinadas características significativas desde el punto de vista genético, y sobre sus familiares.

2.2 Objeto de automatización

En la nueva versión de alasARBOGEN se automatizará todo el proceso de representación de la información acerca de un paciente. También se automatizarán todos los procesos de intercambio, almacenamiento, consulta y modificación de los datos, de una forma mucho más eficiente y cómoda.

Se realizarán a través de una aplicación de escritorio los árboles genealógicos correspondientes a cada familia que se desee estudiar, y se ejecutarán a través de una aplicación web las consultas necesarias sobre los datos almacenados, sin necesidad de disponer de la aplicación de escritorio.

La solución propuesta resuelve el problema de la gestión de la información en el nuevo sistema a realizar. Con la implementación de una base de datos y de las clases necesarias para la conexión de estos datos se puede facilitar el proceso de atención a los pacientes en los diferentes centros médicos. Se logrará almacenar y explotar grandes volúmenes de información, y por tanto, realizar las investigaciones pertinentes.

2.3 Modelo de Dominio

En la figura 5 se representan las principales entidades definidas dentro del dominio. Se representan, además, todas las relaciones y la forma en que están estructuradas todas las entidades, así como el flujo de actividades existentes entre ellas.

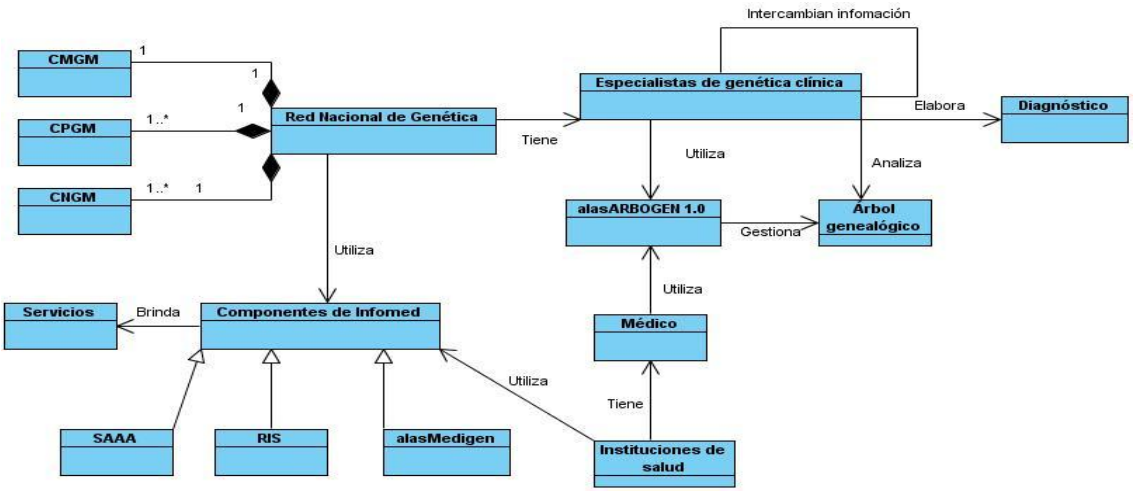


Fig. 5 Modelo de Dominio.

2.4 Descripción y fundamentación de la arquitectura

La figura 6 muestra la composición del sistema: aplicación escritorio, aplicación web y base de datos. A través de la capa de acceso a datos se realizará todo el proceso que esté relacionado con almacenar, consultar y/o modificar la información dentro de la base de datos.

La aplicación web debe posibilitar la descarga y/o instalación de la aplicación de escritorio. Estará encargada de analizar la información disponible en la base de datos y los patrones entre los árboles; así como de emitir los reportes bajo diferentes criterios y visualizar los árboles desde la BD para propiciar el intercambio entre los especialistas.

Mediante la aplicación de escritorio se genera la información genealógica concerniente a la familia que se desea estudiar. La información resultante es procesada por el componente de comunicación entre aplicaciones, que después de ser validada es transferida al servidor de bases de datos, de forma segura por HTTP INVOKER.

El componente de comunicación está formado por dos librerías independientes, una para el lado del cliente y otra para el lado del servidor, en esta última se almacena toda la lógica de validación del XML a través del esquema.

La capa de acceso a datos contiene la lógica de mapeo de la información de las bases de datos con los objetos de las clases que se deciden hacer persistentes. Esto se logra con la utilización del framework Hibernate. Una vez obtenidos los ficheros, el siguiente paso es la implementación de la capa de acceso a los elementos del modelo de negocio, mediante el uso del patrón DAO.

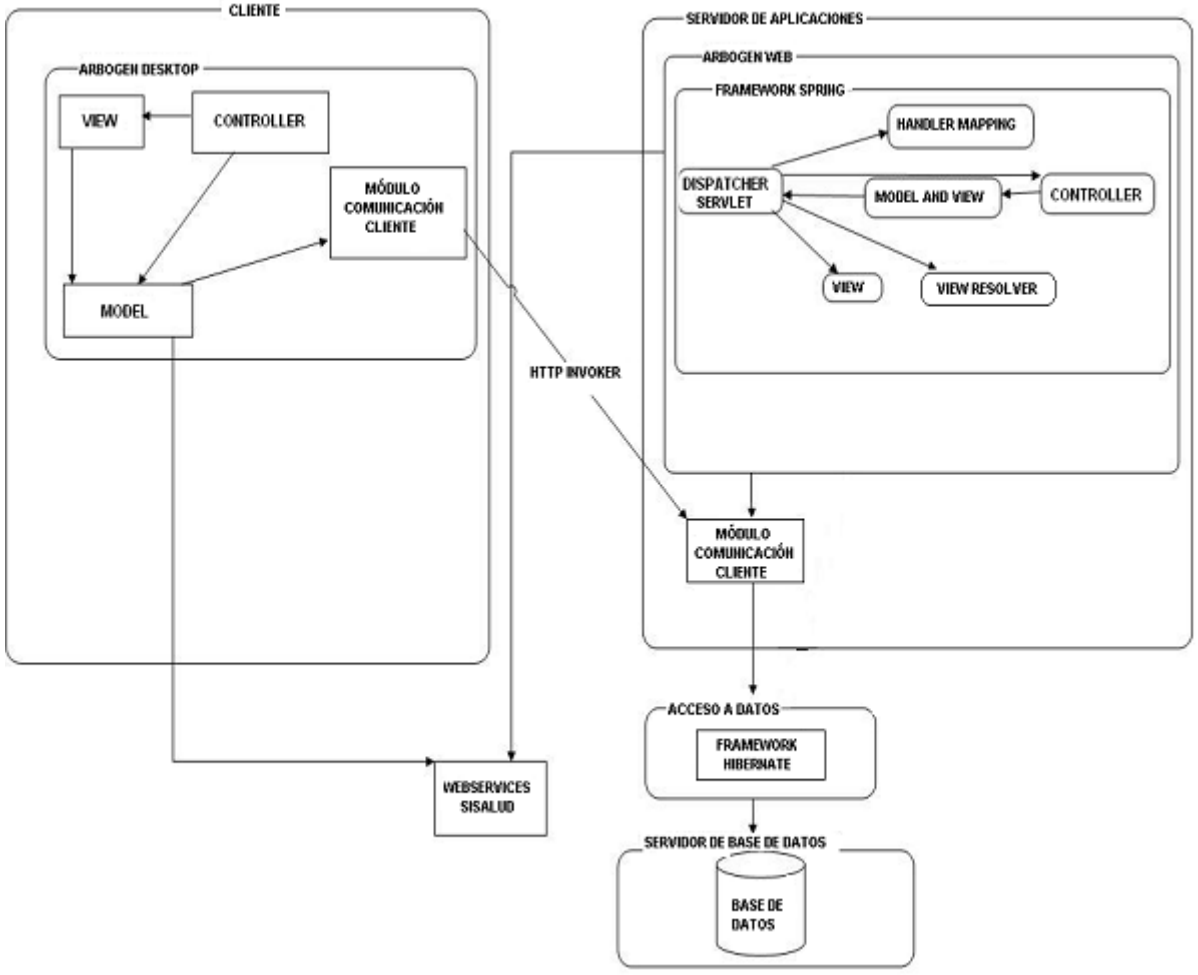


Fig. 6 Descripción de la arquitectura.

2.5 Requisitos del sistema

2.5.1 Requisitos funcionales

Para que el sistema cumpla con las funcionalidades necesarias se definen numerosos requisitos funcionales. En la propuesta de la presente solución, debido a que se gestiona la información almacenada en las tablas de la base de datos, influyen directamente los siguientes requisitos funcionales, que representan un 61,9 % del total de requisitos para las aplicaciones web y escritorio:

Requisitos funcionales para la aplicación web:

RF1. Autenticar Usuario.

RF2. Gestionar Usuario.

- RF2.1- Registrar usuario.
- RF2.2- Modificar usuario.
- RF2.3- Listar usuarios.
- RF2.4- Eliminar usuario.

RF4. Cargar árbol genealógico desde fichero.

RF5. Generar Reporte.

RF6. Gestionar Nomencladores.

RF7. Compartir árbol genealógico.

RF8. Cambiar Contraseña.

Requisitos funcionales para la aplicación escritorio:

RF2. Gestionar Relaciones entre Individuos.

- RF 2.1 Crear relación entre individuos.
- RF 2.2 Eliminar relación entre individuos.

RF3. Gestionar Símbolos.

- RF 3.1 Crear nuevo símbolo.
- RF 3.2 Modificar símbolos.
- RF 3.3 Eliminar un símbolo.

RF4. Gestionar Enfermedades.

- RF 4.1 Crear una nueva enfermedad.
- RF 4.2 Eliminar una enfermedad.

RF5. Gestionar Muestras.

- RF 5.1 Insertar nuevas muestras.
- RF 5.2 Modificar una muestra.
- RF 5.3 Eliminar una muestra.

RF6. Gestionar Familia.

- RF 6.1 Crear una nueva familia.
- RF 6.2 Buscar una familia.

RF 6.3 Actualizar datos de una familia.

RF 6.4 Guardar familia.

RF 6.5 Exportar familia.

RF12. Generar Reporte del Árbol Genealógico.

2.5.2 Requisitos no funcionales

El sistema debe permitir un acceso rápido y seguro desde cualquier centro de genética del país. Teniendo en cuenta que los servicios de conexión a la red nacional de salud podrían verse interrumpidos, la aplicación de escritorio permitirá guardar los datos de forma local y estos podrán enviarse luego de restablecida la conexión.

Para cumplir lo antes planteado se definen los requisitos no funcionales que el sistema debe cumplir.

Rendimiento:

La velocidad de procesamiento del sistema debe ser rápida, al igual que la capacidad de respuesta.

Soporte:

Las necesidades de los usuarios deben quedar satisfechas después de la puesta en explotación del sistema, a través de actualizaciones y mejoras. Para lograr esto se elaboraran una serie de manuales y videos tutoriales, así como la atención en línea.

Software:

Se requiere para el funcionamiento del sistema disponer de un servidor con el sistema operativo Linux, Apache 2.0, Java Web Start, la Máquina Virtual de Java 5.0 y PostgreSQL 8.3 o versiones superiores. Los usuarios del sistema deberán contar con un navegador Internet Explorer 5.5, Mozilla Firefox 2.0, o el Opera 8.5, todos ellos en la versión mencionada o una superior.

Hardware:

Para el desarrollo y ejecución del sistema se precisó que los requisitos estuvieran en función de los requerimientos que propone J2EE para el desarrollo de aplicaciones en Java.

Para el servidor de aplicaciones:

- Microprocesador Pentium III a 2.8 GHz o superior.
- 1GB de RAM o superior.
- 300 MB de espacio en disco duro para el despliegue de la aplicación web y la salva de la aplicación de escritorio.

Para el cliente:

- Conexión al servidor a través de un modem o tarjeta de red.
- Procesador Pentium III o superior.
- 256 MB RAM como mínimo (Recomendado 512 MB RAM o superior).
- 500 MB de espacio libre en disco (Recomendado 800 MB de espacio libre en disco).

Se hace necesario contar con una impresora para la impresión de los reportes.

Disponibilidad:

El sistema debe estar disponible a tiempo completo, y recuperarse rápidamente ante cualquier tipo de fallo. Deben ser creadas copias de respaldo de forma periódica, de manera tal que el sistema pueda restaurarse en caso de fallo crítico o pérdida de información. El sistema debe ser capaz de funcionar, independientemente de que los servicios de los diferentes componentes de SISalud no estén disponibles.

Requisitos legales:

Las tecnologías y herramientas que se utilicen para desarrollar el sistema deben estar bajo la licencia de software libre.

Persistencia:

La información generada por el sistema debe ser almacenada en bases de datos permanentemente con el objetivo de poder generar reportes y realizar estudios posteriores. Teniendo en cuenta además, que en la aplicación de escritorio la información persistirá en ficheros, en el caso de que la base de datos no se encuentre disponible en un momento determinado.

Portabilidad:

El sistema debe permitir ejecutarse en cualquiera de los sistemas operativos más usados en la actualidad (Windows o Linux cualquiera de sus versiones).

