Universidad de las Ciencias Informáticas

Facultad 7

Título: Diseño del modelo de datos del Sistema de Colaboración Médica v 2.0

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO

DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Yoelvis Pozo Alfonso

Tutor: Ing. Héctor Manuel Solís Mulet

Ciudad de La Habana, julio de 2010

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Yoelvis Pozo Alfonso	Ing. Héctor Manuel Solís Mulet

Datos de Contacto

Ing. Héctor Manuel Solís Mulet: Graduado con Título de Oro de Ingeniero en Ciencias Informáticas en la UCI en el año 2008. Posee Categoría Docente de Profesor Adiestrado. Ha participado en proyectos de desarrollo de Sistemas Informáticos para la Salud desde el año 2005. Imparte la asignatura de Gestión de Software y Sistemas de bases de datos. Actualmente se desempeña como Jefe de Colectivo de las asignaturas que imparte en la Facultad 7 y es miembro del grupo de bases de datos del Departamento de Sistemas de Apoyo a la Salud (SAS) del Centro Especializado en Soluciones Informáticas Médicas (CESIM).

A mis abuelos por confiar en mí y darme siempre su ejemplo, por haberme ayudado a lograr la gran meta de mi vida, sin ellos esto sería imposible.

A mis padres por darme su ejemplo y ser fuente de inspiración en mi vida.

Agradezco a Fidel Castro y la Revolución cubana, sin los cuales esta carrera y este sueño que hoy vivo no hubieran sido más que una utopía.

Agradezco a las personas que han contribuido a la elaboración del presente trabajo, a mi tutor Héctor, a los compañeros del tribunal y al oponente por su crítica constructiva.

A mis amigos, pues aunque todos no estén presentes en este momento, me han acompañado en los buenos y malos momentos de la vida, sorteando dificultades y aprendiendo a ser mejor persona, especialmente a Pichardo, Enrique, Daynier, Luis Angel, Samuel, Marena, Isandra, Lázaro y el resto de mis compañeros que los recordaré siempre.

A todos mis vecinos por aguantarme en el barrio y por brindarme todo su apoyo y darme los mejores consejos desde allá, en especial a María, Caridad, Fresolina, Aymé, Emilia, Odalys, Briseida, Reinier, Juana, Rosita, los jimaguas, Norma, en fin a todos, discúlpenme si se me quedó algunos.

Agradecer a todos mis profesores que ayudaron en mi preparación como profesional, en especial a Hugo Vargas.

A mis abuelos, por cumplir su deseo más añorado.

A mi madre por ser una batalladora incansable.

A toda mi familia por contar con ustedes en todo momento.

A todos mis amigos que se preocuparon y me ayudaron a la realización de este trabajo.

Resumen

Entre las premisas de la política del gobierno cubano están la solidaridad e internacionalismo, que se ven reflejados en la colaboración médica cubana. La Unidad Central de Cooperación Médica (UCCM) es la entidad encargada de procesar toda la información relacionada con esta actividad, apoyada del Sistema de Colaboración Médica v1.0 (SCM v1.0) que es un software que gestiona la información referente a los colaboradores cubanos. En aras de aumentar las prestaciones del SCM v1.0 se decidió crear una segunda versión del mismo surgiendo de este modo la necesidad de crear una base de datos para guardar todo el volumen de información gestionado por este sistema.

El objetivo del presente trabajo es diseñar una base de datos para almacenar la información generada en el proceso de gestión del Sistema de Colaboración Médica v.2.0. Para su realización se utilizó como gestor de base de datos PostgreSQL 8.3, como herramienta de modelado el Enterprise Architect versión 7.1 y para la implementación del modelo de datos el Mapeo de Objeto Relacional (ORM) Doctrine del framework Symfony. La siguiente investigación compensará las necesidades de la UCCM en cuanto a almacenamiento de la información, seguridad y portabilidad del diseño propuesto.

Palabras Claves: información, base de datos.

Índice

Introducción

Los sistemas de información existen desde las primeras civilizaciones. Los datos se recopilaban, se estructuraban, se centralizaban y se almacenaban convenientemente. El objetivo inmediato de este proceso era poder recuperar estos mismos datos u otros datos derivados de ellos en cualquier momento, sin necesidad de volverlos a recopilar, este paso solía ser el más costoso o incluso irrepetible. Desde la antigüedad ha sido necesario almacenar grandes cantidades de datos con el objetivo de tomar decisiones y realizar acciones más pertinentes que las que se realizarían sin dicha información, es por ello que surge lo que se conoce como base de datos (BD). [1]

En la actualidad, con el auge de las tecnologías en el sector de la informática aparecen nuevos dispositivos electromagnéticos que ofrecen mayores posibilidades de almacenamiento y economizan un tiempo considerable en la búsqueda y tratamientos de datos. Al mismo tiempo, se han enriquecido las bases de datos mediante el uso de diferentes metodologías de diseño y herramientas como los Sistemas Gestores de Bases de Datos (SGBD) permitiendo el desarrollo de aplicaciones que facilitan la comunicación y la gestión de grandes volúmenes de datos.

La Universidad de las Ciencias Informáticas (UCI) como proyecto de la Revolución Cubana, denominada en sus inicios "Proyecto Futuro", es el buque insignia de la muy joven Industria Cubana del Software y su tarea primordial es la producción de software que mejoren las prestaciones y servicios de las instituciones y empresas en todas las esferas de la nación. Hasta el momento varios de sus objetivos fundamentales como el de informatizar muchos procesos que se llevan a cabo dentro y fuera de la Universidad han sido logrado. Un ejemplo de esto es el sector de la salud donde se han informatizado todas las áreas de trabajo con el objetivo de mejorar las prestaciones hospitalarias de los centros asistenciales de salud.

En los últimos años, el sistema de salud cubano se ha caracterizado por una amplia experiencia en su labor internacional, que se inició en 1962 con el envío de la primera misión médica cubana a Argelia y desde ese momento se ha colaborado con más de 101 países de Asia, África y América Latina. Debido a la importancia que tiene para Cuba prestar ayuda solidaria a países hermanos, se crea la Unidad Central de Cooperación Médica (UCCM), con el objetivo de mantener toda la información posible de la persona que esté realizando la misión, asegurada y archivada, la unidad sería la encargada de todo lo relacionado con el colaborador cubano dentro y fuera del territorio nacional. En sus inicios todo el proceso se realizaba

de forma manual, dando lugar a pérdidas irreparables de información en ocasiones, además de horas engorrosas de trabajo y papeleo.

La UCCM tuvo la idea de crear un sistema que permitiera el control de la colaboración médica cubana, tratando de evitar que la información se manejara manualmente, con el objetivo de obtener mayor seguridad y agilizar los procesos de gestión de la información manejada por esta institución. La UCI fue la escogida para realizar este sistema y cuenta hoy con una primera versión del software nombrado, Colaboración Médica.

Este sistema es ampliamente usado ya que permite la gestión de la información de todo el personal médico y no médico en misiones internacionalistas, facilitando así un mejor control del recurso humano. Además, tiene un alcance de tres niveles (municipio, provincia y nación) y cada nivel podrá acceder a la información que en este se maneja garantizando la integridad, manejabilidad y confiabilidad de los datos mediante la integración a los registros disponibles en el Registro Informatizado de la Salud (RIS). De manera adicional Colaboración Médica cuenta con un módulo para gestionar el pago a los cooperantes y a sus designados.

La BD empleada en Colaboración Médica v1.0 presenta algunas limitaciones, algunas de ellas son las siguientes:

1. Posee gran parte de la lógica del negocio dentro de la BD (representada a través de funciones, procedimientos almacenados) impidiendo la portabilidad y escalabilidad del sistema a otros SGBD.
2. No permite almacenar información referente a la trazabilidad de las acciones de los usuarios en el sistema, lo cual constituye un serio problema de seguridad debido a que es imposible conocer los accesos y peticiones realizados por los mismos y de esta manera imposibilita la realización de auditoría al sistema.
3. La BD no permite almacenar toda la información necesaria para la gestión de la misma.

El equipo de desarrolladores del sistema de Colaboración Médica decidió realizar una segunda versión del mismo que se ajuste en un mayor grado a los requerimientos de la UCCM en cuanto al almacenamiento y la gestión de la de la información. En este sentido, se pretende utilizar nuevas tecnologías como el framework Symfony, lo que representaría un cambio sustancial en la manera en que se almacenan los

datos en la BD de la versión precedente. Por tanto, se necesitan realizar algunos cambios en el diseño la BD para que se adapte mejor al modelo de persistencia que propone este framework para el desarrollo de software.

Luego de un análisis de la problemática anterior y con el fin de solucionar estas necesidades se propone como **problema a resolver:** ¿Cómo lograr un diseño de BD óptimo para almacenar todos los datos generados en el proceso de gestión de la información del sistema de Colaboración Médica v.2.0?

Para este problema se define como **objeto de estudio:** El proceso de gestión de la información del personal de la UCCM, y como **campo de acción**: La información general del personal de la UCCM que cumple misiones médicas.

Se define como **objetivo general** de la investigación:

- Diseñar una BD para almacenar los datos generados en el proceso de gestión de la información del Sistema de Colaboración Médica v.2.0.

Para dar cumplimiento al objetivo anteriormente expresado, se deben cumplir las siguientes **tareas científicas:**

1. Asimilar los conceptos relacionados con el diseño de bases de datos relacional, para una mejor profundización de los elementos que integran una BD relacional.
2. Asimilar la arquitectura propuesta por el CESIM y el Departamento SAS para el trabajo con Sistemas de bases de datos para facilitar el estudio sobre las principales herramientas y metodologías a utilizar.
3. Analizar los procesos de negocio y la BD empleada por el Sistema de Colaboración Médica v1.0, para obtener una mayor comprensión de la lógica del negocio utilizada en la BD del Sistema de Colaboración Médica v.1.0.
4. Asimilar el modelo del framework symfony, de forma específica el trabajo con el ORM Doctrine, con el objetivo de analizar las pautas que propone este modelo para la implementación del diseño de la BD.

5. Realizar el diseño de la BD del Sistema de Colaboración Médica v.2.0.
6. Implementar sobre el modelo de datos propuesto una política de trazabilidad en las acciones que llevan a cabo los usuarios en la base de datos.
7. Validar teóricamente el modelo de datos propuesto.
8. Realizar pruebas de rendimiento sobre la BD propuesta.

En sentido general se puede destacar que el desarrollo de la nueva versión de la BD para el sistema Colaboración Médica v2.0 proporcionará un grupo de beneficios entre ellos se pueden mencionar los siguientes:

1. Aumento de la portabilidad del diseño de la BD mediante la eliminación de la lógica del negocio dentro de la implementación de la misma.
2. Permite almacenar información referente a la trazabilidad de las acciones de los usuarios en el sistema, permitiendo un eficiente análisis de la auditoría del mismo.
3. Corrige la limitación de la versión precedente en cuanto a la gestión de la información relacionada en la BD.
4. Optimización del diseño de la BD, mediante la eliminación de la redundancia de información.

La investigación consta de 3 capítulos, en los cuales se desarrollan aspectos de importancia para lograr el objetivo que se persigue.

Capítulo #1: Fundamentación teórica

1. Estudio de estado del arte del tema bases de datos en el mundo, nacional y de la Universidad.
2. Definir herramientas y metodologías que se utilizarán en la propuesta de solución.