2.6 Diagrama de clases del diseño

A continuación se exponen ejemplos de los diagramas de clases del diseño, tanto de la aplicación escritorio como web (ver figuras 7 y 8), debido a que fue necesario realizar un profundo análisis de estos. Como resultado del análisis se pretende definir cuál es la información imprescindible para el buen funcionamiento del sistema a desarrollar. Luego se procederá a realizar el diagrama de clases persistentes con la información seleccionada, mediante el cual quedarán representados los datos que se deben almacenar en la base de datos.
Cada entidad representada, en los siguientes ejemplos, definen una clase dentro del diagrama de clases persistentes a realizar, como de igual forma ocurre en el resto de los diagramas de clases del diseño. (Ver Anexos 1 y 2).

2.6.1 Diagrama de clases de la aplicación escritorio

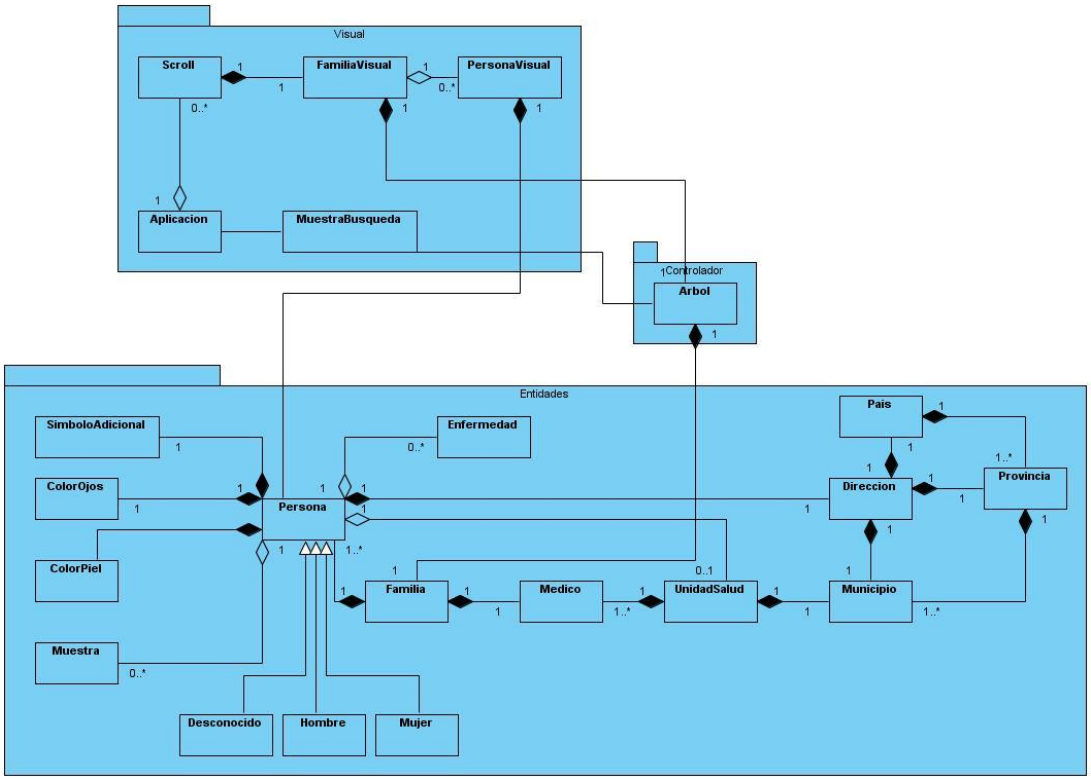


Fig. 7 DC_CU_Buscar en Arbol.

2.6.2 Diagrama de clases de la aplicación web

.

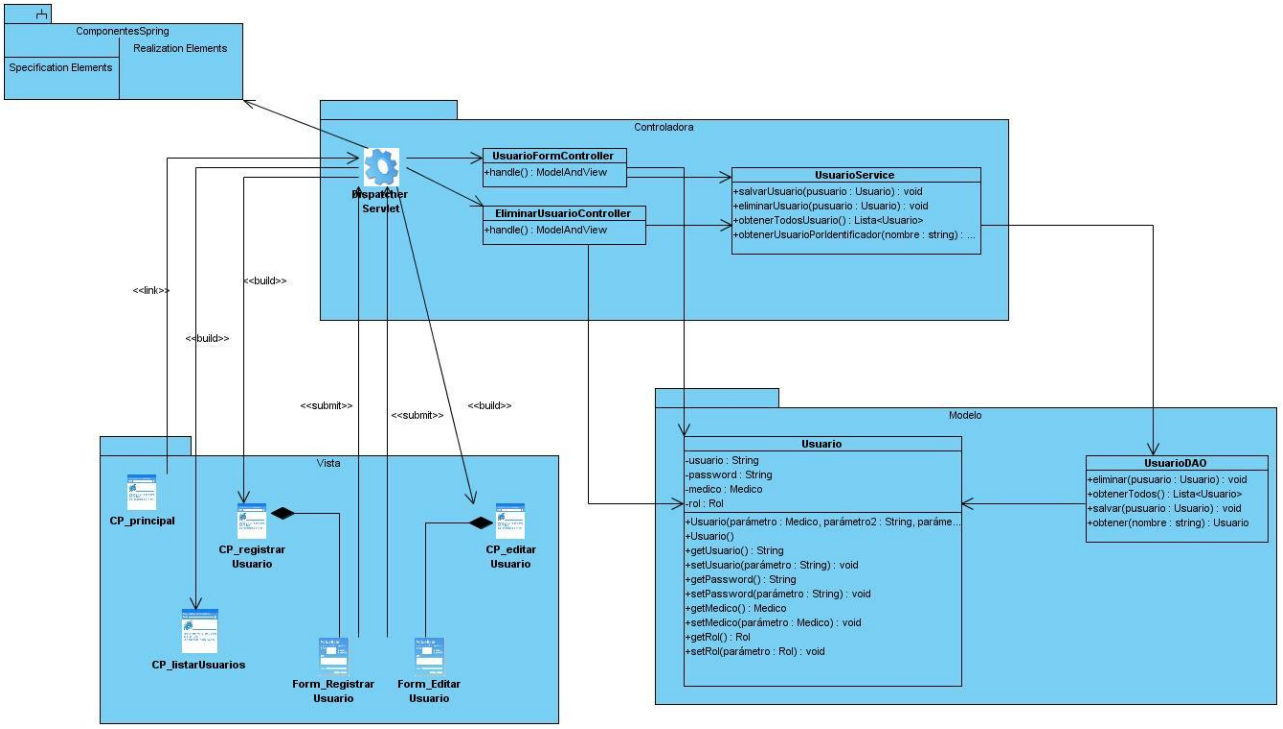


Fig. 8 DCD_CU_GestionarUsuario.

2.7 Identificación de las clases persistentes

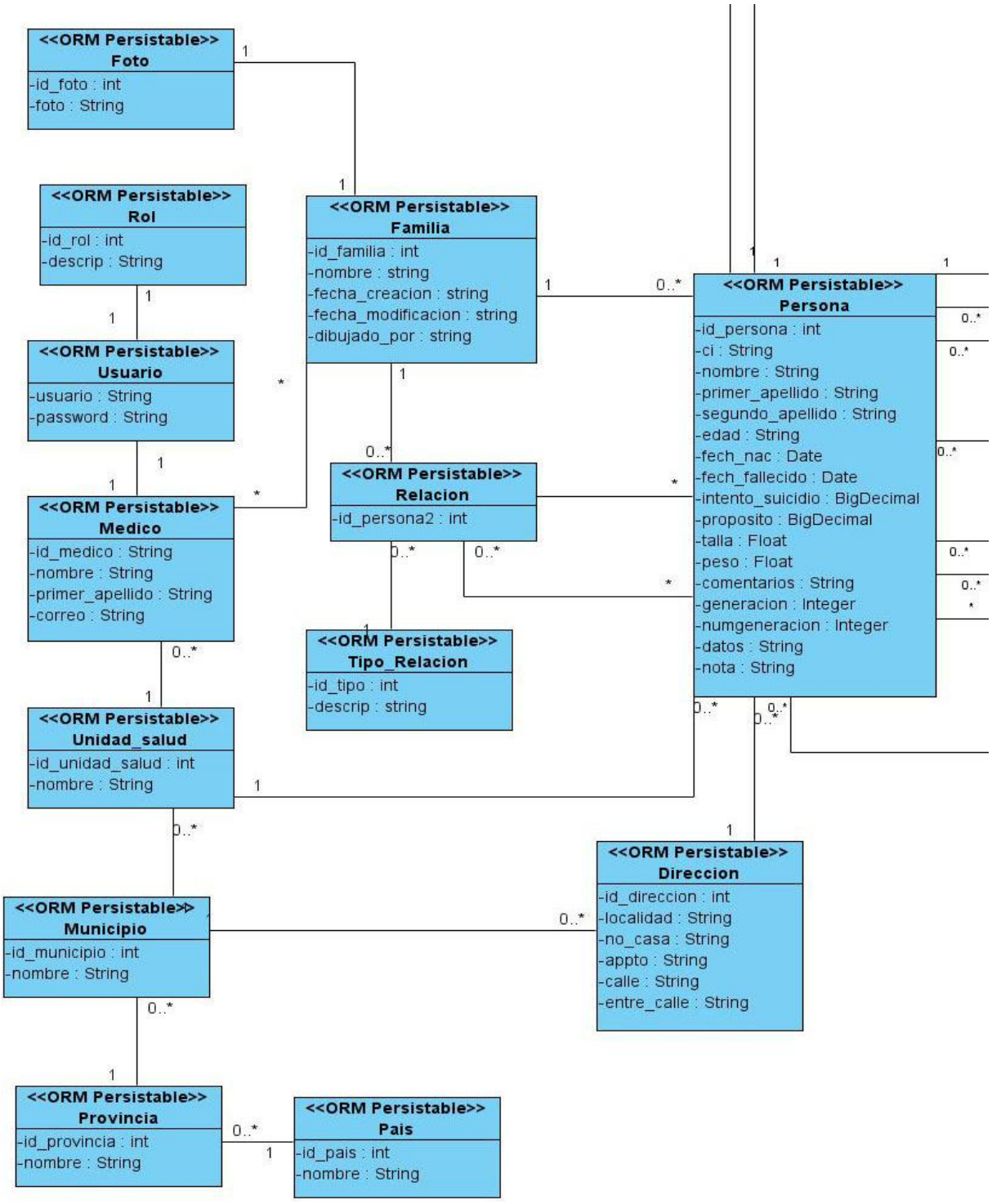
La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes son aquellas cuyos objetos deben ser almacenados en algún repositorio como una base de datos relacional. Estos objetos deben contener sólo los datos de las entidades que se modelan y sus relaciones con otros objetos persistentes.

Las clases persistentes, por lo general, tienen como origen las clases clasificadas como entidad porque modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso. Todas las clases identificadas en el dominio del análisis son persistentes.

El diagrama de clases persistentes se utiliza para modelar la estructura lógica de la BD, donde las clases son representadas por tablas y los atributos de clase por columnas. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos (HERNÁNDEZ, 2004).

En la figura 9 se muestra el diagrama de clases persistentes del sistema alasARBOGEN 2.0, obtenido a partir de los diagramas de clases del diseño. En él se representan las clases persistentes en su totalidad, así como las relaciones existentes entre ellas y los atributos que las componen.