Capítulo #2: Análisis y descripción de la solución propuesta

1. Estrategia de integración con los módulos que componen el sistema.

2. Arquitectura de la propuesta de solución.

3. Requisitos, tanto funcionales como no funcionales, que se tienen en cuenta en el desarrollo de la propuesta de solución.

4. Diagramas que muestran el diseño de la solución, además de la descripción de cada una de las tablas que los componen.

Capítulo #3: Validación del diseño realizado

1. Se valida el diseño realizado de forma teórica teniendo en cuenta aspectos como la integridad, la normalización, la seguridad, entre otros.

2. Se valida funcionalmente el diseño realizado mediante la aplicación de pruebas de carga y estrés al mismo.

Capítulo1: Fundamentación Teórica

En el presente capítulo se brinda información acerca del estudio realizado sobre el origen y desarrollo de las BD, técnicas empleadas para su diseño y modelación, las herramientas, SGBD, además de valorar cuál es el modelo, diseño, herramientas y gestores más acordes a utilizar en el desarrollo del presente trabajo de diploma.

1.1 Base de Datos

La BD es un componente fundamental de un sistema de información. El escenario actual de la tecnología de bases de datos es resultado del perfeccionamiento que ha tenido lugar en el procesamiento de datos y en la gestión de la información, por lo cual se hace necesario que sus características fundamentales queden recogidos en esta primera parte investigativa.

1.1.1 Tipología de Base de Datos

Las bases de datos pueden ser clasificadas desde diferentes puntos de vista teniendo en cuenta diversos criterios de clasificación:

- Teniendo en cuenta la variabilidad de los datos almacenados:

BD estáticas: Son BD de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones. Son aquellas BD en las que no se puede modificar la información que está guardada, es decir, solo se puede consultar. [2]

BD dinámicas: en estas bases de datos la información almacenada se modifica con el tiempo, por lo que en ellas están presentes todas las operaciones de gestión (agregar, modificar-actualizar, y eliminar) por lo que la información guardada en ellas sufre cambios constantemente. [3]

- De acuerdo con el modelo de administración de datos:

Modelo orientado a objetos: Conjuga de forma centralizada los conceptos de abstracción, jerarquía, modularidad, persistencia, tipos y concurrencia. Un modelo orientado a objetos tiene obligatoriamente que cumplir con lo siguiente: la estructura básica de trabajo son los objetos, no algoritmos; donde cada objeto no es más que una instancia de una clase ya definida y que dichas clases estarán relacionadas únicamente por relaciones de herencia.

Modelo relacional: organiza la información en forma de tablas bidimensionales y de tal forma es que es percibida por los usuarios. Para interactuar con la información en dichas tablas este modelo tiene definido un grupo de operadores, y para construir las consultas a bases de datos relacionales el lenguaje más utilizado es Structured Query Language o Lenguaje Estructurado de Consultas (SQL), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante el diseño de estas BD se pasa por un proceso conocido como normalización, con el fin de eliminar redundancias y evitar inconsistencias en las tablas.

Modelo jerárquico:

Este modelo utiliza árboles para la representación lógica de los datos. Este árbol está compuesto de unos elementos llamados nodos. El nivel más alto del árbol se denomina raíz. Cada nodo representa un registro con sus correspondientes campos [4]

Modelo de red:

En este modelo las entidades se representan como nodos y sus relaciones son las líneas que los unen. En esta estructura cualquier componente puede relacionarse con cualquier otro. [5]

Los conceptos básicos en el modelo en red son:

1. Un hijo puede tener varios padres.
2. El tipo de registro, que representa un nodo.
3. Elemento, que es un campo de datos.
4. Agregado de datos, que define un conjunto de datos con nombre.

1.1.1.1 Elección para la solución en propuesta:

Una BD para un sistema de colaboración médica sufrirá cambios constantemente, por lo que la misma será de tipo dinámica. En cuanto a la tipología, para la solución propuesta la mejor opción la representa

una BD de modelo relacional, ya que otras tipologías como la orientada a objetos imponen restricciones como la estructura básica (los objetos) y la relación entre estos (sólo por herencia). Además, tipologías como la jerárquica o de modelo de red son de muy complejo entendimiento y sus mecanismos de reducción de redundancia son deficientes. Por otra parte, el modelo relacional brinda las siguientes ventajas:

1. Fácil comprensión: al ignorar el almacenamiento físico de los datos, se centra en el modelo lógico de la BD, por lo que su interpretación humana resulta más sencilla.

2. Garantiza la integridad referencial: al eliminar un registro elimina todos los registros relacionados dependientes.

3. Posee mecanismos de reducción de redundancia de datos: estos mecanismos permiten optimizar la BD y evitar inconsistencias o duplicidades de datos.

1.2 Tendencias y Tecnologías Actuales

Para lograr un buen diseño y modelado de una BD se necesita de una serie de herramientas que permitan la gestión de los procesos inherentes, control sobre la redundancia de los datos, así como su consistencia, comparación de datos por usuarios autorizados, la integridad de los datos, la seguridad con protección frente a usuarios no autorizados, la accesibilidad a los datos a través de SGBD, mantenimiento e independencia de los datos, concurrencia, servicios de copias de seguridad y de recuperación ante fallos, así como, lograr que todo el proceso de creación y modelado sea lo más sencillo y rápido posible. A continuación se exponen características de algunos SGBD y herramientas que facilitan el proceso de diseño y modelado de bases de datos.

1.3 Fundamentación de un Sistema Gestor de Base de Datos

Cuando se trabaja con la gestión de la información se hace necesario trabajar de la manera más sencilla y organizada posible. Es ahí donde surgen SGBD para almacenar de forma segura y organizada la información procesada en cualquier organización. Estos sistemas están destinados a ofrecer mecanismos de gestión de BD con el fin de acelerar el trabajo, aligerarlo y ofrecer mecanismos de automatización de los procesos.

PostgreSQL es un sistema gestor de bases de datos objeto-relacional (ORDBMS por sus siglas en inglés) basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker.

PostgreSQL es una derivación libre (Open Source) de este proyecto. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL presenta características la herencia, diferentes tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

A continuación se enumeran las principales características de este gestor de bases de datos:

1. Implementación del estándar SQL92/SQL99.

2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo: fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.

3. Incorpora una estructura de datos array.

4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.

5. Permite la declaración de funciones propias, así como la definición de disparadores.

6. Soporta el uso de índices, reglas y vistas.

7. 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.

8. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

9. Presenta avanzadas funcionalidades como son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arrays.

10. Presenta otras características denominadas ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad):

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

- **Consistencia** es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la BD.
- **Aislamiento** es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- **Durabilidad** es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. [6]

Además PostgreSQL presenta:

Altamente Extensible. PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.

Soporte SQL Comprensivo. PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial. PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la BD.

API Flexible. La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, .NET y Pike.

Lenguajes Procedurales. PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

MVCC. MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún DBMS con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en que las lecturas tienen que esperar para acceder a información de la BD. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros. Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo entre fila porque un

lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles. [7]

Write Ahead Logging (WAL). La característica de PostgreSQL conocida como Write Ahead Logging (escritura por delante del registro), incrementa la dependencia de la base de datos al registro de cambios antes que estos sean escritos en la base de datos, garantizando que en un momento dado, exista un conflicto de conexión u otro problema, existirá un registro de las transacciones a partir del cual se podrá restaurar la base de datos al resolverse la contrariedad existente. [8]

Algunos límites de PostgreSQL se incluyen a continuación**:** [9]

1. Máximo de base de datos: ILIMITADO
2. Máximo de tamaño de tabla: 32TB
3. Máximo de tamaño de registro: 1.6TB
4. Máximo de tamaño de campo: 1GB
5. Máximo de registros por Tabla: ILIMITADO
6. Máximo de campos por tabla: 250 a 1600 (depende de los tipos usados)
7. Máximo de índices por tabla: ILIMITADO
8. Número de lenguajes en los que se puede programar funciones: aproximadamente 10 (pl/pgsql, pl/java, pl/perl, pl/python, tcl, pl/php, C, C++, Ruby, etc.)
9. Métodos de almacenamiento de índices: 4 (B-tree, Rtree, Hash y GisT)

1.4 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

Definición de UML: (UML por sus siglas en inglés *Unified Modeling Language*) es un lenguaje estándar diseñado para especificar, visualizar, construir y documentar software orientado a objetos. No es un proceso, método o metodología, es simplemente un lenguaje de modelado. [10].

Características principales:

1. Es un lenguaje de representación visual.
2. Permite combinar diversos elementos gráficos y crear diagramas.

3. Se usa sólo para modelar sistemas con tecnología orientada a objetos. El UML describe lo que hará un sistema pero no dice cómo implementarlo.

Tipología de los diagramas UML:

1. Diagramas de estructura estática: Describen las propiedades estructurales del sistema. Por ejemplo: los diagramas de Casos de Uso, de clases, y de objetos.
2. Diagramas de comportamiento: estos son los diagramas de estado, los de interacción (que pueden ser de secuencia o de colaboración) y los diagramas de actividades.
3. Diagramas de implementación: diagramas de componentes y despliegue

Relaciones presentes en los diagramas UML:

1. Dependencia: esta es una relación entre dos elementos (uno dependiente y uno independiente), mediante la cual se expresa que un cambio en el elemento independiente afectará al elemento dependiente de este.
2. Asociación: describe conexiones entre objetos. Un tipo de asociación muy común es la agregación.

3. Generalización especialización: describe relaciones entre objetos especializados (hijos) que derivan de un objeto más general (padre), lo cual sirve para heredar comportamientos y estructuras con el fin de poder sustituirse entre sí o especializarse en funciones más específicas [11]

1.5 El Proceso Unificado de Desarrollo de Software (RUP) como metodología para el desarrollo de la solución propuesta

El Proceso Unificado de Desarrollo (RUP) es la metodología de desarrollo más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es una metodología robusta que se combina con UML para la generación del producto final: artefactos, modelos, documentos anexos, ficheros ejecutables, o cualquier otro tipo de fichero que conforme el programa. RUP define quién (trabajadores) debe hacer qué (artefactos) cómo (actividades) y cuándo (flujo de actividades).

Características de RUP:

1. Iterativo e incremental.
2. Centrado en la arquitectura.
3. Guiado por casos de uso.

Fases de RUP:

RUP posee cuatro fases fundamentales: Inicio (o concepción), Elaboración, Construcción, y Transición, las cuales están divididas en diversos flujos de trabajo.