2.7.1 Diagrama de clases persistentes



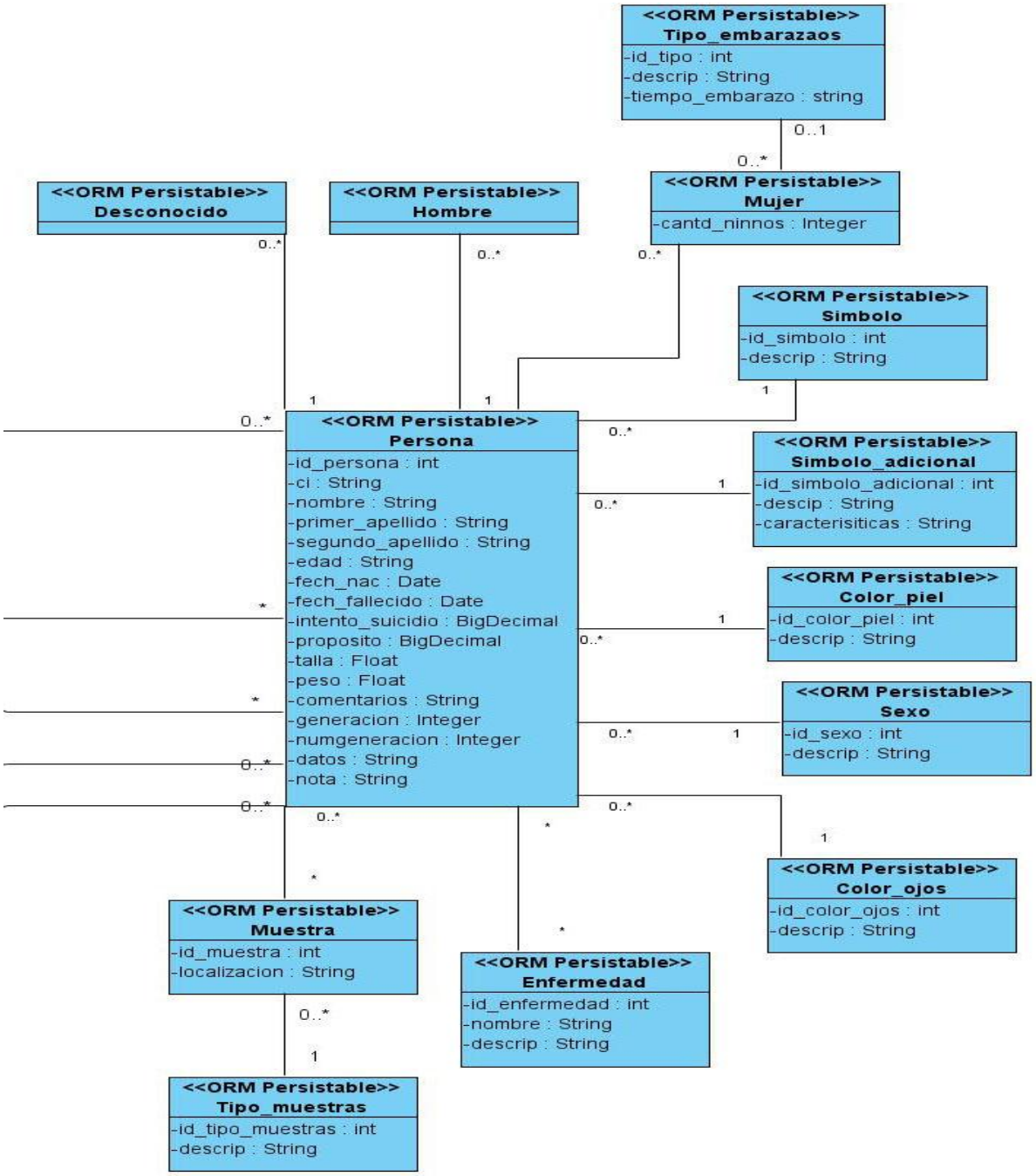


Fig.9 Diagrama de clases persistentes.

2.7.2 Descripción de las clases

Tabla 2.1 Descripción de la clase: Persona

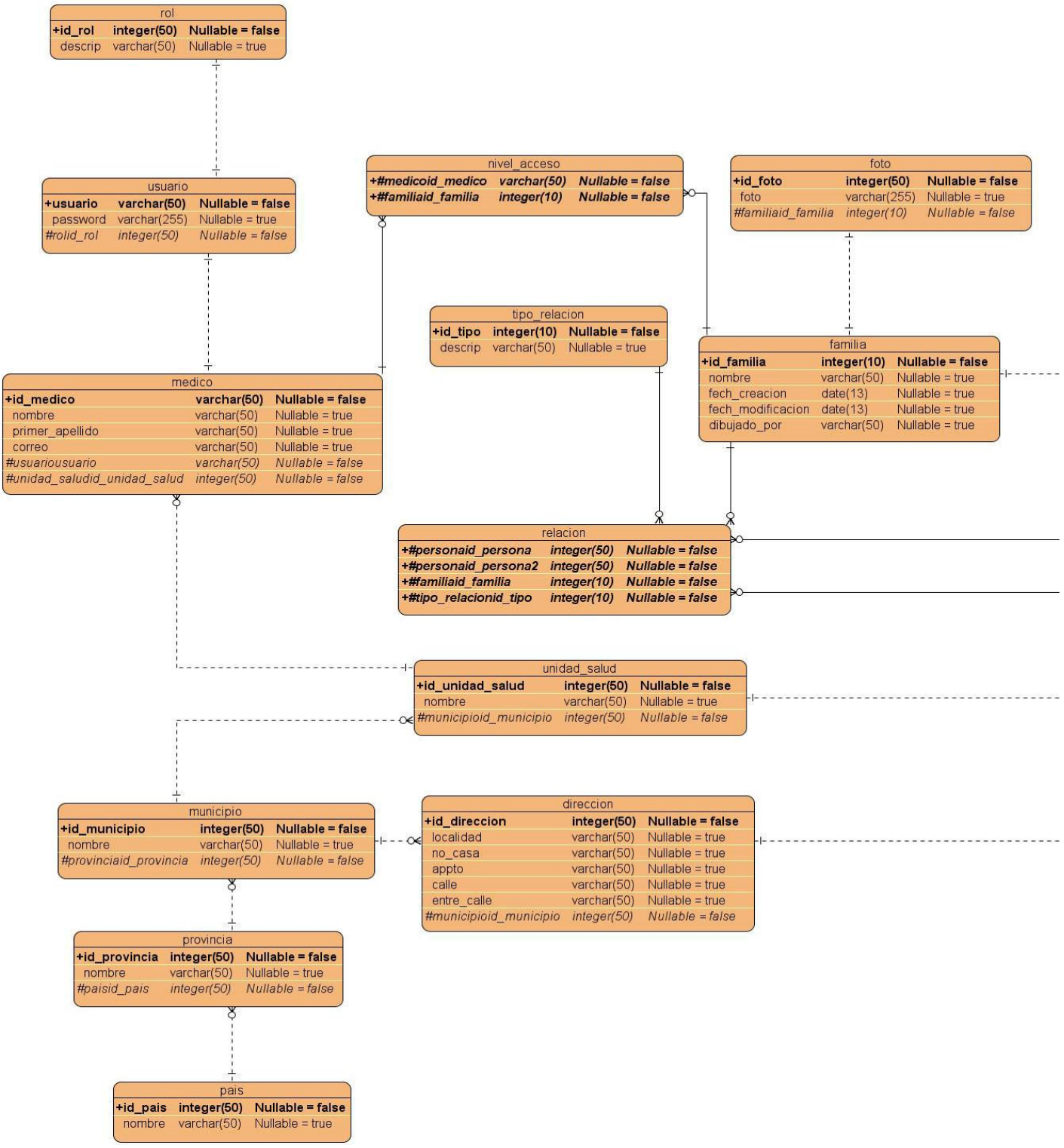
Tipo de clase: Entidad	
Atributo	**Tipo**
id	int
direccion	Direccion
enfermedad	Enfermedad
simbolo	Simbolo
sex	Sexo
colorOjos	ColorOjos
colorP	ColorPiel
familia	Familia
muestra	Muestra
simboloAdicional	SimboloAdicional
ci	String
nombre	String
primerApellido	String
segundoApellido	String
edad	int
fechaNacimiento	Date
fechaMuerte	Date
intentoSuicidio	boolean
proposito	boolean
talla	float
peso	float
comentarios	String
generacion	int
nota	String
datos	String
numGeneracion	int
unidadSalud	UnidadSalud
enfermos	List<PersonaEnfermedad>
muestras	List<MuestraPersona>

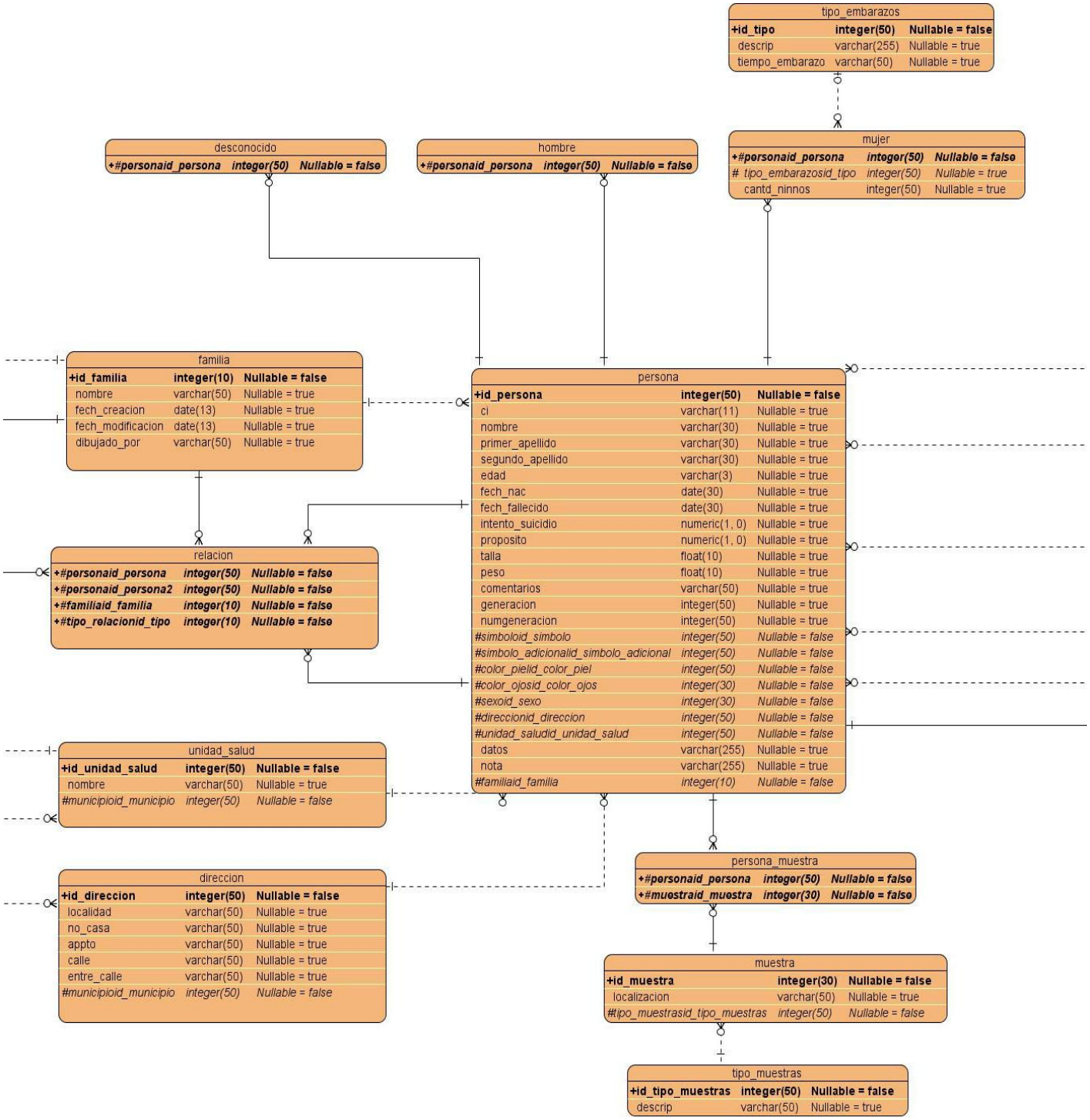
(Ver Complemento 2)

2.8 Diseño de la base de datos

Una vez conformado el diagrama de clases persistentes se puede obtener el modelo de datos que describe la representación física y lógica de los datos persistentes. Este diagrama consta de 28 tablas (Fig. 10).

2.8.1 Modelo de datos


rol
+id_rol integer(50) Nullable = false
descrip varchar(50) Nullable = true
usuario
+usuario varchar(50) Nullable = false
password varchar(255) Nullable = true
#rolid_rol integer(50) Nullable = false
nivel_acceso
+#medicoid_medico varchar(50) Nullable = false
+#familiaid_familia integer(10) Nullable = false
foto
+id_foto integer(50) Nullable = false
foto varchar(255) Nullable = true
#familiaid_familia integer(10) Nullable = false
tipo_relacion
+id_tipo integer(10) Nullable = false
descrip varchar(50) Nullable = true
familia
+id_familia integer(10) Nullable = false
nombre varchar(50) Nullable = true
fech_creacion date(13) Nullable = true
fech_modificacion date(13) Nullable = true
dibujado_por varchar(50) Nullable = true
medico
+id_medico varchar(50) Nullable = false
nombre varchar(50) Nullable = true
primer_apellido varchar(50) Nullable = true
correo varchar(50) Nullable = true
#usuariousuario varchar(50) Nullable = false
#unidad_saludid_unidad_salud integer(50) Nullable = false
relacion
+#personaid_persona integer(50) Nullable = false
+#personaid_persona2 integer(50) Nullable = false
+#familiaid_familia integer(10) Nullable = false
+#tipo_relacionid_tipo integer(10) Nullable = false
unidad_salud
+id_unidad_salud integer(50) Nullable = false
nombre varchar(50) Nullable = true
#municipioid_municipio integer(50) Nullable = false
municipio
+id_municipio integer(50) Nullable = false
nombre varchar(50) Nullable = true
#provinciaid_provincia integer(50) Nullable = false
direccion
+id_direccion integer(50) Nullable = false
localidad varchar(50) Nullable = true
no_casa varchar(50) Nullable = true
appto varchar(50) Nullable = true
calle varchar(50) Nullable = true
entre_calle varchar(50) Nullable = true
#municipioid_municipio integer(50) Nullable = false
provincia
+id_provincia integer(50) Nullable = false
nombre varchar(50) Nullable = true
#paisid_pais integer(50) Nullable = false
pais
+id_pais integer(50) Nullable = false
nombre varchar(50) Nullable = true