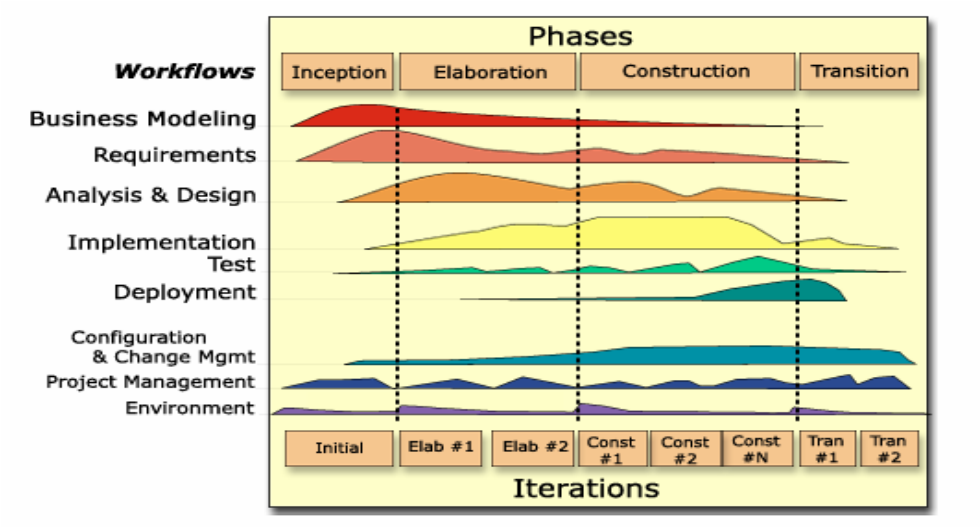


Fig. 1: RUP en 2 dimensiones

Para la mayoría de proyectos de desarrollo de aplicaciones, la tecnología utilizada para los datos persistentes es una base de datos relacional. El diseñador de base de datos es responsable de definir el diseño detallado de la base de datos, incluyendo tablas, índices, vistas, restricciones, desencadenantes, procedimientos almacenados y otras construcciones específicas de la base de datos necesarias para almacenar, recuperar y suprimir objetos persistentes. Esta información se mantiene en el Producto de trabajo: Modelo de datos.

El ámbito de las tareas efectuadas por el rol diseñador de base de datos varía dependiendo del tamaño y la complejidad del esfuerzo de desarrollo de la aplicación y del tipo de mecanismos de almacenamiento de datos persistentes utilizados para el proyecto.

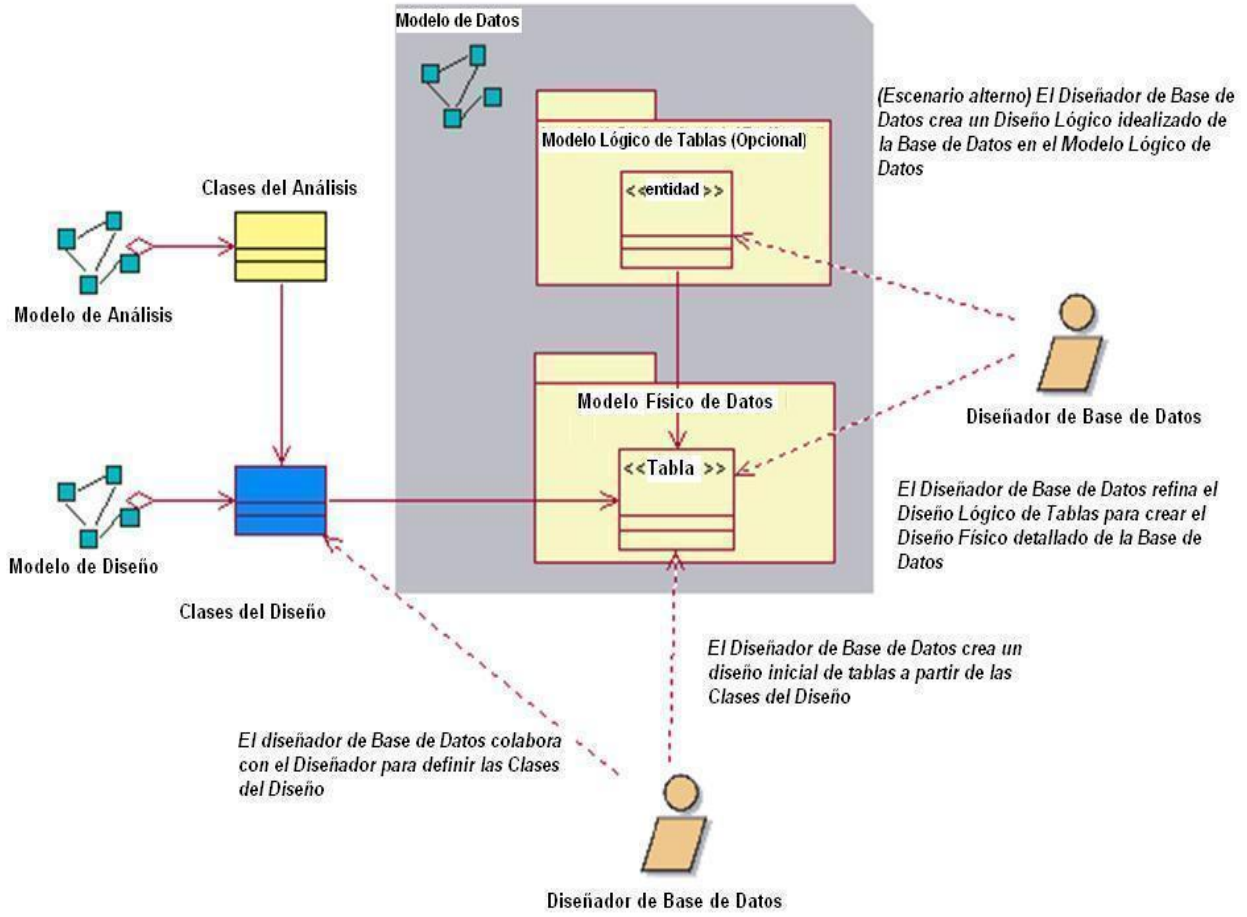


Fig. 2. Actividades del Rol de Diseñador de Base de Datos según RUP

1.6 Herramienta para el Modelado

Las Herramientas CASE permiten aumentar la productividad en el desarrollo de software. Reduciendo el coste de las mismas en términos de tiempo y dinero pues mejora la comunicación entre los desarrolladores del proyecto. Además, facilita un mejor intercambio de ideas con el cliente y entre los propios integrantes del equipo de desarrollo, permitiendo modelar los procesos de negocio de las empresas. Aunque no son suficientes si no van acompañadas de las técnicas y metodologías adecuadas,

si no se implantan de la forma correcta o si no se complementan por otros aspectos relativos a la calidad. El entorno CASE debe estar integrado con otros aspectos que también persiguen la mejora del software como son las métricas, el modelo de proceso, evaluación de capacidad de desarrollo y calidad.

1.6.1 Herramientas CASE para UML

Enterprise Architect es una herramienta de uso muy sencillo, que aborda el diseño y análisis UML y cubre el desarrollo de software desde la captura de requerimientos a lo largo de las etapas de análisis, diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, Windows, diseñada para ayudar a construir software robusto y fácil de mantener.

Además, permite generar documentación e informes flexibles y de alta calidad. Proporciona trazabilidad completa desde el análisis de requerimientos y los artefactos de diseño, hasta la implementación y el despliegue. En combinación con la asignación de recursos y tareas que incorpora, los equipos de Gestión de Proyectos y Calidad están dotados con toda la información necesaria para ayudarles a controlar los proyectos y sus entregas. Las bases de Enterprise Architect están sustentadas en la especificación de UML. Además usa Perfiles UML para extender el dominio de modelado, mientras que la validación del modelo asegura la integridad del proyecto. Combina los procesos de negocio, información y flujos de trabajo en un modelo usando las extensiones gratuitas para BPMN y Eriksson-Penker. [12]

1.7 Herramienta de desarrollo

1.7.1Herramienta pg. Admin.

Es una de las más utilizadas plataformas de desarrollo para PostgreSQL. Está diseñada para responder a todo tipo de necesidades, tanto de consultas SQL simples hasta la elaboración de BD complejas. La interfaz gráfica soporta todas las características de PostgreSQL y es de fácil administración. La aplicación también posee un editor de sintaxis SQL, y la conexión al servidor puede hacerse a través del protocolo TCP-IP o Unix Domain Sockets sin la necesidad de drivers especiales para la conexión con la BD. Permite crear y visualizar diagramas de entidad relación. Se puede combinar con la herramienta Visual Paradigm para utilizarlas una en apoyo a la otra. Permite crear condiciones que deben prevalecer para mantener la integridad relacional, establecer y modificar relaciones, crear y generar el código que construye a la BD en el servidor, así como la ingeniería inversa. Entre sus principales características se encuentran:

1. Multiplataforma

2. Diseñado para múltiples versiones de PostgreSQL.
3. Interfaz multilingüe para más de 25 traducciones.
4. El acceso a todos los objetos de PostgreSQL [13]

1.7.2 Apache JMeter

JMeter una herramienta Java dentro del proyecto de Jakarta que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web y bases de datos. El proyecto Jakarta es el que se encarga de crear y mantener todas las soluciones Open Source (código abierto) creadas para la plataforma Java. Se destaca por su versatilidad, estabilidad, y por ser de uso gratuito. JMeter permite realizar pruebas webs clásicas, pero también permite realizar test de FTP, JDBC, Web Service (en Beta) entre otras. También permite la ejecución de pruebas distribuidas entre distintos ordenadores para realizar pruebas de rendimiento además de mostrar los resultados de las pruebas en una amplia variedad de informes y gráficas. [14]

1.7.3 DBDesigner Fork

DB Designer Fork es un programa de diseño visual de bases de datos que integra el diseño entidad-relación y la creación de bases de datos. Una vez creada tu base de datos, podrás crear los scripts correspondientes para diferentes gestores, como Firebird/InterBase, Microsoft SQL Server, MySQL, Oracle o PostgreSQL. [15]

1.8 Tecnologías del Software.

1.8.1Framework Symfony

Diseñado con el objetivo de optimizar la creación de las aplicaciones web con el uso de sus características. Posee una librería de clases que permiten reducir el tiempo de desarrollo. Está desarrollado en PHP5, se puede utilizar en plataformas Unix, Linux y Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón MVC, soporta AJAX, plantillas y un gran número de bases de datos. Symfony es un enorme conjunto de herramientas y utilidades que minimizará el desarrollo de la aplicación a desarrollar, ya que es una de las mejores copias para PHP del famoso framework Ruby on Rails. También ha tomado las mejores ideas de Rails y de muchos otros

frameworks, incorporando ideas propias y el resultado es un framework elegante, estable, productivo y muy bien documentado.

Entre las múltiples ventajas técnicas de Symfony se pueden citar:

- Abstracción de bases de datos vía Mapeo Relacional de Objetos **(**ORM**):** las tablas de la BD están disponibles como objetos en el código. La capa ORM está basada en Propel o Doctrine**.**
- Un parseador YML **(**YAML**)** propio, de forma que los ficheros de configuración y la descripción del modelo de datos pueden ser descritos de forma sencilla y rápida (a diferencia de los ficheros XML, con una gran cantidad de tags de apertura y cierre).
- Documentación de gran calidad, así como una amplia y activa comunidad de desarrolladores.
- Symfony genera código orientado a objetos para las funcionalidades más comunes del manejo de bases de datos.
- Genera interfaces CRUD (Crear Leer Actualizar Eliminar) para las tablas de la BD.

1.8.2 ORM.

Un ORM es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una BD relacional, es decir, las tablas de la BD pasan a ser clases y los registros, objetos que se pueden manejar con facilidad. Utilizar un ORM tiene una serie de ventajas que facilitan enormemente tareas comunes y de mantenimiento:

1. **Reutilización:** La principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.
2. **Encapsulación:** La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.
3. **Portabilidad:** Utilizar una capa de abstracción permite cambiar en mitad de un proyecto de una BD MySQL a cualquier SGBD sin ningún tipo de complicación. Esto es debido a que no utilizan una sintaxis específica de un SGBD para acceder al modelo, sino una sintaxis propia el ORM utilizado que es capaz de traducir a diferentes tipos de bases de datos.
4. **Seguridad:** Los ORM suelen implementar mecanismos de seguridad que protegen la aplicación de los ataques más comunes como inyecciones SQL.