tipo_embarazos
+id_tipo integer(50) Nullable = false
descrip varchar(255) Nullable = true
tiempo_embarazo varchar(50) Nullable = true
desconocido
+#personaid_persona integer(50) Nullable = false
hombre
+#personaid_persona integer(50) Nullable = false
mujer
+#personaid_persona integer(50) Nullable = false
tipo_embarazosid_tipo integer(50) Nullable = true
cantd_ninnos integer(50) Nullable = true
familia
+id_familia integer(10) Nullable = false
nombre varchar(50) Nullable = true
fech_creacion date(13) Nullable = true
fech_modificacion date(13) Nullable = true
dibujado_por varchar(50) Nullable = true
persona
+id_persona integer(50) Nullable = false
ci varchar(11) Nullable = true
nombre varchar(30) Nullable = true
primer_apellido varchar(30) Nullable = true
segundo_apellido varchar(30) Nullable = true
edad varchar(3) Nullable = true
fech_nac date(30) Nullable = true
fech_fallecido date(30) Nullable = true
intento_suicidio numeric(1, 0) Nullable = true
proposito numeric(1, 0) Nullable = true
talla float(10) Nullable = true
peso float(10) Nullable = true
comentarios varchar(50) Nullable = true
generacion integer(50) Nullable = true
numgeneracion integer(50) Nullable = true
#simboloid_simbolo integer(50) Nullable = false
#simbolo_adicionalid_simbolo_adicional integer(50) Nullable = false
#color_pielid_color_piel integer(50) Nullable = false
#color_ojosid_color_ojos integer(30) Nullable = false
#sexoid_sexo integer(30) Nullable = false
#direccionid_direccion integer(50) Nullable = false
#unidad_saludid_unidad_salud integer(50) Nullable = false
datos varchar(255) Nullable = true
nota varchar(255) Nullable = true
#familiaid_familia integer(10) Nullable = false
relacion
+#personaid_persona integer(50) Nullable = false
+#personaid_persona2 integer(50) Nullable = false
+#familiaid_familia integer(10) Nullable = false
+#tipo_relacionid_tipo integer(10) Nullable = false
unidad_salud
+id_unidad_salud integer(50) Nullable = false
nombre varchar(50) Nullable = true
#municipioid_municipio integer(50) Nullable = false
direccion
+id_direccion integer(50) Nullable = false
localidad varchar(50) Nullable = true
no_casa varchar(50) Nullable = true
appto varchar(50) Nullable = true
calle varchar(50) Nullable = true
entre_calle varchar(50) Nullable = true
#municipioid_municipio integer(50) Nullable = false
persona_muestra
+#personaid_persona integer(50) Nullable = false
+#muestraid_muestra integer(30) Nullable = false
muestra
+id_muestra integer(30) Nullable = false
localizacion varchar(50) Nullable = true
#tipo_muestrasid_tipo_muestras integer(50) Nullable = false
tipo_muestras
+id_tipo_muestras integer(50) Nullable = false
descrip varchar(50) Nullable = true

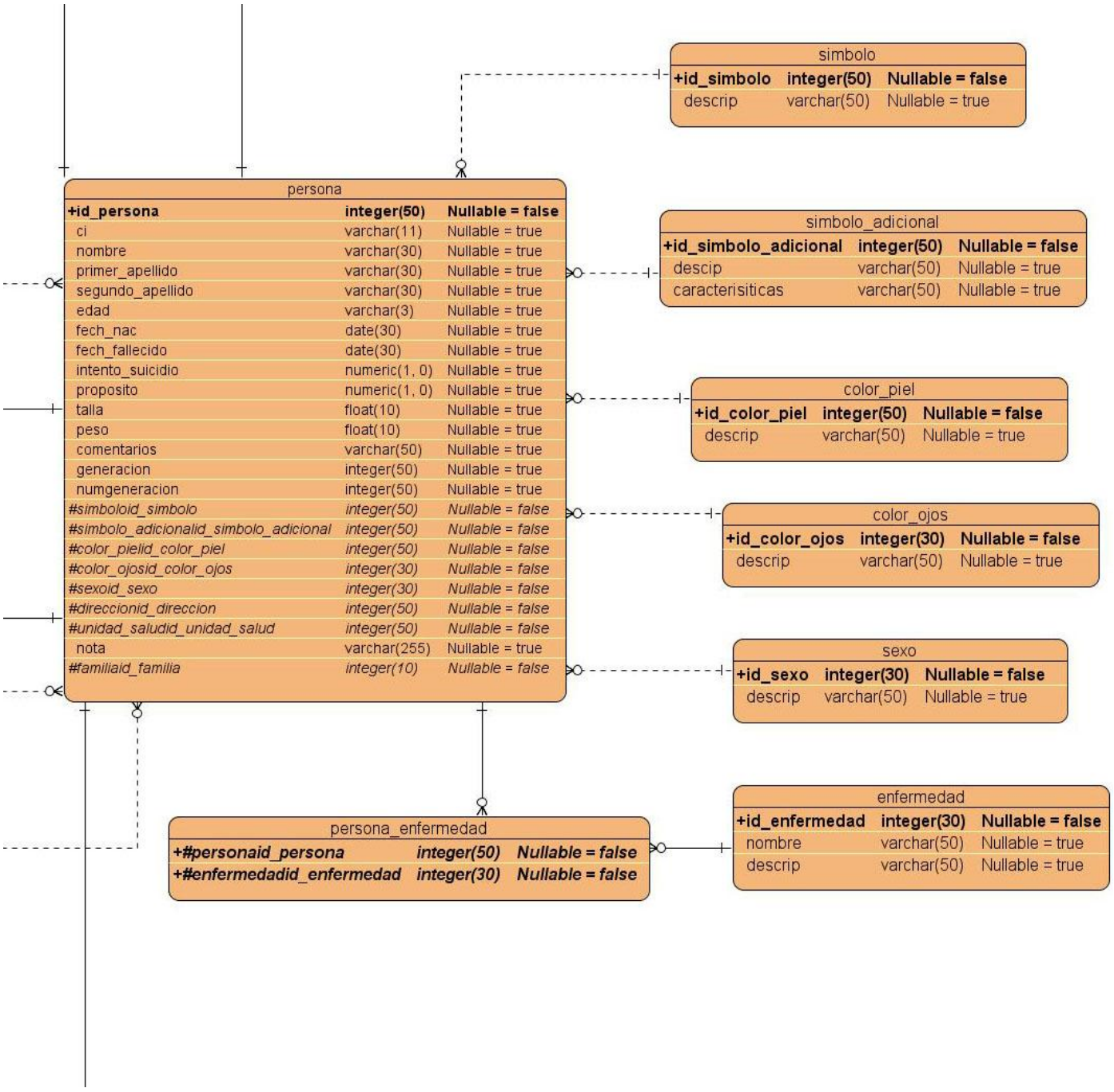


Fig. 10 Modelo de datos.

2.8.2 Descripción de las tablas del modelo de datos

Tabla 2.27 Descripción de la tabla: persona

Nombre: persona		
Descripción: Se almacenan todos los datos significativos para el sistema que posee una persona.		
Atributo	**Tipo**	**Descripción**
id_persona	INTEGER	Llave primaria, identificador de la persona.
id_direccion	INTEGER	Llave foránea, identificador de la dirección.
id_enfermedad	INTEGER	Llave foránea, identificador de la enfermedad.
id_simbolo	INTEGER	Llave foránea, identificador del símbolo.
id_sexo	INTEGER	Llave foránea, identificador del sexo.
id_color_ojos	INTEGER	Llave foránea, identificador del color de ojos.
id_color_piel	INTEGER	Llave foránea, identificador del color de piel.
id_simbolo_adicional	INTEGER	Llave foránea, identificador del símbolo
ci	VARCHAR	Número del carnet de identidad de la persona.
nombre	VARCHAR	Nombre de la persona.
primer_apellido	VARCHAR	Primer apellido de la persona.
segundo_apellido	VARCHAR	Segundo apellido de la persona.
edad	VARCHAR	La edad de la persona.
fech_nac	DATE	La fecha de nacimiento de la persona.
fech_fallecido	DATE	Si la persona esta fallecida, la fecha en que
intento_suicidio	NUMERIC	Si la persona intentó suicidarse o no.
proposito	NUMERIC	Indica si la persona es objeto de estudio.
talla	FLOAT	La talla de la persona.
peso	FLOAT	El peso de la persona.
comentarios	VARCHAR	Un comentario que se le realice a una persona.
nota	VARCHAR	Descripción de la persona.
datos	VARCHAR	Algún dato específico de la persona.
generación	INTEGER	Indica a que generación pertenece la persona.
numgeneración	INTEGER	Indica el lugar dentro de la generación.
id_unidad_salud	INTEGER	Llave foránea, identificador de la tabla unidad de
id_familia	INTEGER	Llave foránea, identificador de la tabla familia.

(Ver Complemento 1)

2.9 Implementación de la capa de acceso a datos

La capa de acceso a datos utiliza las entidades (que se encuentran ubicados en un paquete), del sistema que se van a referenciar en el acceso a datos; además, esta capa contiene varios elementos organizados de la siguiente forma:

En el paquete mapeo se encuentran los XML de configuración a través de los que se mapean las entidades con las tablas de la base de datos.

En el paquete DAO se maneja todo lo referente a la capa de acceso a datos de la aplicación, de tal forma que el sistema logra abstraerse de todo lo referente a la persistencia de los datos y enfocarse en la tecnología de persistencia utilizada. Esto implica que al cambiar o actualizar la tecnología de persistencia no necesariamente se verán afectadas otras partes de la aplicación. Por cada entidad persistente del dominio se crea un objeto DAO. Este objeto va a constar de una interfase que expone las funcionalidades necesarias y de una implementación para cada interfase (relacionada directamente con la tecnología de persistencia utilizada). Esta implementación se encarga de todas las tareas como crear, eliminar y modificar su entidad correspondiente.

A continuación se muestra como se define en los archivos XML del paquete config, todas las propiedades para la conexión:

```
<property name="hibernate.connection.driver_class">org.postgresql.Driver
</property>
<property name="hibernate.connection.password">postgres
</property>
<property name="hibernate.connection.url">jdbc:postgresql://10.31.8.182:5901/
arbogenultimo </property>
<property name="hibernate.connection.username">postgres
</property>
<property name="hibernate.default_schema">public</property>
<property name="hibernate.dialect">org.hibernate.dialect.DB2Dialect
</property>
```

Servicio de persistencia y carga

En este servicio, la persistencia consiste en capturar los datos enviados de la aplicación web y/o de escritorio a través del componente de comunicación y persistirlos en la base de datos. Por otra parte se

cargan los datos de la base de datos y se envían a la aplicación correspondiente, a través del mismo componente.

Para persistir los datos en la base de datos, las aplicaciones envían la información de un objeto familia y el usuario (para la autenticación), a través del componente de comunicación, donde la información va a estar representada en un grafo. Posteriormente se recorre el grafo y los datos obtenidos se insertan en la BD a través de los DAO.

Para la carga de los datos, la información es extraída a través de los DAO y mediante un algoritmo es transformada en un grafo. Este es enviado a las diferentes aplicaciones utilizando el componente antes mencionado.

CONCLUSIONES

En el capítulo 2 se realizó un análisis de las principales ineficiencias del sistema alasARBOGEN 1.0 y se describieron las funcionalidades a automatizar con el objetivo de superar esas ineficiencias.

Mediante la elaboración del Modelo de Dominio se estudiaron las entidades relacionadas directa e indirectamente al sistema, debido a que no se pudo definir de manera clara y precisa un Modelo de Negocio. Basado en todo el análisis realizado se representó, a través de un diagrama, la arquitectura del nuevo sistema a desarrollar acompañado por la descripción de las funciones de sus principales componentes.

Se expusieron los requisitos funcionales que deben ser tenidos en cuenta a la hora de modelar la base de datos y se explicaron los requisitos no funcionales que fueron definidos de manera general.

Luego de analizar los diagramas de clases de diseño fueron identificadas las clases que deben persistir, se realizó el diagrama correspondiente con el objetivo de modelar la estructura lógica de la BD y se describieron las clases identificadas. Una vez obtenida esta información se elaboró el modelo de datos que describe la representación física y lógica de los datos persistentes. Este cuenta con 28 tablas explicadas a través de sus principales características.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

INTRODUCCIÓN

Este capítulo comenzará con un análisis de la validación teórica del diseño, mediante el estudio y profundización de conceptos significativos como la integridad, seguridad de la base de datos y redundancia de la información. También se incluye todo el proceso para normalizar la BD, imprescindible luego del modelado de esta.

Un segundo paso será la validación funcional con el objetivo de comprobar que la BD cumple con los requisitos funcionales definidos. Se harán varias pruebas con el fin de simular una carga de producción real y observar el comportamiento de la BD en esta situación.

Por último se realizará un estudio de la técnica conocida como balanceo de carga. Su objetivo será proponer la aplicación de la técnica antes mencionada, para alcanzar un tiempo de respuesta mucho más eficiente de la BD, en un entorno más complejo con un mayor número de usuarios.