5. **Mantenimiento del código:** Gracias a la correcta ordenación de la capa de datos, modificar y mantener el código es una tarea sencilla. [16]

Doctrine es un tipo de ORM para PHP 5.2.3 y posterior que se sitúa encima de una potente capa de abstracción de bases de datos Uno de sus principales características es la opción de escribir las consultas de BD en un lenguaje personalizado de SQL llamada Doctrine Query Language (DQL), inspirado en Hibernate HQL**.** Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria.

1.8.2.1 Lenguaje DQL

DQL es un lenguaje de consulta de objetos creado para ayudar a los usuarios en la recuperación de objetos complejos. En comparación con el uso de SQL primas, DQL tiene varios beneficios. [17]

1. Desde el principio ha sido diseñado para recuperar los registros (los objetos) no un conjunto de resultado en las filas.
2. DQL es portátil en diferentes bases de datos.
3. DQL tiene algunos muy complejos e integrados algoritmos, como (el algoritmo de límite de registros) que puede ayudar al desarrollador a recuperar objetos de manera eficiente.
4. Es compatible con algunas funciones que pueden ahorrar tiempo cuando se trata de uno-a-muchos, muchos-a-muchos de datos relacionados con condiciones de búsqueda.

En este primer capítulo se ha visto el estado del arte de las tecnologías relacionadas con el acceso y almacenamiento de información existente en el mundo, sus principales características, así como sus ventajas y desventajas. El estudio de las principales herramientas y metodologías ha proporcionado elementos que justifican la elección de determinadas técnicas empleadas (diseño e implementación) en el desarrollo del modelo de datos del Sistema de Colaboración Médica v.2.0.

Capítulo 2: Descripción y análisis de la solución propuesta

El capítulo siguiente describe, la estrategia de integración con otras herramientas de desarrollo, en este caso con el framework Symfony, la descripción de la arquitectura a utilizar, una breve descripción de cada tabla y los valores que almacenan las mismas, así como los modelos correspondientes a la BD.

2.1 Integración con el Framework Symfony

La información mínima que necesita Symfony para realizar peticiones a la BD es su nombre, los datos de acceso y el tipo de BD. Esta información se indica en el archivo databases.yml que se encuentra en el directorio config.

Las opciones de la conexión se establecen para cada entorno. Se pueden definir diferentes opciones para los entornos prod (entorno de producción), dev (entorno de desarrollo) y test (ambiente de prueba), o para cualquier otro entorno definido en la aplicación. Para el caso de la conexión con el PostgreSQL el prefijo que se debe utilizar es pgsql, para modificar el fichero.yml y se debe verificar que las librerías pdo.so y pdo_pgsql.so (pertenecientes al sistema operativo Linux) estén activas en el servidor web que se esté utilizando. También es posible redefinir esta configuración en cada aplicación, estableciendo diferentes valores para las opciones en un archivo específico de la aplicación, de esta forma, se pueden disponer de políticas de seguridad diferentes para las aplicaciones públicas y las aplicaciones de administración del proyecto, y definir distintos usuarios de bases de datos con privilegios diferentes.

Para que este framework pueda interactuar con la BD manejada por el SGBD PostgreSQL es necesario realizar una serie de cambios en algunos de los archivos de configuración que ofrece el gestor.

Para lograr la conexión a la BD con el SGBD PostgreSQL, se configuran los archivos postgresql.conf y el pg_hba.conf. En el primero, se busca la configuración llamada --Connection Settings-- y dentro de ella listen_addresses, ahí se debe de especificar si se desea que la conexión sea solo del propio equipo (localhost) o que otros equipos se puedan conectar (*), además del puerto (port) por el cual se conectarán a la BD (5432 por defecto). En el caso del archivo pg_hba.conf se localiza la configuración llamada --IPv4 local connections—o Ipv6 y en dependencia de la versión de la dirección del ip, se configura las bases de datos, los usuarios, los host a permitir y el método de autenticación a utilizar, por defecto es md5, por

ejemplo, la configuración: Host all all 10.0.0.1/8 md5 indica que se podrán conectar todos los host desde cualquier trama de IP contenida en esa dirección IP a todas las bases de datos que existen en el servidor utilizando cualquier cuenta de usuario del gestor a través del método de autenticación md5.

2.2 Descripción de la arquitectura

El sistema Colaboración Médica en su versión 2.0 utiliza como lenguaje de programación PHP 5.2.6 y como framework Symfony, este es uno de los frameworks más utilizados para PHP y utiliza el estilo arquitectónico MVC (Modelo-Vista-Controlador) como base de su funcionamiento, es fácilmente integrable a las aplicaciones debido a su composición y a que contiene diferentes clases de gran utilidad.

Será utilizado además, Doctrine que es un potente y completo ORM para PHP 5.2 o posterior. Su principal ventaja radica en poder acceder a la BD utilizando la programación orientada a objetos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente. Es fácilmente integrado a los principales frameworks de desarrollo utilizados actualmente dentro de los que se encuentra el Symfony. Permite exportar una BD existente a sus clases correspondientes y también a la inversa, es decir, convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una BD. El SGBD elegido en el área de BD para Colaboración Médica es el PostgreSQL.

2.3 Requisitos No Funcionales

Usabilidad

1. Preparar a los Administradores en la gestión de Roles y Permisos.
2. Acceso fácil y rápido a los usuarios.

Rendimiento

1. El sistema debe responder en un tiempo relativamente rápido a las peticiones del usuario (menos de 5 segundos).
2. El sistema necesita un servidor de BD en una PC o servidor con (1 o superior) giga de RAM y 1 disco duro de 250 Gigas.

Requerimientos de Hardware

PC Servidor Web

Software instalado en el Servidor Web.

1. Servidor Web Apache 2.2. X o superior.

Servidor de Bases de datos

Software instalado en el Servidor de BD.

1. Servidor de bases de datos PostgreSQL8.3.
2. Servidor de BD (Intel Pentium IV o superior. D 2x2 cache. 3.00 GHz, Memoria RAM 1 GB, Almacenamiento en discos de 250 GB)

Requisitos de Seguridad

Seguridad del sistema.

1. *Confidencialidad*: la información manejada por el sistema está protegida de acceso no autorizado y divulgación.
2. *Integridad:* la información manejada por el sistema es objeto de cuidadosa protección contra la corrupción y estados inconsistentes. Se incluyen también mecanismos de chequeo de integridad y realización de auditorías por personal calificado de la entidad. –
3. *Disponibilidad:* los usuarios autorizados (autenticados por dominio y según su roll) se les garantizará el acceso a la información, los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

2.4 Modelo de Clases Persistentes

Las clases persistentes tienen como origen las clases entidades mediante las cuales modelan los datos perdurables en el tiempo. Los diagramas de clases persistentes se basan en las entidades de los diagramas de clases del análisis y fundamentalmente en los diagramas de clases del diseño. A continuación se muestra el diagrama general de clases persistentes. (ANEXO1)

2.4.1 Descripción de las clases persistentes

A continuación se ofrece la descripción de las clases persistentes más significativas modeladas en el diagrama anterior. En cada descripción se especifica el nombre, atributos, tipo de cada uno de estos, y descripción tanto de la clase como de los atributos.

Tabla 2.1Descripción de la clase Colaborador

Nombre: Colaborador		
Descripción: registra los datos del colaborador		
Atributos	Tipo	Descripción

id	int	Identificador del colaborador.
ci	varchar	Carnet de Identidad del colaborador.
nombre	varchar	Nombre del colaborador.
apellido1	varchar	Primer apellido del colaborador.
apellido2	varchar	Segundo apellido del colaborador.
sexo	varchar	Sexo.
ministerio_trabajo	varchar	Ministerio en el que reside.
dirección_particular	varchar	Dirección particular.
dirección_particular_provincia	varchar	Dirección particular provincia.
dirección_particular_municipio	varchar	Dirección particular municipio.
raza	varchar	Raza del colaborador.
teléfono	varchar	Teléfono.
correo	varchar	Correo.
foto	blob	Foto.
nombre_madre	varchar	Nombre de la madre.
nombre_padre	varchar	Nombre del padre.
madre_viva	bool	Conocer si la madre está viva.
padre_vivo	bool	Conocer si el padre está vivo.
estado civil	int	Identificador del estado civil del colaborador.
necesidad_casa	bool	Conocer si necesita de alojamiento.
nombre_esposa	int	Iidentificador de la esposa del colaborador.

Tabla2.2 Descripción de la clase Misiones

Nombre: Misiones		
Descripción: registra los datos de las misiones		
Atributos	Tipo	Descripción
id	int	Identificador de la misión.
id_ciudadano_autoriza	int	Identificador del ciudadano que autoriza la misión.
id_ciudadano_avisar_urgencia	int	Identificador del ciudadano para casos de urgencia.
fecha_inicio	timestamp	Fecha en que se inició la misión.
fecha_fin	timestamp	Fecha en que culminó la misión.
tiempo_estancia	int	Tiempo en meses que demora la misión.
teléfono_avisar_urgencia	varchar	Teléfono del ciudadano en casos de urgencia.
descripción_evaluación	varchar	Descripción realizada para la misión.
teléfono_designado_ciudadano	varchar	Teléfono del designado.

Tabla2.3 Descripción de la clase Movimiento

Nombre: Movimiento

Descripción: registra los datos de los movimientos realizado en las misiones		
Atributos	Tipo	Descripción
id_ciudadano_mueve	int	Identificador del ciudadano que está realizando el movimiento.
fecha_inicio	timestamp	Fecha en que se realiza el movimiento.
motivo	varchar	Descripción del motivo del movimiento.
fecha_fin	timestamp	Fecha fin del movimiento.
fecha_pronóstico	timestamp	Fecha pronóstico del movimiento.

Tabla2.4 Descripción de la clase Vitalicio

Nombre: Vitalicio		
Descripción: registra los datos de las cuentas que son vitalicios		
Atributos	Tipo	Descripción
Cod_CFamiliar	varchar	Código de la cuenta
sucursal	int	Sucursal a la que pertenece la cuenta.
Importe	double	Importe almacenado en la cuenta.

Tabla2.5 Descripción de la clase Nómina_AyudaFamiliar

Nombre: Nómina_AyudaFamiliar		
Descripción: registra los datos de las cuentas que son de ayudar familiar		
Atributos	Tipo	Descripción
fecha	timestamp	Fecha en que se realiza la cuenta.
importe	double	Importe recibido en la cuenta
crédito	double	Importe total de la cuenta
débito	double	Importe que debe pagar.
sucursal	int	Sucursal a la que pertenece la cuenta.
crc	varchar	Código que es generado para verificar que las tarjetas de pago corresponden con la que está registrada.
observ	varchar	Observación.

Tabla2.5 Descripción de la clase Nómina_Congelada

Nombre: Nómina_Congelada		
Descripción: registra los datos de las cuentas que son congeladas.		
Atributos	Tipo	Descripción
importe	double	Importe recibido en la cuenta
crédito	double	Importe total de la cuenta
débito	double	Importe que debe pagar.
sucursal	int	Sucursal a la que pertenece la cuenta.
crc	varchar	Código que es generado para verificar que las tarjetas de pago corresponden con la que está

		registrada.
fecha	timestamp	Fecha de creación de la cuenta
generada	int	Para verificar si ya se generó los ajustes de una persona
observ	varchar	Observación.