3.1 Validación teórica del diseño

La validación teórica del diseño incluye fundamentalmente un análisis muy detallado de la integridad de la información, característica altamente deseada porque asegura la calidad de almacenamiento y disponibilidad de los datos. Con su tratamiento se evitan errores de entrada introducidos por los usuarios o cualquier otra circunstancia de intento de violación de la información existente en la base de datos.

3.1.1 Integridad

La integridad es uno de los factores más importantes a la hora de realizar el diseño de una base de datos. Esta se divide en varios aspectos como son:

- Integridad referencial

Es un sistema de reglas, utilizadas en la mayoría de las bases de datos relacionales, para asegurar la validez de los registros de tablas relacionadas, y garantizar que no sean borrados o modificados los datos relacionados, lo que provocaría errores de integridad.

Cuando se define una columna como llave foránea, las filas de la tabla pueden contener en esa columna o bien el valor nulo (ningún valor), o bien un valor que existe en la otra tabla. Eso es lo que se denomina integridad referencial y consiste en que datos que referencian otros datos (llaves foráneas) deben ser siempre correctos. La integridad referencial hace que el sistema gestor de la base de datos se asegure que no haya en las llaves foráneas valores que no estén en la tabla principal. La integridad referencial se activa en cuanto se crea una llave foránea, a partir de este momento se comprueba su valor cada vez que se modifiquen datos que puedan alterarla (*aula.com*, 2000).

Para mantener esta integridad en la base de datos se utilizaron llaves foráneas o externas, las cuales obligan a que los valores introducidos en las columnas, marcadas por esta restricción, correspondan a valores en la tabla referenciada, y permiten realizar acciones en caso de actualización o eliminación de los valores. Se definió que la actualización o eliminación de datos se realice en cascada.

El SGBD utilizado (PostgreSQL) maneja de manera muy positiva la integridad referencial y asegura que dentro de un diseño bien elaborado, las referencias por llaves foráneas sean las correctas.

- Integridad de dominio

Viene dada por la validez de las entradas para los atributos en las tablas de una BD. Puede ser necesario definir más de una restricción de dominio para describir por completo un dominio. Mientras mayor sea la cantidad de reglas definidas, mejor será el funcionamiento de la base de datos.

Existe una integridad de dominio básica, como no poder introducir letras en un campo destinado para almacenar números. Además hay normas o reglas de integridad de dominio, que pueden indicar en qué campos es necesario introducir obligatoriamente algún valor (no pueden tener como valor NULL), para que la base de datos no tenga datos sin conectar, cuando se tengan relaciones o dependencias entre tablas. También sucede en el caso del formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KE (*SQL Server 2008* 2009).

Para garantizar la integridad de dominio se crearon reglas para validar aquellos datos que se deben encontrar dentro de un rango específico de valores, donde se verifica que solo sean introducidos datos correctos en el sistema.

- Integridad de la entidad

Las restricciones de entidades aseguran la integridad de las entidades que son modeladas por el sistema. En el nivel más simple, la existencia de una llave principal es una restricción de entidad que impone la regla: *cada entidad debe estar identificada de forma única*. En esta no está permitido que algún componente de la llave primaria acepte valores nulos (LUQUE, 2003).

En el caso de la BD que se está analizando, cada entidad tiene definida su llave primaria, que no es nula y no se repite.

3.1.2 Normalización de la Base de datos

La normalización es una técnica para diseñar la estructura lógica de los datos de un sistema de información en el modelo relacional. Es una etapa posterior a la correspondencia entre el esquema conceptual y el esquema lógico, que elimina las dependencias, no deseadas, entre atributos.

El proceso de normalización es un estándar que consiste, básicamente, en un proceso de conversión de las relaciones, que evita:

- Redundancia de los datos: repetición de datos en un sistema.
- Anomalías de actualización: inconsistencias de los datos como resultado de datos redundantes y actualizaciones parciales.
- Anomalías de borrado: pérdidas no intencionadas de datos debido a que se han borrado otros datos.
- Anomalías de inserción: imposibilidad de adicionar información en la base de datos debido a la ausencia de otros datos.

Existen varias formas normales para las relaciones: primera forma normal (1NF), segunda forma normal (2NF), tercera forma normal (3NF), entre otras. Se dice que una relación está en una determinada forma normal si satisface un cierto conjunto de restricciones.

- 1NF: Una relación está en 1NF si todos los dominios simples subyacentes contienen sólo valores atómicos.

- 2NF: Una relación está en 2NF si está en 1NF y todos los atributos no llaves (o que no formen parte de la llave) dependen por completo de la llave primaria.

- 3NF: Una relación está en 3NF si está en 2NF y todos los atributos no llave dependen de manera no transitiva de la llave primaria (CHÁVEZ, 2005).

La base de datos en cuestión, actualmente se encuentra en 3FN, porque cumple con las especificaciones de las tres formas expuestas antes; es decir, en ella no existen atributos multievaluados ni dependencias transitivas entre las relaciones, como tampoco redundancia de la información. La 3FN es la más usada en la totalidad de los productos que utilizan bases de datos porque garantiza una redundancia casi nula de información.

En el proceso de normalizar la base de datos de la solución propuesta se presentaron situaciones como la siguiente:

La tabla *persona* se encontraba compuesta por el atributo *direccion* (Fig. 11), el cual se puede clasificar como un atributo multievaluado, dado que está compuesto por otros atributos como es el caso de *localidad*, *no_casa, calle,* entre otros.

Fig. 11 Ejemplo de normalización de la tabla persona.

Para llevar la base de datos a 1FN se requiere eliminar los atributos multievaluados, por lo que se conformó una tabla *direccion (Fig. 12)* relacionada con la tabla *persona* para eliminar *direccion* como atributo.

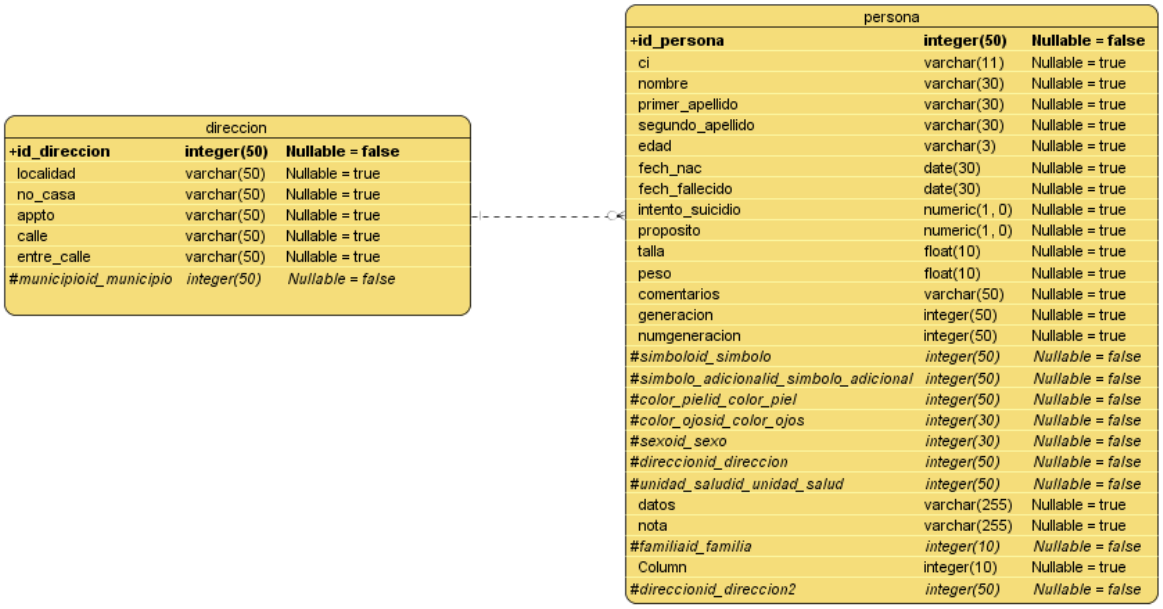


Fig. 12 Ejemplo de normalización de la tabla persona y de la tabla dirección.

3.1.3 Análisis de redundancia de información

La redundancia de datos es aquella información duplicada o almacenada varias veces en la misma base de datos. Esto dificulta la tarea de modificación de datos y es el motivo más frecuente de su inconsistencia. Además representa un mayor espacio de almacenamiento, que influye negativamente en el tiempo de acceso a los datos. Con un buen diseño de una base de datos se logrará evitar la aparición de información repetida o redundante en el sistema (CORONADO, 2004).

Luego de llevar la BD del sistema alasARBOGEN 2.0 a 3FN, quedó libre de redundancias; es decir, fue eliminada por completo la presencia de datos repetidos innecesariamente.

Uno de los métodos utilizados para eliminar estas inconsistencias es el uso de los nomencladores para todos aquellos datos comunes dentro de la base de datos. Este método fue aplicado en el diseño del sistema que se está desarrollando.

Entre los nomencladores definidos en la BD del sistema alasARBOGEN 2.0 se pueden mencionar símbolo, tipo_embarazos, color_piel, color_ojos, sexo, enfermedad, municipio, provincia, país entre otros. (Ver figura 10).

3.1.4 Análisis de la seguridad de la base de datos

La seguridad, dentro de la base de datos realizada como parte de la solución, se maneja a través del SGBD utilizado. PostgreSQL materializa la seguridad en tres aspectos:

- Seguridad en la manipulación de los ficheros.
- Seguridad en el acceso de los clientes.
- Definición de los privilegios para acceder a los objetos de la base de datos a los usuarios.

La información más crítica se encuentra en $PGDATA. Todos los ficheros deben pertenecer al usuario postgres y este es el único que puede leer, escribir y ejecutar sobre los directorios.

Todos los usuarios emplean las mismas conexiones locales vía *sockets* y se puede restringir el uso para algunos usuarios del SO. Los permisos de este socket se pueden configurar en postgresql.conf con los parámetros unix_socket_directory, unix_socket_group y unix_socket_permission.

Seguridad en el acceso de los clientes

Es importante poder definir desde qué ordenador se pueden conectar a la base de datos, así como a través de qué usuarios y a qué bases de datos se conectan.

La configuración de este nivel de seguridad se realiza en los ficheros pg_hba.conf (hba = host base dauthentication) y pg_ident.conf.

Definición de los privilegios a los usuarios para acceder a los objetos de la base de datos

Los privilegios son definidos mediante tres aspectos fundamentales:

- Autorización de alto nivel en la creación de roles (SUPERUSER, CREATEUSER, CREATEDB).
- Utilización de las sentencias SQL GRANT y REVOKE para la gestión de los privilegios sobre los objetos de las bases de datos.
- Empleo del concepto de pertenencia a rol y herencia de privilegios para mejorar la eficiencia de la gestión de privilegios del sistema (ALARCÓN).

3.2 Validación Funcional

Para comprobar que la BD cumple con los requisitos funcionales definidos, es necesario realizarle pruebas antes de dar por terminado su proceso de diseño y creación. El propósito de estas pruebas es simular una carga de producción real y observar cómo se comporta la base de datos bajo cargas intensivas. Esto permite solucionar los problemas de rendimiento, antes de poner la BD en marcha.

3.2.1 Llenado voluminoso e inteligente de la base de datos

Las pruebas de volumen centran su trabajo en poblar la BD de grandes cantidades de información para determinar si hay algún momento donde esta alcance sus límites de almacenamiento y cause fallas en el sistema, así se identifica la carga máxima que puede manejar la base de datos en un período determinado. En el mundo existen algunas herramientas para el llenado voluminoso de BD.

Para la realización de pruebas de llenado voluminoso e inteligente a la BD se utilizó la herramienta EMS Data Generator for PostgreSQL. Esta herramienta permite la generación de datos para una o varias tablas a la vez, donde se definen para cada campo el rango de valores admisibles, en dependencia del tipo, además de la cantidad de tuplas que se desean generar. De esta manera se valida de forma automática la integridad referencial, ya que genera los datos que provienen de otras tablas para evitar errores. La generación de los datos es de forma aleatoria e incremental (*Free Dowload Manager*, 2007).

Para representar estas pruebas se muestra a continuación un ejemplo de los resultados obtenidos:

Tabla	Cantidad de registros	Generados a la BD (seg)
familia	2500	6
tipo_relacion	2500	5

color_ojos	2500	5
color_piel	2500	5
simbolo_adicional	2500	6
tipo_simbolos	2500	6
símbolo	2500	9
enfermedad	2500	6
muestra	2500	6
tipo_ muestras	2500	5
país	2500	5
provincia	2500	6
municipio	2500	6
rol	2500	4
tipo_embarazos	2500	5
unidad_salud	2500	6
persona	2500	19
usuario	2500	6
medico	2500	7
nivel_acceso	2500	6
direccion	2500	7
sexo	2500	5
relacion	2500	6
hombre	2500	5
mujer	2500	7
desconocido	2500	5

En la realización de la prueba de volumen a la base de datos, los resultados obtenidos muestran un buen comportamiento y una rápida respuesta de la BD frente a un gran flujo de información, teniendo en cuenta la búsqueda que realiza el software en el catálogo de llaves para validar las llaves foráneas y los dominios de los campos.