2.5 Modelo Físico

El Modelo Físico de Datos se diseñó a partir de un análisis de las clases persistentes, documentos y plantillas generadas por el proyecto con el objetivo de asimilar como se gestiona la información en la UCCM. Para el diseño del modelo se utilizó la herramienta de diseño DBDesigner Fork que permite generar de forma automática la BD en PostgreSQL. Para consultar el modelo remitirse al (ANEXO2).

2.6 Descripción de las Tablas

Nombre: Colaborador		
Descripción: Almacena toda la información referente al colaborador		
Atributo	**Tipo**	**Descripción**
id	integer	Id del colaborador.
nombre	varchar(50)	Nombre del colaborador.
apellido1	varchar(50)	Primer apellido del colaborador.
apellido2	varchar (50)	Segundo apellido del colaborador.
sexo	varchar(15)	Tipo de sexo del colaborador.
ministerio_trabajo	varchar(15)	Ministerio de trabajo al que pertenece.
dirección_particular	varchar(100)	Dirección particular del colaborador.
dirección_particular_municipio	varchar(50)	Dirección del municipio en que reside.
dirección_particular_provincia	varchar(50)	Dirección de la provincia en que reside.

raza	varchar(1)	Tipo de raza.
teléfono	Integer	Teléfono del colaborador de su dirección particular.
correo	varchar(25)	Correo del colaborador.
foto	blob	Foto del colaborador.
nombre_padre	varchar(30)	Almacena el nombre del padre.
nombre_madre	varchar(30)	Almacena el nombre de la madre.
padre_vivo	bool	Conocer si el padre vive o no.
madre_vivo	bool	Conocer si la madre vive o no.
estado_civil	Integer	Identificador del estado civil del colaborador
necesidad_casa	bool	Conocer si necesita casa o no.
nombre_esposa	Integer	Identificador del nombre de la esposa.

Nombre: Colaborador_Salud		
Descripción: Almacena información referente a los colaboradores médicos que son profesionales de la salud.		
Atributo	**Tipo**	**Descripción**
Colaborador_id	integer	Identificación del colaborador.
id_profesional_salud	Integer	Almacena el id del profesional de salud.
id_tipo_profesional	Integer	Almacena el id dado por su profesión.

Nombre: Colaborador_No_Salud

Descripción: Almacena información referente a los colaboradores médicos que no son profesionales de la salud.

Atributo	Tipo	Descripción
colaborador_id	integer	Identificador del colaborador.
cod_cargos_id	Integer	Id del cargo.
cod_grado_cientifico	Integer	Id del grado científico.
cod_tipo_profesional_no_salud	Integer	id del tipo de profesional no salud
centro_trabajo	varchar(50)	Almacena el centro de trabajo.
centro_trabajo_provincia	Integer	Almacena el identificador de la provincia en que se encuentra el centro de trabajo.
centro_trabajo_municipio	Integer	Almacena el identificador del municipio en que se encuentra el centro de trabajo.

Nombre: Cod_Tipo_Profesional_No_Salud

Descripción: Esta tabla almacena información sobre los diferentes tipos de profesionales que no son colaboradores de Salud.

Atributo	Tipo	Descripción
id	integer	Id del tipo de profesional no salud.
tipo_profesional	varchar(50)	Almacena el tipo de profesional no salud.

Nombre: Cod_Grado_Científico

Descripción: Esta tabla almacena información sobre los diferentes tipos de grados científicos.

Atributo	Tipo	Descripción
id	integer	Id del grado científico.
grado_científico	varchar(50)	Almacena el grado científico.

Nombre: Cod_Cargos		
Descripción: Esta tabla almacena información sobre los diferentes tipos de cargos para los profesionales que no son colaboradores de Salud.		
Atributo	**Tipo**	**Descripción**
id	integer	Id del cargo.
nombre_cargo	varchar(50)	Nombre del cargo.

Nombre: Cod_ Profesión_No_Salud		
Descripción: Esta tabla almacena información sobre la profesión de los profesionales que no son colaboradores de Salud.		
Atributo	**Tipo**	**Descripción**
id	integer	Id de la profesión.
profesión	varchar(50)	Nombre de la profesión.

Nombre: Cod_Organizaciones_Políticas		
Descripción: Esta tabla almacena información sobre los diferentes tipos de organizaciones políticas.		
Atributo	**Tipo**	**Descripción**
id	integer	Id de la organización política.

nombre_organización	varchar(50)	Nombre de la organización a que pertenece.
siglas	varchar(10)	Siglas abreviadas de la organización.

Nombre: Colaborador_Organizaciones _Políticas		
Descripción: La siguiente tabla es una relación de muchos a muchos entre Cod_Organizaciones_Políticas y pp_Colaborador.		
Atributo	**Tipo**	**Descripción**
cod_organizaciones_politicas_id	integer	Identificador de Organizaciones Políticas.
colaborador_id	Integer	Identificador de colaborador.

Nombre: Misiones		
Descripción: Esta tabla almacena toda la información referente a las misiones.		
Atributo	**Tipo**	**Descripción**
id	integer	Id de la misión.
colaborador_id	integer	Identificador del colaborador.
cod_evaluación_colaborador_misión_id	Integer	Identificador de la evaluación del colaborador.
cod_relevo_id	Integer	Identificador del relevo.
cod_dedicación_misión_id	Integer	Identificador de la dedicación.
país_id	Integer	Identificador del país.
cod_tipo_misión_id	Integer	Identificador del tipo de misión
tiempo_estancia	Integer	Tiempo de desarrollo de la misión
fecha_inicio	Datetime	Fecha en que comienza la misión
fecha_fin	Datetime	Fecha en que culmina la misión
idciudadano_avisar_urgencia	Integer	Identificador para casos de urgencia del colaborador que se encuentra en misión

idciudadano_autoriza	Integer	Identificador del ciudadano que autoriza la misión
teléfono_avisar_urgencia	Integer	Teléfono para casos de emergencia del colaborador en misión
Teléfono_designado_ciudadano	Integer	Teléfono del designado para casos de emergencia.

Nombre: Cod_Evaluación_Colaborador_Misión		
Descripción: Esta tabla almacena información sobre las evaluaciones realizadas cuando se culmina la misión.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de la evaluación
calificacion_misión	varchar(50)	Calificación dada por la misión.

Nombre: Cod_Relevo		
Descripción: Esta tabla almacena información sobre los relevos para las misiones que se están realizando.		
Atributo	**Tipo**	**Descripción**
id	integer	Almacena el id del relevo
nombre_relevo	varchar(50)	Nombre del relevo

Nombre: Cod_Dedicación _Misión		
Descripción: Esta tabla almacena información sobre las funciones que están realizando cada colaborador en la misión		
Atributo	**Tipo**	**Descripción**
id	integer	Almacena el id de la dedicación.

nombre_dedicación_misión	varchar(50)	Nombre de la dedicación en la misión

Nombre: Cod_Tipo _Misión		
Descripción: Esta tabla almacena información sobre los diferentes tipos de misiones existentes.		
Atributo	**Tipo**	**Descripción**
id	integer	Almacena el id del tipo de misión.
nombre_tipo_misión	varchar(50)	Nombre del tipo de misión
descripción	varchar(100)	Breve descripción de la misión

Nombre: Cod_Grupo_Área_Geográfica		
Descripción: Esta tabla almacena los distintos grupos de áreas geográficas donde están ubicadas las diferentes misiones.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del grupo de áreas geográficas
nombre_grupo_área_geográfica	varchar(50)	Nombre del grupo del área geográfica.

Nombre: Cod _Area_Geográfica		
Descripción: Esta tabla almacena las áreas geográficas donde están ubicadas las diferentes misiones.		
Atributo	**Tipo**	**Descripción**
id	integer	Almacena el id del área geográfica.
cod_grupo_área_geográfica _id	Integre	Identificador del grupo de áreas geográficas
nombre_área_geográfica	varchar(50)	Nombre del de área geográfica.

Nombre: Cod_País		
Descripción: Esta tabla almacena los países donde están ubicadas las diferentes misiones.		
Atributo	**Tipo**	**Descripción**
id	integer	Almacena el id de país.
cod_área_geográfica_id	Integer	Identificador del área geográfica.
nombre_país	varchar(50)	Nombre del país.

Nombre: Cod_Estados_Países		
Descripción: Esta tabla almacena los estados de los países donde están ubicadas las diferentes misiones.		
Atributo	**Tipo**	**Descripción**
id	integer	Almacena el id del estado de un país.
cod_país_id	Integer	Identificador de país.
nombre_estados_países	varchar(50)	Nombre del estado.

Nombre: Expediente		
Descripción: Esta tabla almacena los datos de los expedientes de los colaboradores		
Atributo	**Tipo**	**Descripción**
id_colaborador	integer	Identificador del colaborador
fecha_entrada	datetime	Fecha de entrada del expediente a la UCCM
entrado_por	integer	Usuario que se encuentra en el sistema en ese momento.
id_estado_expediente	integer	Identificador del estado del expediente

Nombre: Vitalicio

Descripción:

Esta tabla almacena los datos de las cuentas que cobran por vitalicio

Atributo	Tipo	Descripción
id_colaborador	integer	Identificador del colaborador
cod_cfamiliar	varchar(20)	Código de la cuenta de ayuda familiar que no es más que un número de cuenta bancaria
sucursal	integer	Identificador de la sucursal por la que se efectúa el pago
importe	integer	Importe que recibe el colaborador en misión por cuenta vitalicio

Nombre: Nómina_AyudaFamiliar

Descripción:

Esta tabla almacena los datos de las cuentas que cobran ayuda familiar

Atributo	Tipo	Descripción
id_colaborador	integer	Identificador del colaborador
fecha	datetime	Fecha en que se produjo la nómina de pago
crc	varchar(20)	Código que es generado para verificar que las tarjetas de pago corresponden con la que está registrada.
importe	double	Importe que recibe el colaborador por la cuenta de ayuda familiar
sucursal	integer	Identificador de la sucursal por la que se efectúa el pago
observación	varchar(100)	Descripción que es realizada cuando se produce un crédito, débito en otras
débito	double	En el caso que deba recibir un importe debido a un problema de nómina o transacción bancaria.

crédito	double	En el caso de que deba pagar un crédito debido a un préstamo u otras operaciones bancarias.

Nombre: **Nómina_Congelada**		
Descripción: Esta tabla almacena los datos de las cuentas que son por nóminas congeladas		
Atributo	Tipo	Descripción
id_colaborador	integer	Identificador del colaborador
fecha	datetime	Fecha en que se produjo la nómina de pago
importe	double	Importe que recibe el colaborador por la cuenta de ayuda familiar
sucursal	integer	Identificador de la sucursal por la que se efectúa el pago
crc	varchar(20)	Código que es generado para verificar que las tarjetas de pago corresponden con la que está registrada.
débito	double	En el caso que deba recibir un importe debido a un problema de nómina o transacción bancaria.
crédito	double	En el caso de que deba pagar un crédito debido a un préstamo u otras operaciones bancarias.
generada	integer	Para verificar si ya se generó los ajustes de una persona
observación	varchar(100)	Descripción que es realizada cuando se produce un crédito, débito en otras

Nombre: Personalización

Descripción:

Esta tabla almacena los datos de las cuentas ya que es la encargada de abrir las cuentas en el banco a los colaboradores

Atributo	Tipo	Descripción
id_colaborador	integer	Identificador del colaborador
Cod_CFamiliar	integer	Número de cuenta de ayudar familiar
Cod_CCong	integer	Número de cuenta de nóminas congeladas
fecha	datetime	Fecha en que se realizó la personalización
id_sucursal	integer	Identificador de la sucursal a que pertenece

Nombre: Cod_Monto_X_Tiempo

Descripción:

Esta tabla almacena cuanto se le debe pagar a un colaborador basado en el tiempo que lleva de misión y en la misión a que pertenece.