3.2.2 Prueba de carga intensiva

Para la realización de este tipo de pruebas a la BD del sistema alasARBOGEN2.0 se utilizó la herramienta EMS SQL Query 2010 for PostgreSQL, con el objetivo de probar su correcto funcionamiento y los tiempos de ejecución. Las consultas generadas para la realización de pruebas se seleccionan de acuerdo con las

operaciones que se esperan resulten las más utilizadas una vez implantada la BD. A continuación se muestran ejemplos de ello.

- Listar todos los usuarios que son administradores

```
SELECT
  public.usuario.usuario
FROM
  public.usuario
WHERE
  public.usuario.id_rol = 1
```

Para un volumen de 1000 tuplas se generaron 181 resultados en un tiempo de 125 ms.

- Listar los árboles creados en la unidad de salud Frank País.

```
SELECT
  public.unidad_salud.nombre,
  public.arbol.fech_modificacion,
  public.arbol.fech_creacion,
  public.arbol.descirp,
  public.arbol.id_arbol,
  public.unidad_salud.id_unidad_salud
FROM
  public.unidad_salud
  INNER JOIN public.medico ON (public.unidad_salud.id_unidad_salud = public.medico.id_unidad_salud)
  INNER JOIN public.nivel_acceso ON (public.medico.id_medico = public.nivel_acceso.id_medico)
  INNER JOIN public.arbol ON (public.nivel_acceso.id_arbol = public.arbol.id_arbol)
```

WHERE

public.unidad_salud.id_unidad_salud = 2

Para un volumen de 1000 tuplas se generaron 111 resultados en un tiempo de 62 ms.

- Listar las personas que hayan intentado suicidarse

SELECT

public.persona.primer_apellido,

public.persona.nombre,

public.persona.segundo_apellido

FROM

public.persona

WHERE

public.persona.intento_suicidio = 1

Para un volumen de 1000 tuplas se generaron 131 resultados en un tiempo de 94 ms.

A pesar de que no fueron definidos, por parte de los clientes, requisitos que señalen los tiempos de respuestas admisibles para la aplicación, se consideran aceptables los resultados obtenidos en las pruebas. Esto se debe a que el volumen de tuplas generadas contra el tiempo de respuesta en ms ofrece un óptimo resultado.

3.3 Propuesta de balanceo de carga en un servidor de base de datos

Con el acceso simultáneo de un gran número de usuarios a un servidor de BD, surge un grupo de problemas que deben ser tratados, como es el alcance máximo de procesamiento. Esto trae consigo los denominados “cuellos de botellas”, que representan una disminución del rendimiento dentro del servidor en cuestión, por lo que el tiempo de respuesta a las consultas realizadas a la BD es cada vez mayor.

El sistema de base de datos desarrollado puede, en un futuro, ser desplegado en un entorno mucho más complejo, donde se incremente considerablemente el número de usuarios. Esto traería consigo los inconvenientes antes expuestos, de ahí la propuesta de utilizar una técnica que equilibre el trabajo a realizar. Esta técnica es capaz de repartir de forma equitativa las solicitudes de los usuarios entre varios servidores dispuestos al mismo nivel. El objetivo es que los servidores se encuentren trabajando de forma alineada, como si constituyeran un solo servidor. Esta técnica se denomina “balance o balanceo de carga” (*ARQHYS*, 2010).

La técnica balanceo de carga se puede desarrollar a través de la implementación de un clúster.

3.3.1 Clúster

Un clúster es la unión, mediante una red de alta velocidad, de múltiples ordenadores de tal forma que el conjunto es visto como un único ordenador (Anexo 3).

Existen diferentes tipos de clúster:

- High Performance Clúster.
- Activo / Pasivo.
- Activo / Activo.
- Grid Computing (LAZO, 2008).

Se implementa un clúster con el objetivo principal de alcanzar características tales como:

- Alto rendimiento.
- Alta disponibilidad.
- Equilibrio de carga.
- Escalabilidad.

Para que un sistema clúster funcione no es necesario que todos los ordenadores dispongan del mismo hardware y sistema operativo (clúster heterogéneo). Este tipo de sistemas debe disponer de un interfaz de

manejo de clústeres, la cual se encargue de interactuar con el usuario y los procesos, distribuyendo la carga entre las diferentes máquinas del grupo.

Por norma general un clúster hace uso de diferentes componentes para su funcionamiento:

- Nodos (ordenadores o servidores).
- Sistema operativo.
- Conexión de red.
- Middleware (capa entre el usuario y el sistema operativo).
- Protocolos de comunicación y servicio.
- Aplicaciones.

Es necesario especificar que un middleware es el software que actúa entre el sistema operativo y las aplicaciones. Este software provee una única interfaz de acceso al sistema, denominada Single System Image (SSI). Este sistema se encarga de la escalabilidad del clúster y al detectar nuevas máquinas las añade al grupo.

Un middleware también provee al sistema de herramientas de mantenimiento para procesos pesados como son las migraciones, el balanceo de carga y la tolerancia de fallos (*¿Qué es exactamente un sistema Cluster?* , 2009).

3.3.2 Herramientas para el balanceo de carga dentro de una base de datos

Existen varias herramientas que cuentan, entre sus principales funcionalidades, con el balanceo de carga para una base de datos. A continuación se mencionan un grupo de herramientas dentro de esta categoría:

PgCluster:

Es un potente software recomendado para balancear carga dentro de una base de datos. Incorpora principalmente replicación de datos y tolerancia a fallos. Está caracterizado por ser síncrono; cuando un servidor deja de funcionar y se vuelve a poner en funcionamiento automáticamente inicia la sincronización

de datos. Brinda ventajas como es la replicación de datos por *querys* y se replican también objetos adicionales como las funciones definidas por el usuario. PgCluster funciona dentro de la misma base de datos.

Sequoia:

Es un software que tiene entre sus funcionalidades balanceo de carga y trabaja con cualquier base de datos con código Java. Esto se debe a que está hecho para clientes Java Database Connectivity (JDBC, por sus siglas en inglés), lo que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java y con la gran ventaja que representa una solución completa. Sequoia consta de una configuración un poco complicada, pero se encuentra bien documentado. Puede ser combinado con el software Myosotis, que crea otra capa intermedia (middleware) entre la aplicación y el software Sequoia (GARCÍA, JUAN CARLOS PUJOL).

Pgpool-II:

Limita el excedente de conexiones. PostgreSQL soporta un cierto número de conexiones concurrentes y rechaza las que superen esa cifra. Aumentar el límite máximo de estas, incrementa el consumo de recursos y afecta al rendimiento del sistema. Pgpool-II tiene también un límite máximo, pero las conexiones extras se mantienen en una cola, en lugar de devolver el error inmediatamente.

Pgpool-II mantiene abiertas las conexiones a los servidores PostgreSQL y las reutiliza siempre que se solicita una nueva con las mismas propiedades (nombre de usuario, base de datos y versión del protocolo). Ello reduce la sobrecarga y mejora la productividad global del sistema.

El uso de la función de replicación en Pgpool-II permite crear una copia en dos o más discos físicos, de modo que el servicio puede continuar sin parar los servidores en caso de fallo en algún disco.

Si se replica una base de datos, la ejecución de una consulta SELECT en cualquiera de los servidores devolverá el mismo resultado. Pgpool-II se aprovecha de la característica de replicación para reducir la carga en cada uno de los servidores PostgreSQL, distribuyendo las consultas SELECT entre los múltiples servidores, así aumenta la productividad global del sistema. El balanceo de carga es óptimo cuando hay muchos usuarios ejecutando varias consultas al mismo tiempo (SABATER, 2008).

Dado lo anteriormente planteado, se recomienda, el uso de Pgpool-II debido a que brinda la posibilidad de balancear carga en una BD, así como provee un *pool* de conexiones, su replicación es sincrónica y realiza consultas de forma paralela.

CONCLUSIONES

Se realizó la validación teórica del diseño, en la cual se definieron y analizaron términos como integridad, seguridad de la base de datos y redundancia de la información. Se logró llevar la BD a 3FN, por lo que cumple con las restricciones establecidas.

Mediante la validación funcional se desarrollaron las pruebas de llenado voluminoso e inteligente de la base de datos y la prueba de carga intensiva. Estas pruebas permitieron solucionar los problemas de rendimiento, antes de poner la BD en marcha.

Finalmente, se realizó la propuesta de implementar la técnica de balanceo de carga en el servidor de base de datos, si el sistema es desplegado en un entorno mucho más complejo. Esta técnica logrará resolver los problemas de respuesta a consultas dentro de la BD, que son comunes cuando un gran número de usuarios accede simultáneamente al servidor en cuestión.

CONCLUSIONES

- La implementación de la base de datos a través de sus 28 tablas, garantiza la persistencia de la información gestionada por el sistema.
- La capa de acceso a datos común garantiza un único acceso de la aplicación web y de escritorio a la base de datos.
- Los datos gestionados por la aplicación de escritorio son enviados de forma segura a través de HTTP Invoker.
- Se desarrolló una solución informática para gestionar los datos del sistema alasARBOGEN 2.0 que garantiza la seguridad, disponibilidad e integridad de la información procesada.

RECOMENDACIONES

Luego del desarrollo de la presente investigación se recomienda:

- Aplicar la técnica de balanceo de carga en el servidor de base de datos, cuando el sistema sea desplegado en un entorno de mayor complejidad.
- Realizar pruebas de estrés utilizando el Jmeter.

REFERENCIAS BIBLIOGRÁFICAS

2007. *Taringa* Disponible en: http://www.taringa.net/posts/downloads/884761/Modelado-de-Base-de-Datos-,--ER-Studio-6_0_1.html.

ALARCÓN, J. *Administración PostgreSQL*. Disponible en: http://www.gvpontis.gva.es/fileadmin/conselleria/images/Documentacion/migracionSwAbierto/SITARGES/manual.pdf

ALVAREZ, M. A. Descripción y características de este potente y moderno lenguaje de programación. 2001, nº Disponible en: http://www.desarrolloweb.com/articulos/497.php.

ÁLVAREZ, R. *Solución Infórmatica para el Centro Nacional de Balance Alimentario: Implementación de un componente de software para la transferencia de datos a través de Internet.* . UCI, 2008.

ANDALUCIA, J. D. *Marco de desarrollo de la Junta de andalucia* [Consultado el: 27 de febrero de 2010]. Disponible en: http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Hibernate.

ANDRÉS, M. M. M. [Consultado el: 22 de enero de 2010]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node4.html.

ARCEL LABRADA BATISTA, D. G. M. *Sistema de Gestión de Información de la Facultad 8. Diseño de la Base de Datos.* Universidad de las Ciencias Informáticas 2007.

ARQHYS [Consultado el: 24 de abril de 2010]. Disponible en: http://www.arqhys.com/contenidos/carga-balance.html.

aula.com [Consultado el: 14 de abril de 2010]. Disponible en: http://www.aulaclic.es/sql/b_8_1_1.htm.

Ayuda de RUP.

Ciberaula [Consultado el: 2 de marzo de 2010]. Disponible en: http://java.ciberaula.com/articulo/diseno_patrones_j2ee/

COMMONS., C. *Metodologías de desarrollo de software* [Consultado el: 27 de febrero de 2009]. Disponible en: http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

CORONADO, S. P. *MySQL con clase* [Consultado el: 26 de febrero de 2010]. Disponible en: http://mysql.conclase.net/curso/.

CURTO, J. [Consultado el: 16 de noviembre de 2009]. Disponible en: http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion/

CHÁVEZ, C. A. G. Diseño de base de datos relacionales. 2005, nº Disponible en: http://www.mailxmail.com/curso-diseno-base-datos-relacionales/diseno-logico-bases-datos.

Framework. 2010, nº Disponible en: http://www.radiobaracoa.icrt.cu/index.php?option=com_content&view=article&id=1842:framework-refencia-&catid=81:downloads&Itemid=101.

Free Dowload Manager [Consultado el: 21 de febrero de 2010]. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

GARCÍA, I. R. P. *Conexion Cubana* [Consultado el: 10 de noviembre de 2009]. Disponible en: http://www.conexioncubana.net/index.php?st=content&sk=view&id=2547&sitd=356&limit=1&limitstart=5

GARCÍA, J. C. P. *CLUSTERS DE SERVIDORES PARA APLICACIONES WEB*

GONZÁLEZ, C. D. *Curso PostgreSQL* [Consultado el: 26 de febrero de 2010]. Disponible en: http://www.usabilidadweb.com.ar/postgre.php.

GUI de PostgreSQL [Consultado el: 28 de febrero de 2010]. Disponible en: http://wiki.postgresql.org/wiki/Gu%C3%ADa_de_la_Comunidad_para_las_herramientas_GUI_de_PostgreSQL.

HERNÁNDEZ, A. *La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo.* . 2004, Disponible en: http://redalyc.uaemex.mx/redalyc/pdf/615/61570402.pdf.

ING ALFONSO CLARO ARCEO, I. R. R. R., ING. REYNALDO ÁLVAREZ LUNA. *SISTEMA PARA LA REPRESENTACIÓN DE ÁRBOLES GENEALÓGICOS* 3p. Disponible en: http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=2&ved=0CAoQFjAB&url=http%3A%2F%2Finformatica2009.sld.cu%2FMembers%2Faclaro%2Fsistema-para-la-representacion-de-arboles-genealogicos%2Fat_download%2Ftrabajo&rct=j&q=%EF%83%98%09Cyrillic%3A+Posee+todas+las+caracter%C3%ADsticas+necesarias+para+dibujar+l%C3%ADneas+gen%C3%A9ticas.&ei=_OrZS9aoFsH68AbSj9Vi&usg=AFQjCNHuise4VIgyLWVtncsYv9yvjKtwow

Itera [Consultado el: 27 de febrero de 2010]. Disponible en: http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=1.

JELSOFT. [Consultado el: 1 de marzo de 2010]. Disponible en: http://foro.ignetwork.net/showthread.php?t=15188.

Kioskea.net [Consultado el: 18 de febrero de 2010]. Disponible en: http://es.kioskea.net/contents/bdd/bddintro.php3

LAURA BERMEJO SANZ, E. G. M. Eclipse como IDE. nº Disponible en: http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf.

LEON, L. P. D. [Consultado el: 22 de enero de 2010]. Disponible en: http://family-tree-pilot.softonic.com/

LUQUE, L. C. *Programación Internet* [Consultado el: 19 de abril de 2010]. Disponible en: http://www.formauri.es/arrobamasmas/Cursos/index.php?apdo=05&curso=51&cap=5.

MINAY, C. Modelos de Base de Datos. nº p. 1,2. Disponible en: http://www.scribd.com/doc/17170125/Modelos-de-Bases-de-Datos.

MONTENEGRO, E. M. *PROGRAMACIÓN DISTRIBUIDA CON RMI* [Consultado el: 6 de marzo de 2010]. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=rmi.

MORATALLA, J. *Bases de datos con SQL*. publicado el: 11 de febrero 2010 de 2001, última actualización: 11 de febrero 2010. Disponible en: http://www.scribd.com/doc/4386117/Bases-De-Datos-Con-SQL-Server-2000.

¿Qué es exactamente un sistema Cluster? . 2009, Disponible en: http://www.taringa.net/posts/info/3214119/Qu%C3%A9-es-un-Cluster-y-c%C3%B3mo-funciona.html.

RAMOS, D. A. D. *Informatica en salud 2009* [Consultado el: 16 de noviembre de 2009]. Disponible en: http://informatica2009.sld.cu/conferencias/informatizacion-en-el-sistema-nacional-de-salud-de-cuba

RODRÍGUEZ, J. T. Q. *SISTEMAS DISTRIBUIDOS CON COMPONENTES*. 23 p. Disponible en: http://www.uv.mx/iiesca/revista/documents/distribuidos1999-2000.pdf.

SABATER, J. Replicación y alta disponibilidad de PostgreSQL con pgpool-II. 2008, nº Disponible en: http://linuxsilo.net/articles/postgresql-pgpool.html#sobre-pgpool-ii.

SERRANO, J. *UML - Lenguaje Unificado de Modelado* [Consultado el: 27 de febrero de 2010]. Disponible en: http://programando.foroactivo.com.es/umd-200901-ingenieria-del-software-f3/uml-lenguaje-unificado-de-modelado-t9.htm.

SIERRA, M. *INGENIERÍA DEL SOFTWARE I.* En *Trabajando con Visual Paradigm for UML*.

Soporte Linux [Consultado el: 26 de febrero de 2009]. Disponible en: http://www.soportelinuxdeguate.com/cms/index.php?option=com_content&view=article&id=53&Itemid=59.

SQL Server 2008 publicado el: 19 de abril de 2009, última actualización: 19 de abril. Disponible en: http://msdn.microsoft.com/es-es/library/ms184276.aspx.

TORRIJOS, R. L. *Programación en catellano* [Consultado el: 16 de febrero de 2010]. Disponible en: http://www.programacion.com/articulo/invocacion_remota_de_metodos_rmi_107/2.

VASQUEZ, L. Y. Lenguajes de programación. 1998, nº Disponible en: http://html.rincondelvago.com/lenguajes-de-programacion_10.html.

WebTaller [Consultado el: 28 de febrero de 2009]. Disponible en: http://www.webtaller.com/construccion/lenguajes/java/lecciones/que-es-java.php

BIBLIOGRAFÍA

1. *ANE*. (n.d.). Retrieved from http://aceproject.org/ace-es/topics/ve/ved/ved02/ved02a/ved02a03
2. *Balanceod e Carga*. (n.d.). Retrieved from Balanceod e Carga
3. Comunicaciones, M. d. (n.d.). *Ministerio de la Informática y las Comunicaciones*. Retrieved noviembre 22, 2009, from Ministerio de la Informática y las Comunicaciones: http://www.mic.gov.cu//hmicfunciones.aspx.
4. Craig Walls, R. B. *Spring in Action (Second Edition).*
5. García, L. (2008). Sistema de control de versiones: SUBVERSION.
6. *gfnjxfg*.
7. González, C. S. *Seguridad no intrusiva con Acegi Security System for Spring.*
8. González, H. S. (2003). *Manual Hibernate*.
9. Graells, D. P. (n.d.). Retrieved febrero 22, 2010, from http://www.pangea.orgperemarquestic.net
10. *IBM*. (n.d.). Retrieved from Proven best practices for software and systems delivery and implementation and for effective project management: http://www-01.ibm.com/software/awdtools/rup/
11. *Integra soluciones avanzadas*. (2010, febrero 4). Retrieved from http://www.integraas.com/Lanzada-la-version-Europa-de-Eclipse.html
12. Ivar Jacobson, G. B. (2000). *El Proceso Unificado de Desarrollo de Software*. Madrid.
13. Juan Medín Piñeiro, A. G. (2006). Hacia una arquitectura con JavaServer Faces, Spring, Hibernate y otros frameworks. Sevilla, España.
14. Lien Le Sanchez, R. R. (2008). alasARBOGEN: aplicación informática para la representación de árboles genealógicos. Ciudad de la Habana, Cuba.
15. *Linalco*. (n.d.). Retrieved from http://www.linalco.com/balanceo-de-carga-lb-linux.html
16. Lornel A. Rivas, M. P. (n.d.). Herramientas de Desarrollo de Software: Hacia la Construcción de una Ontología.
17. *LWP*. (n.d.). Retrieved from http://www.lawebdelprogramador.com/cursos/mostrar.php?id=72&texto=PostgreSQL

18. *Manual de Java*. (n.d.). Retrieved enero 11, 2010, from http://www.manual-java.com/manualjava/caracteristicas-java.html
19. Marinilli, M. (2001). *Java Deployment whit JNLP and WebStart.*
20. *masadelante.com*. (n.d.). Retrieved from http://www.masadelante.com/faqs/base-de-datos
21. Mendizábal, L. (2001). *GestioPolis*. Retrieved from http://www.gestiopolis.com/canales/gerencial/articulos/59/reglasdeorocom.htm
22. Morán, J. M. (2009). Framework para la capa de presentación de aplicaciones web.
23. Orallo, E. H. (n.d.). El Lenguaje Unificado de Modelado (UML).
24. *PostgreSQL*. (n.d.). Retrieved from http://www.postgresql.org/
25. Pressman, R. S. (2005). *Ingenieria de Software: Un enfoque práctico.*
26. Rico, M. A. (2006). Sistema de administración y control de renta de películas y libros vía web utilizando Spring. Cholula, Puebla, México.
27. Roberth G. Figueroa, C. J. (n.d.). METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.
28. Ruiz, J. J. (n.d.). APLICACIÓN WEB DE UNA EMPRESA DE PRODUCTOS HORTOFRUTÍCOLAS. Málaga.
29. Sancho, J. B. (n.d.). Aplicaciones web en cliente: Applets.
30. Stefan Küng, L. O. (n.d.). TortoiseSVN Un cliente de Subversion para Windows .
31. Sun Microsystems, Inc. (2002). *JavaServer Pages™ Standard Tag Library.*
32. *Ubunto*. (n.d.). Retrieved from http://www.guia-ubuntu.org/index.php?title=PostgreSQL
33. *Ubunto.es.* (n.d.). Retrieved from http://www.ubuntu-es.org/index.php?q=node/17862

ANEXOS

Anexo 1. *Diagramas de clase de la aplicación escritorio*

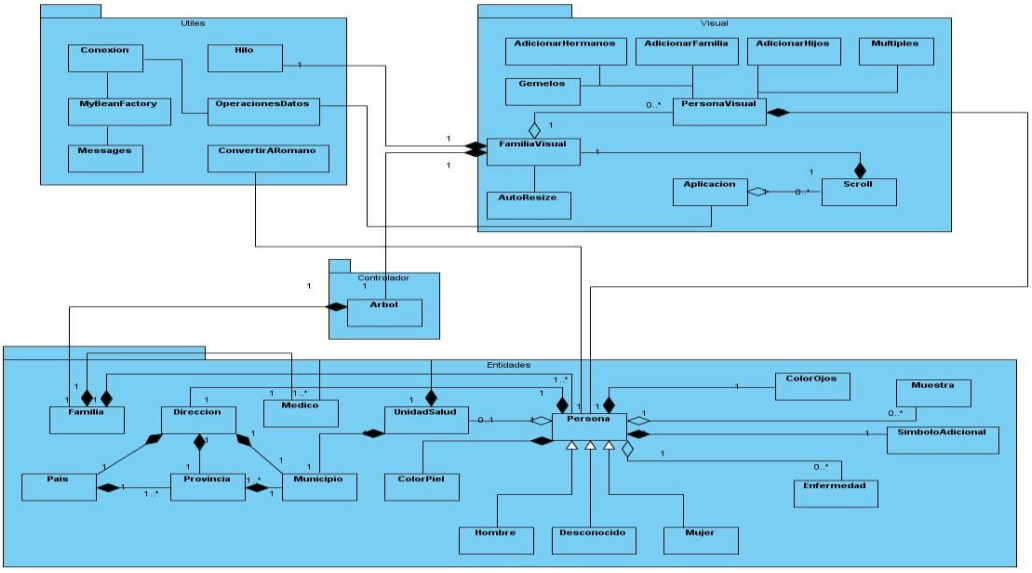


Fig.1.1 DC_CU_Gestionar gráficamente un individuo.

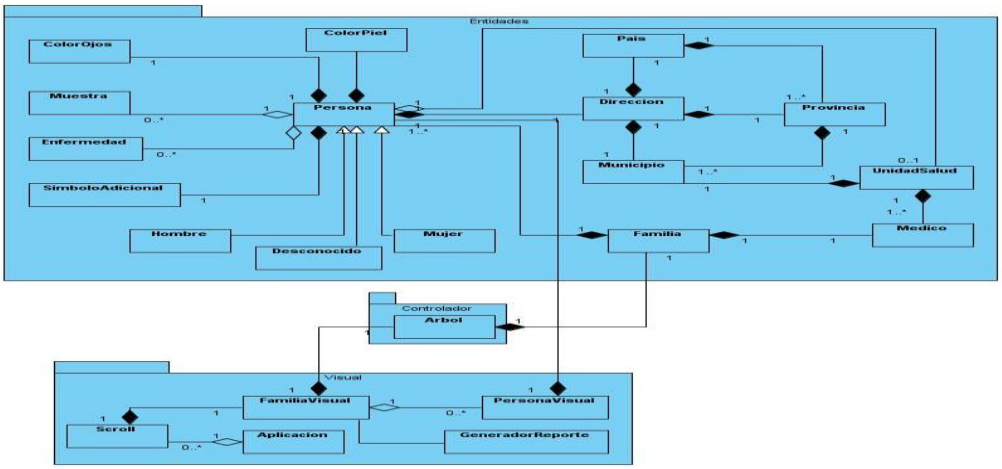


Fig.1.2 DC_CU_Gestionar Reporte Arbol.

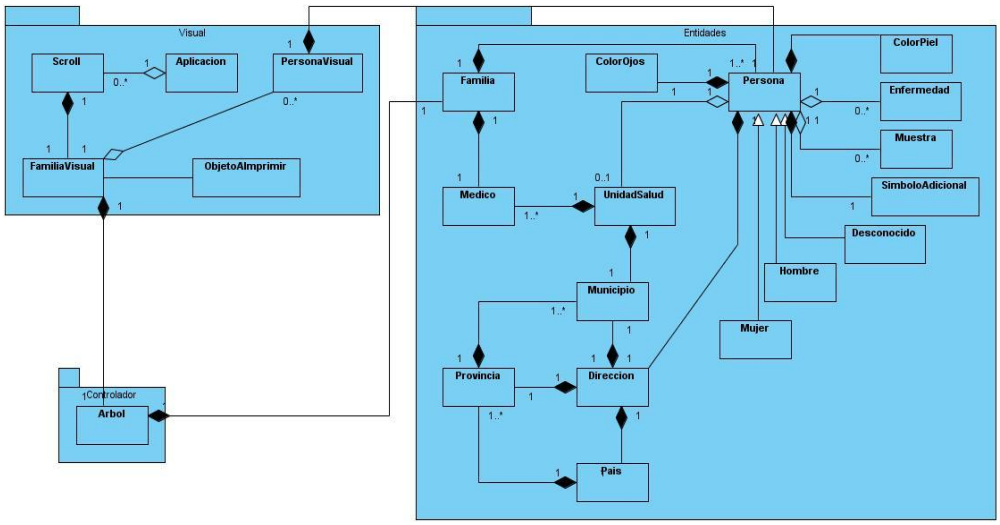


Fig.1.3 DC_CU_Imprimir Arbol.