Atributo	Tipo	Descripción
id	integer	Identificador del monto_x_tiempo
cod_forma_pago_id	integer	Identificador de forma pago
cod_tipo_misión_id	integer	Identificador del tipo misión
monto	integer	Cantidad de dinero en dependencia del tipo misión
meses	integer	Tiempo que lleva en la misión

Nombre: Cod_Forma_Pago

Descripción:

Esta tabla almacena la forma de pago en dependencia del tipo de cuenta.

Atributo	Tipo	Descripción

id	integer	Identificador de forma de pago
forma_pago	varchar(50)	Si es por cuenta especial, ayuda familiar o cuenta congelada

Nombre: Movimiento		
Descripción: Esta tabla almacena los diferentes movimientos que realizan los colaboradores.		
Atributo	**Tipo**	**Descripción**
fecha_inicio	date	Fecha en que se produce el movimiento.
misiones_id	integer	Identificador de misión
cod_tipo_mov_id	integer	Identificador del tipo movimiento
Id_ciudadano_mueve	integer	Identificador del ciudadano que está registrando el movimiento
motivo	varchar(100)	Descripción del movimiento
fecha_fin	date	Fecha en que culmina el movimiento
fecha_pronóstico	date	Fecha en que se pronostica el fin del movimiento
meses	integer	Tiempo que lleva en la misión

Nombre: Cod_Tipo_Movimiento		
Descripción: Esta tabla almacena los diferentes tipos de movimiento que realiza el colaborador.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del tipo de movimiento.
tipo_movimiento	varchar(50)	Tipo de movimiento que esta realizando ya sea por fin de misión, vacaciones entre otras.

descripción	varchar(100)	Descripción realizada de acuerdo con el movimiento.

Nombre: Cod_Estado_Expediente		
Descripción: Esta tabla almacena los diferentes tipos de estados en que se puede encontrar un expediente.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del estado de expediente.
estado	varchar(50)	Tipo de estado en que se encuentra el expediente ya sea incompleta o completa.

Nombre: Cod_Estados_Colaborador		
Descripción: Esta tabla almacena los diferentes tipos de estados en que se puede encontrar el colaborador.		
Atributo	**Tipo**	**Descripción**
id	Integer	Identificador del estado del colaborador.
cod_tipo_mov_id	Integer	Identificador del tipo de movimiento.
estado	varchar(50)	Tipo de estado en que se encuentra el colaborador ya sea activo, pasivo, aspirante, etc.

Nombre: Pasaporte		
Descripción: Esta tabla almacena los datos de los pasaportes de los colaboradores cuando se confeccionan.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del pasaporte
fecha_confección	datetime	Fecha en que se realizó la confección del pasaporte

observación	varchar(100)	descripción realizada sobre la confección del pasaporte
id_ciudadano_autoriza	integer	Identificador del ciudadano que autoriza
estado_pasaporte_id	integer	Identificador del tipo de pasaporte.

Nombre: Solicitud _Pasaporte		
Descripción: Esta tabla almacena los datos de las solicitudes de pasaporte que se confeccionan.		
Atributo	Tipo	Descripción
id	integer	Identificador de solicitud de pasaporte.
fecha_salida	datetime	Fecha en la que se debe producir el vuelo.
id_ciudadano_login	integer	Identificador del ciudadano que está interactuando con el sistema.
clave_org	varchar(20)	Clave de identificación de la organización.
fecha_actual	datetime	Fecha en el momento realizada la solicitud.
colaborador_id	integer	Identificador del colaborador.
tipo_solicitud_id	integer	Identificador de tipo de solicitud.
trámite_pasaporte_id	integer	Identificador de trámite de pasaporte.
país_id	integer	Identificador del país
tipo_pasaporte_id	integer	Identificador del tipo pasaporte.
categoría_trámite_id	integer	Identificador de categoría trámite.

Nombre: Cod_Estado_Pasaporte

Descripción:

Esta tabla almacena los diferentes tipos de estados en que se encuentra un pasaporte.

Atributo	Tipo	Descripción
id	integer	Identificador de estado pasaporte.
estado_pasaporte	varchar(50)	Estado en que se encuentra el pasaporte ya sea si está activo, inhabilitado o prórroga (que necesita volverse a renovar).

Nombre: Cod_Categoría_Trámite

Descripción:

Esta tabla almacena los diferentes tipos de categorías de trámites por las que se pueden realizar una solicitud de pasaporte.

Atributo	Tipo	Descripción
id	integer	Identificador de categoría tramite.
categoría_ trámite	varchar(50)	Categoría del tramité por la que se está realizando la solicitud ya sea por nueva confección, deterioro o perdida del pasaporte o por pasaporte vencido

Nombre: Cod_Trámite_Pasaporte

Descripción:

Esta tabla almacena los diferentes tipos de trámite de pasaporte que se pueden realizar.

Atributo	Tipo	Descripción
costo	integer	costo asociado en dependencia del trámite de pasaporte
id	integer	Identificador de trámite de pasaporte.
trámite_pasaporte	varchar(50)	Trámite de pasaporte por la cual se esta realizando la tramitación del pasaporte ya sea por confección, permiso de salida o prórroga.

Nombre: Cod_Tipo_Solicitud		
Descripción: Esta tabla almacena los diferentes tipos de solicitud de pasaporte que se pueden realizar		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador de estado pasaporte.
tipo_solicitud	varchar(50)	Tipo de solicitud por la que se realizó la solicitud del pasaporte ya sea normal, urgente o inmediato.

Nombre: Funcionarios_Autorizados		
Descripción: Esta tabla almacena los funcionarios que estan autorizados a firmar el modelo AO1.		
Atributo	**Tipo**	**Descripción**
id_funcionario	integer	Identificador del funcionario autorizado.

Nombre: Cod_Tipo_Pasaporte		
Descripción: Esta tabla almacena los diferentes tipos de pasaportes que se pueden realizar		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del tipo pasaporte.
tipo_pasaporte	varchar(50)	Especifica el tipo de pasaporte ya sea si es ordinario u oficial.
prórroga	integer	Especifica el tiempo de prórroga en que se deba validar nuevamente el pasaporte.
vencimiento	integer	Tiempo en que vence el pasaporte.

Nombre: AO1

Descripción:

Esta tabla almacena los datos del modelo que se recogen para la confección o rehabilitación del pasaporte

Atributo	Tipo	Descripción
id	integer	Identificador del modelo AO1.
dirección_particular	varchar(50)	Dirección particular del solicitante de pasaporte.
tiempo_estancia	integer	Tiempo de estancia en la misión.
centro_trabajo	varchar(50)	Centro de trabajo en el que reside.
folio	integer	Número de registro del modelo.

Nombre: Acta_Efectivo

Descripción:

Esta tabla almacena los datos del modelo que recoge el importe de operación que se realiza con el pasaporte.

Atributo	Tipo	Descripción
id_acta	integer	Identificador del acta de efectivo.
número_acta	integer	Número de registro del acta.
id_ciudadano_tramitador	integer	Identificador del ciudadano que realiza el acta.
id_ciudadano_tramitador	integer	Identificador del ciudadano que autoriza la realización del acta de efectivo.
importe	integer	Importe en dependencia del tipo de solicitud.

Nombre: Sol_Pasaporte

Descripción:

La siguiente tabla es una relación de muchos a muchos entre Solicitud_Pasaporte y Pasaporte

Atributo	Tipo	Descripción
solicitud_pasaporte_id	integer	Identificador de solicitud pasaporte e identificador de la tabla.
pasaporte_id	integer	Identificador de pasaporte e identificador de la tabla.

Nombre: Acta_Solicitud		
Descripción: La siguiente tabla es una relación de muchos a muchos entre Solicitud_Pasaporte y Acta_Efectivo		
Atributo	**Tipo**	**Descripción**
solicitud_pasaporte_id	integer	Identificador de solicitud pasaporte e identificador de la tabla.
acta_efectivo_id	integer	Identificador de acta efectivo e identificador de la tabla.

Nombre: Colab Hijos		
Descripción: La siguiente tabla almacena datos referentes a los hijos de los colaboradores.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del hijo del colaborador
colaborador_id	integer	Identificador de la tabla colaborador
nombre	varchar(50)	nombre del hijo.

Nombre: Usuario
Descripción: La siguiente tabla almacena los usuarios registrados en el sistema

Atributo	**Tipo**	**Descripción**
id	integer	Identificador del usuario.
usuario_php	varchar(20)	Nombre del usuario.

Nombre: Auditoria		
Descripción: Esta tabla almacena los registros de las trazas de modificación de los datos de las tablas.		
Atributo	Tipo	Descripción
id	integer	Identificador de la traza.
usuario_id	varchar(1)	Identificador de la tabla usuario.
acción	integer	Operación realizada.
nombre_tabla	varchar(50)	nombre de la tabla modificada
id_tabla_modificada	varchar	Identificador de la tabla que se modificó.
atributo_modificado	varchar	Atributo que se modificó.
valor_viejo	varchar	valor que tenía antes de ser modificado
valor_nuevo	varchar	valor nuevo
fecha_hora	timestamp	Fecha y hora en que se produjo la modificación.

En el capítulo se ha descrito el diseño propuesto, se explicó cómo se integra el sistema con el framework Symfony, una breve descripción de la arquitectura. Además se ha descrito cada entidad presente en la BD, y ha sido plasmado el modelo físico de la misma.

Capítulo 3: Validación Del Diseño Realizado

En este capítulo se brinda información sobre la validación tanto, teórica, como funcional del diseño de la BD propuesta. Se muestran las reglas de integridad, así como de normalización a tener en cuenta para obtener un diseño con calidad. Además se analiza la seguridad de la BD, la redundancia de los datos y las herramientas para el llenado voluminoso e inteligente de la misma.

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 evita errores de entrada introducidos por los usuarios descuidados o cualquier otra circunstancia de intento de violar la información existente en la BD.

3.1.1 Integridad

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

1. **Integridad referencial:** Garantiza interrelaciones válidas entre entidades. Implica que los datos sean correctos, sin duplicaciones, pérdida de datos o relaciones mal resueltas. Todas las bases de datos relacionales incluyen ésta propiedad pues el software gestor es responsable de su cumplimiento.

Un ejemplo de aplicación de este concepto en el modelo de datos es el siguiente: se asegura que los atributos cod_relevo_id y cod_tipo_mision_id de la tabla Misiones sean llaves foráneas, debido a que provienen y son llaves primarias de las entidades cod_relevo y cod_tipo_misión respectivamente. Además, cumplen con las restricciones siguientes:

1. Los datos cod_relevo_id y cod_tipo_mision_id fundamental de la tabla Misiones tienen los mismos dominios que en sus entidades de origen.

2. Al introducir los datos del cod_relevo_id y el cod_tipo_misión_id en la tabla Misiones, éstos tienen que encontrarse en su entidad de origen, es decir, la tabla Misiones no puede tener un cod_relevo_id que no

exista en la tabla cod_relevo y su cod_tipo_misión_id debe encontrarse de igual forma en la tabla cod_tipo_misión.