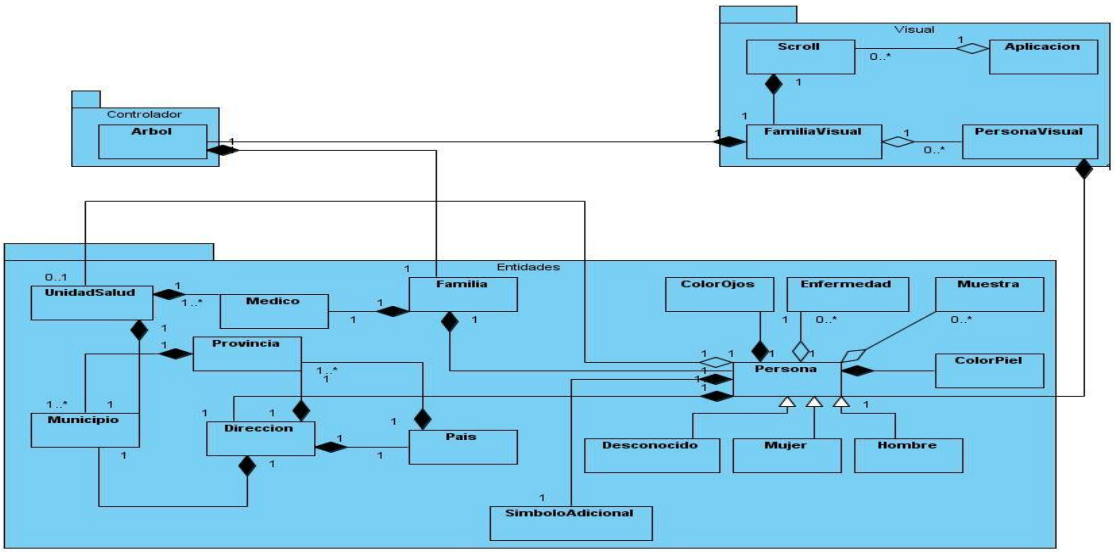


Fig.1.4 DC_CU_Ordenar Arbol.

Anexo 2. *Diagramas de clase de la aplicación web*

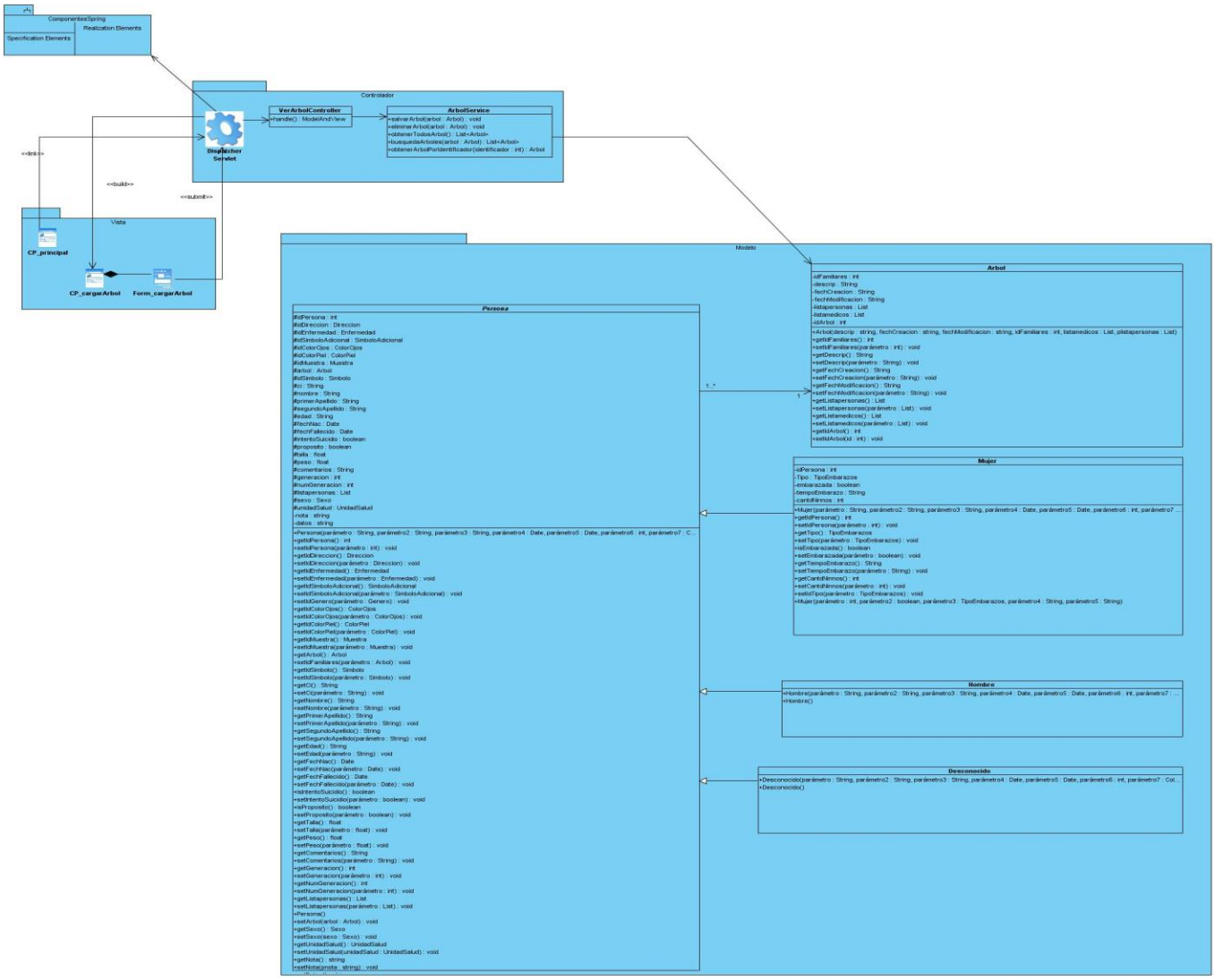


Fig.2.1 DCD_CU_CargarÁrbolGenealógico.

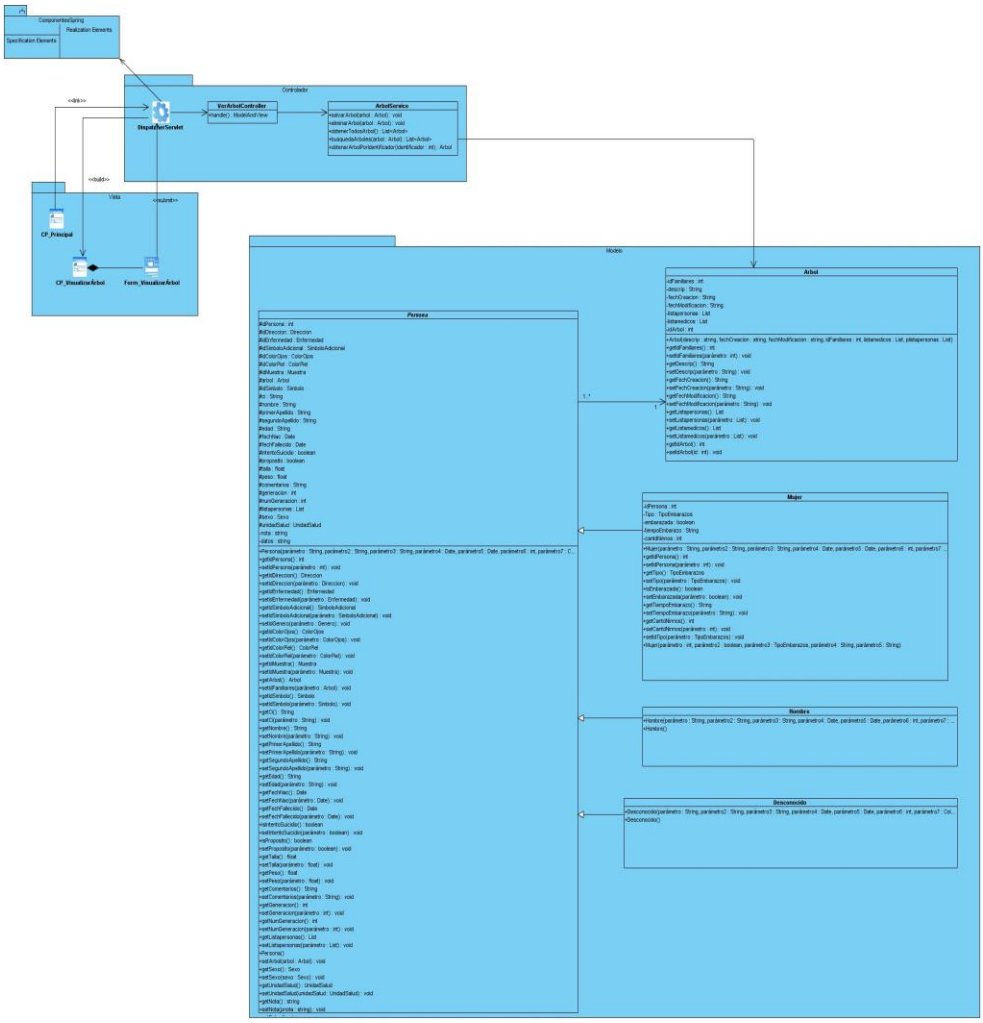


Fig.2.3 DCD_CU_VisualizarÁrbolGenealógico.

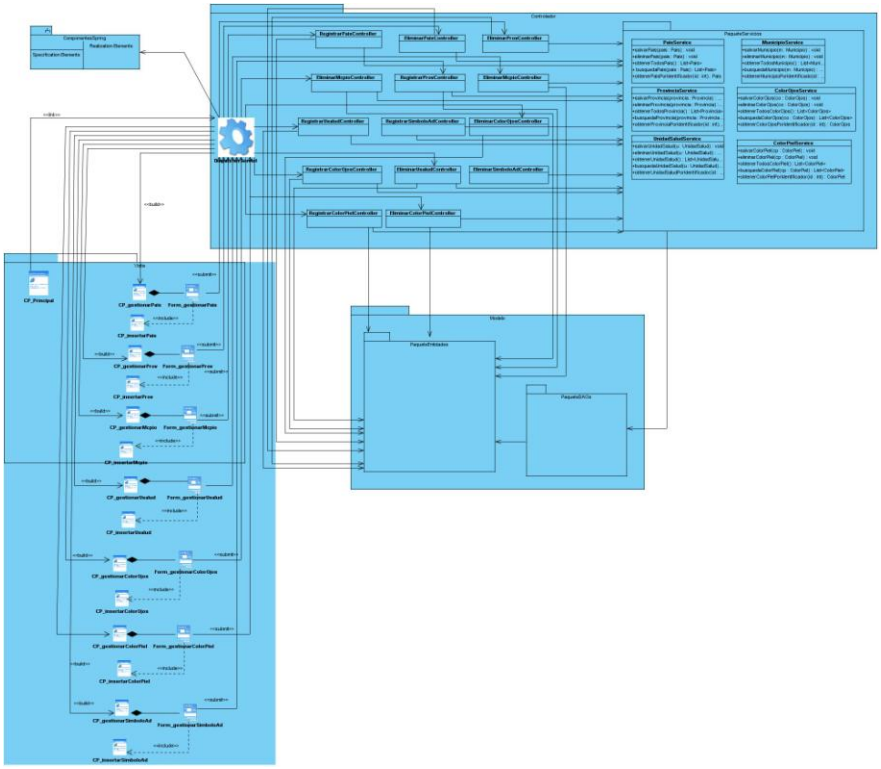


Fig.2.2 DCD_CU_GestionarNomencladores.

Anexo 3

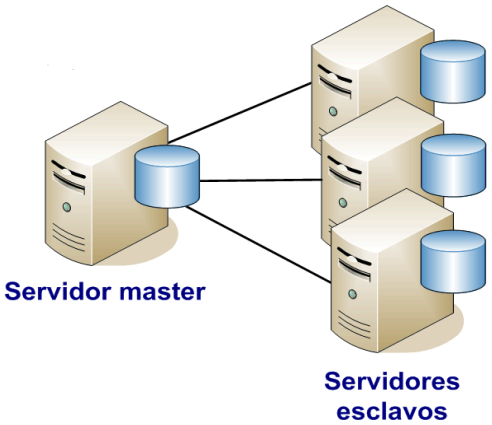


Fig.3.1 Clúster para el balanceo de carga.