2. **Integridad de Dominio:** La integridad de dominio viene dada por la validez de las entradas de los datos para una columna determinada. Se puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KEY, definiciones DEFAULT, definiciones NOT NULL. [18].

3. **Integridad de la Entidad:** define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY.[19]

Restricción de PRIMARY KEY (llave primaria)

Toda entidad debe tener una llave primaria la cual identifica una tupla unívocamente de las demás. Por ejemplo:

En la tabla Colaborador que es donde se almacena toda la información referente a los colaboradores, se definió el atributo colaborador_pkey con la restricción PRIMARY KEY, lo que significa que este campo va a ser NOT NULL y además UNIQUE otra restricción que obliga a que cualquier valor tomado por ese atributo sea único entre todas las tuplas de esa tabla.

3.1.2 Normalización de la Base de datos

Una BD tiene que ser diseñada antes de que pueda ser creada y usada. El diseño debe ajustarse a estándares que permitan el ahorro de memoria, acceso rápido, fácil mantenimiento, portabilidad, facilidad de futuros mejoramientos, buen desempeño y eficiencia de costos, entre otros. El diseño lógico final de una BD debe ser tal que equilibre un desempeño óptimo junto con la integridad de la información. Esto puede ser logrado a través de un proceso conocido como normalización que consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional. Uno de los conceptos fundamentales que se maneja en la normalización es el de dependencia funcional. El proceso de normalización permite evitar algunos elementos que no son deseados en las bases de datos como:

1. Redundancia o repetición de los datos en el sistema.

2. Inconsistencias de actualización de datos como resultado de las actualizaciones parciales y la redundancia de la información.
3. Anomalías de borrado o pérdidas no intencionadas de datos.
4. Anomalías de inserción o imposibilidad de adicionar datos en la BD debido a la ausencia de otros datos.

El resultado del proceso de normalización es la obtención del modelo físico de la BD, en sus distintas formas normales, las utilizadas en el modelo se describen a continuación:

1ra Forma Normal (FN): Una relación está en 1ra FN si cumple con la propiedad de que sus dominios no contengan elementos que a su vez sean conjuntos.

1. La relación no incluye elementos repetitivos.
2. Toda relación normalizada, o sea, con valores atómicos de los atributos.

A continuación, un ejemplo en cual el nombre está compuesto por el nombre, primer apellido y segundo apellido, la primera tabla no se encuentra en primera forma normal (FN).

CI	**nombre**	**sexo**	**correo**
211044012321	Yoelvis Pozo Alfonso	M	yoelvis@uci.cu
211044012128	Marena Herrera Soria	F	marena@uci.cu
211045118401	Carlos Rafael Pichardo	M	carlos@uci.cu

Lo correcto sería:

CI	**nombre**	**primer apellido**	**segundo apellido**	**sexo**	**correo**
211044012321	Yoelvis	Pozo	Alfonso	M	yoelvis@uci.cu
211044012128	Marena	Herrera	Soria	F	marena@uci.cu
211045118401	Carlos	Pichardo	Alfonso	M	carlos@uci.cu

2da FN: Está basada en el concepto de dependencia funcional total, es decir, que los atributos no primos dependen totalmente de los atributos llaves y debe de cumplir con lo establecido en la 1ra FN.

La 2da FN se aplica solamente a los esquemas de relación que tienen claves primarias compuestas por dos o más atributos. Si un esquema de relación está en 1FN y su clave primaria es simple entonces está en segunda forma normal (2FN). Las relaciones que no están en 2FN pueden sufrir anomalías cuando se realizan actualizaciones.

Para la transformación de una relación de 1FN a 2FN hay que eliminar las dependencias parciales de su clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se descomponen en una nueva relación con una copia de los atributos de la clave primaria de los que dependen.

3era FN: Está basada en el concepto de dependencias transitivas.

1. Si y sólo si, la relación está en 2da FN.
2. Ningún atributo no primo de R depende transitivamente de la clave primaria.

Los atributos no llaves deben depender de toda la llave (2FN) de manera directa y no porque dependan de otro atributo no llave de la relación (ejemplo una llave foránea) que a su vez depende de la llave primaria.

id	**tiempo estancia**	**teléfono**	**fecha inicio**	**fecha fin**	**país**	**id país**
1	24	042347108	12-3-2010	25-4-2011	Cuba	1
2	24	078372613	24-5-2010	3-7-2011	Venezuela	2
3	24	043281107	23-9-2012	24-9-2013	Honduras	3

El país depende funcionalmente del Id país, lo que hace que no esté en 3FN. Lo correcto sería:

id	**tiempo estancia**	**teléfono**	**fecha inicio**	**fecha fin**	**id país**
1	24	042347108	12-3-2010	25-4-2011	1
2	24	078372613	24-5-2010	3-7-2011	2
3	24	043281107	23-9-2012	24-9-2013	3

país	**Id país**
Cuba	1
Venezuela	2
Honduras	3

La BD de Colaboración Médica actualmente se encuentra en 3era FN, porque cumple con las especificaciones de las tres formas expuestas anteriormente, es decir, en ella no existen atributos multievaluados, dependencias transitivas entre las relaciones y redundancia de la información. La 3era FN es la más usada casi en la totalidad de los productos que utilizan bases de datos, debido que estas garantizan poca o una redundancia casi nula de información; además, el uso de niveles más altos de normalización podría traer consigo una BD más relacional pero a la vez más compleja para trabajar y los tiempos de respuesta de varias de las consultas no serían los óptimos para responder a las peticiones del sistema.

3.1.3 Análisis de redundancia de información

La redundancia de información se refiere a aquellos datos duplicados que genera inconsistencia en la BD, requiriendo más espacio en disco; sin embargo en proyectos grandes es imposible evitarla completamente, lo que a veces se hace que se desee por cuestiones de rendimiento.

Los procesos de normalización mejoran en buena medida el comportamiento de este parámetro. En el caso de la BD Colaboración Médica v2.0, no existe información redundante, en la misma se analizaron las tablas con sus respectivas relaciones y se fueron eliminando consecuentemente todos los datos reiterativos en la BD.

3.1.4 Análisis de la seguridad de la base de datos

Las bases de datos son víctimas constantemente de ataques por piratas informáticos que tratan de acceder con el objetivo de robar o modificar alguna que otra información o simplemente violar la seguridad del sistema. Debido a esto cada BD debe de tener bien implementada una buena seguridad para evitar estos acometidos.

La seguridad de la aplicación se encuentra garantizada mediante el uso del sistema de autenticación autorización y auditoria (SAAA), que brinda el registro informatizado de la salud (RIS) y que permite la integración a los demás servicios que este registro brinda como son registro de ciudadanos, registro de personal de salud, registro de localidades entre otros, que poseen información crucial para el funcionamiento del sistema.

Es de gran importancia el uso de las potencialidades que brinda el gestor de BD en cuestiones de seguridad, el mismo puede mantener catálogos de los permisos de cada usuario independientemente de

los definidos en el sistema. Esto permite conocer quién puede realizar consultas, actualizaciones, etc., según el juego de privilegios de acceso y de restricciones aplicables a cada objeto de la BD (tablas, vistas y secuencias).

Para evitar la pérdida de los datos en caso de ocurrir alguna falla, PostgreSQL permite realizar salvas de la BD mediante el volcado (dump), pueden realizarse de forma automática o manualmente. Para realizar las copias se utiliza el comando pg_dump que cuenta con una serie de parámetros adicionales para indicar el nombre de la BD, el usuario y contraseña para conectarse a ella, el nombre que tomará el archivo de salva y donde se desea guardarla.

3.1.4 Trazabilidad de las acciones

La trazabilidad es la capacidad que posee un sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos. Al referirse a la trazabilidad se puede decir entonces que existe la posibilidad de conocer o establecer, entre la información que se maneja de un sistema, quién ha realizado qué, y poder llevar el sistema a un estado anterior.

Para lograr la trazabilidad de las acciones que desarrollan los usuarios del Sistema de Colaboración Médica v 2.0 en la BD, se creó un par de tablas adicionales y completamente independientes de los requerimientos de la UCCM que permiten almacenar información útil de las modificaciones realizadas sobre la BD como son: usuario que realiza una acción, la fecha y hora, acción realizada, tabla afectada, valores anteriores y valores modificados, así como su antiguo identificador para poder retornar la BD a su estado inicial en un momento determinado.

Una vez creadas las tablas mencionadas anteriormente, se implementaron un conjunto de Disparadores sobre las tablas de mayor interés por la UCCM para el seguimiento de la trazabilidad de las acciones de los usuarios. Los Disparadores son un mecanismo que ofrece el SGBD PostgreSQL para invocar una función determinada cada vez que ocurre un evento sobre una tabla en particular (entiéndase por evento: una inserción, modificación o borrado de alguna tupla). Las tablas seleccionadas fueron las que almacenan información referente a Colaboradores, Misiones, Movimiento, Vitalicio, Nómina de Ayuda Familiar y Cuentas Congeladas.

3.2 Validación funcional de la base de datos

3.2.1 Ambiente de las pruebas

Las bases de datos donde se realizaron las pruebas llevan como nombre prueba. Se crearon en un servidor con las siguientes características:

1. Sistema Operativo: Debian etch 4.0
2. Memoria RAM: 1 GB
3. Disco duro: 250
4. Velocidad CPU: 3.2:GHz

Para el llenado de las mismas se utilizó la herramienta Advance Data Generator, ya que permite generar datos para una o varias tablas a la vez y respeta la integridad referencial generando los datos que provienen de otras tablas para evitar errores. Se generaron un volumen de 5000 tuplas para las tablas principales (Misiones, Colaborador, Movimiento, Colaborador Salud y Colaborador No Salud) donde se realizarían las consultas más específicas. Se elaboraron consultas de tipo SELECT, INSERT y UPDATE de diferente complejidad (ANEXO3) utilizando el lenguaje SQL y se elaboraron de acuerdo con el modelo de datos diseñado en el presente trabajo de diploma. La herramienta utilizada para medir el rendimiento de las consultas fue el JMeter.

3.2.2 Herramientas para pruebas de carga intensiva (selección de consultas)

El objetivo de las pruebas de carga es verificar el tiempo de respuesta del sistema para transacciones, bajo diferentes condiciones de carga. Las pruebas de carga miden la capacidad del sistema para continuar funcionando apropiadamente bajo diferentes condiciones de carga. La meta de las pruebas de carga es determinar y asegurar que el sistema funciona apropiadamente aún más allá de la carga de trabajo máxima esperada. [20]

Para diseñar pruebas en el JMeter se realiza de la siguiente forma:

El componente principal es denominado Test Plan o Plan de Pruebas donde se definen todos los aspectos relacionados con la prueba. El plan de pruebas se muestra automáticamente cuando se abre la aplicación. Primeramente se crea una Lo Thread Group o Grupo de Hilos, considerado como el grupo de usuarios o peticiones que se desea simular para la BD. Cuando es creado el grupo de hilos se llenan las siguientes opciones:

1. **Nombre:** Se define un nombre
2. **Number of Threads (users):** Equivale al número de usuarios que se desean simular.
3. **Ramp-Up Period:** Es el lapso de tiempo en segundos que se desea tener entre cada grupo de peticiones se usará en este caso 1
4. **Loop Count o Forever:** Se utiliza para indicar si la simulación para grupos de hilos será llevada acabo infinitamente, o por un ciclo determinado de veces.

Una vez definidas las características del Grupo de Hilo se pasa a generar las JDBC Connection Configuration donde se crea una para cada BD a probar. Las JDBC Connection Configuration se usan para configurar las conexiones de las bases de datos Esta posee los siguientes aspectos:

1. **Nombre de la variable:** El que se desee.
2. **Database URL:** Dirección donde se encuentra la BD en el servidor, ejemplo: jdbc:postgresql://10.36.6.222:5432/prueba
3. **JDBC Driver class:** Para PostgreSQL org.postgresql.Driver
4. **El usuario y contraseña** de la BD.

El resto de los campos que aparecen están predefinidos por defecto. Luego se generan las JDBC Request utilizadas para definir las requisiciones de simulación, en la cual aparecerán las siguientes opciones:

1. **Name:** El que se desee dar a la consulta.
2. **Variable Name:** El mismo que se definió en la variable Name de la JDBC
3. **Tipo de consulta:** Select Statement en caso de ser una consulta SELECT y así sucesivamente.
4. **Consulta (Query):** Donde se define la consulta.

Luego se generan los Listener para mostrar los resultados. Para la realización de la prueba se utiliza el listener **Summary Report,** los datos que presentan son:

Label: etiqueta de la muestra

#Muestras: cantidad de peticiones realizadas a la BD.

Media: tiempo promedio en milisegundos para un conjunto de resultados.

Min: tiempo mínimo que demora una petición en acceder a la BD.

Max: tiempo máximo que demora una petición en acceder a la BD.

%Error: porcentaje de peticiones con errores.

Rendimiento: rendimiento medido en las peticiones por segundo / minuto / hora.

3.3 Resultado de las Pruebas

3.3.1 Consultas de Selección

En la tabla 1 se muestra el rendimiento en segundos, así como el número mínimo, máximo y media de las peticiones realizadas para cada una de las 7 consultas de selección.

# Muestras	Media	Mín	Máx	% Errc	Rendimiento/seg
100	16	1	172	0.0	93,50
100	2	2	6	0.0	95,10
100	9	7	38	0.0	90,10
100	2	1	6	0.0	96,00
100	5	4	9	0.0	95,70
100	1674	139	3046	0.0	29,30
100	534	52	1436	0.0	54,60

Tabla1.Resultado de las consultas de selección

En la figura 4 se muestra un gráfico que representa el rendimiento para cada una de las consultas de selección.

Figura 3.Comportamiento del rendimiento para las consultas de selección

El rendimiento se ve afectado pues en las consultas #6 y #7, la cadena de encuentro entre tablas es mayor y por tanto el rendimiento se ve afectado por la cantidad de tuplas que existen en estas tablas y la realización del encuentro entre las tablas implica una demora en la obtención de los resultados.

3.3.2 Consultas de Inserción

En la tabla 2 se muestra el rendimiento en segundos, así como el número mínimo, máximo y media de las peticiones realizadas para cada una de las 4 consultas de inserción.

Rendimiento de las Consul	# Muestras	Media	Mín	Máx	% Error	Rendimiento/seg
1	100	5	0	110	0.0	88,90
2	100	7	0	109	0.0	94,10
3	100	0	0	16	0.0	90,20
4	100	0	0	16	0.0	92,80

Tabla 2. Resultado de las consultas de inserción.

La Figura 5 muestra un gráfico que representa el comportamiento del rendimiento para cada una de las consultas de inserción.

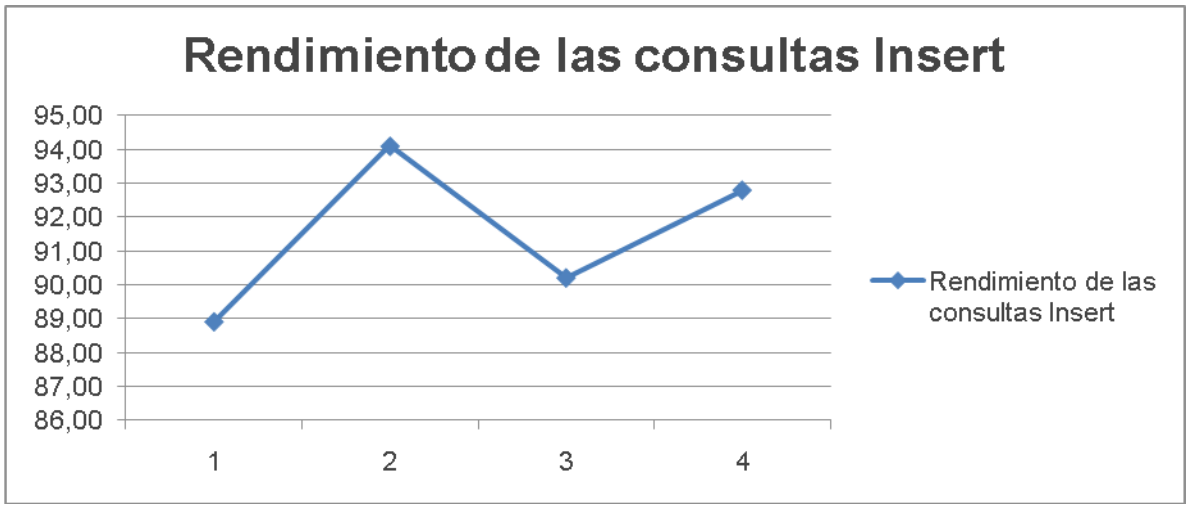


Figura 4. Comportamiento del rendimiento para las consultas de inserción

3.3.3 Consultas de Actualización

En la Tabla 3 se muestra el rendimiento en segundos, así como el número mínimo, máximo y media de las peticiones realizadas para cada una de las 4 consultas de actualización.

Rendimiento de las consul	# Muestras	Media	Mín	Máx	% Error	Rendimiento
1	100	20	0	140	0.0	84,20
2	100	10	0	94	0.0	84,20
2	100	2	0	31	0.0	91,50
4	100	50	0	344	0.0	90,10

Tabla 3. Resultado de las consultas de actualización.

La Figura 6 muestra un gráfico que representa el comportamiento del rendimiento para cada una de las consultas de actualización.

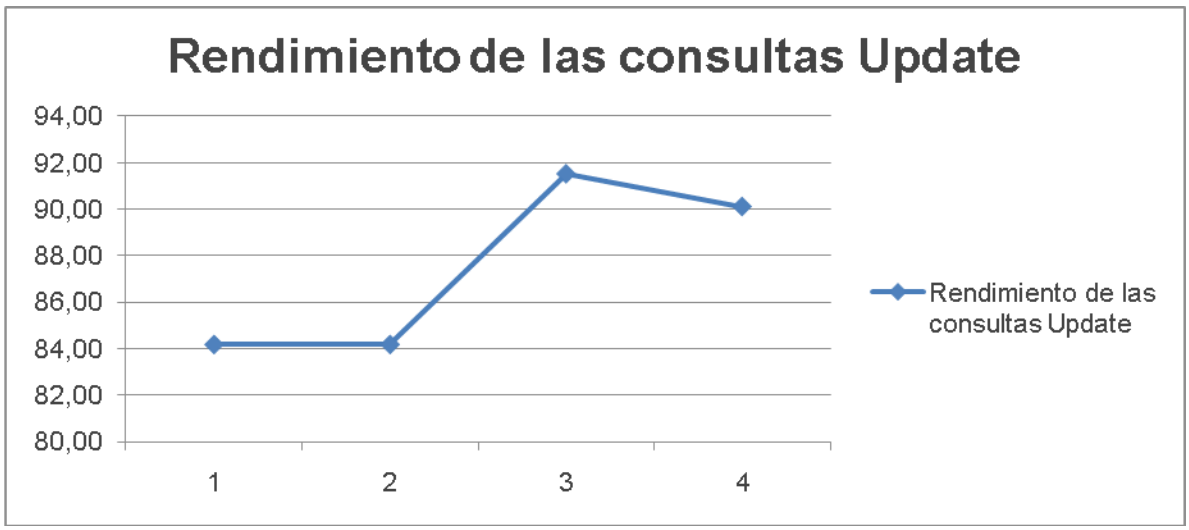


Figura 5. Comportamiento del rendimiento para las consultas de actualización

En este capítulo se ha realizado validación del diseño propuesto. Para ello se han analizado aspectos como: la integridad, normalización, redundancia de la información, seguridad de la BD y la trazabilidad de las acciones. Se realizó pruebas de rendimientos con el objetivo de validar funcionalmente el diseño de la BD para el Sistema de Colaboración Médica v 2.0.

Conclusiones

Con el desarrollo del presente trabajo:

1. Se asimilaron los conceptos relacionados con el diseño de BD relacionales (entidades, atributos y relaciones) lo que permitió el diseño de un modelo de datos capaz de almacenar los datos generados en el proceso de gestión de la información del Sistema de Colaboración Médica v.2.0.
2. Se analizaron los procesos de negocio y la BD empleada por el Sistema de Colaboración Médica v1.0, lo cual arrojó algunas inconsistencias con respecto a requerimientos de la UCCM. Todo esto posibilitó la obtención de un modelo de datos cualitativamente superior a su precedente.
3. Se asimilaron los principales elementos del modelo del framework Symfony, el trabajo con el ORM Doctrine, así como las pautas que propone para la elaboración de modelos de datos lo que permitió añadirle al modelo de datos obtenido mayor portabilidad.

Recomendaciones

Se recomienda:

1. Una vez implementado el diseño expuesto, mantener sobre la BD un proceso de mantenimiento y copias de seguridad periódicas, con la intención de mantener la fiabilidad, integridad y funcionamiento óptimo de la BD.
2. Realizar un estudio que permita conocer la tasa de crecimiento de la tabla traza contemplada en el diseño del modelo de datos propuesto. Para determinar cuando el volumen de información pueda afectar los tiempos de respuestas deseados para las peticiones del Sistema de Colaboración Médica v2.0. Todo ello con el objetivo de implementar posteriormente una estrategia que permita solucionar esta situación.

Referencias Bibliográficas

[1] Hernández Orallo, José. La disciplina de los sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas. Valencia: s.n., 2002. Disponible en:

http://palomo.usach.cl/Docs/BD/JHernandezO-La_disciplinade_los_sistemas.pdf

[2] Frómeta Silvente, Serguéi. 2008. Modelo lógico y físico de la base de datos del módulo de Investigaciones Forenses del proyecto CICPC. Ciudad de la Habana: s. n., 2008.

[3] Ídem a la 2.

[4]. Pérez, Sara. Modelos de Bases de Datos. [En línea] 2001. [Citado el: 2 de abril de 2010.]

http://www.desarrolloweb.com/articulos/modelos-base-datos.html

[5] Ídem a la 4.

[6] Asociación Peruana de Software Libre. Introducción a PostgreSQL. [En línea] [Citado el: 2 de abril] de 2010.]

http://apesol.org.pe.

[7] [En línea] [Citado el: 2 de enero de 2010]

http://www.manualesdeayuda.com/manuales/bases-de-datos/postgresql/.

[8] Ídem a la 7.

[9] Quiñones A. Ernesto. Introducción a PostgreSQL. [En línea]. [Citado el: 23 de mayo de 2010.]

http://postgresql.org.pe/articles/introduccion_a_postgresql.pdf

[10] GRADY, BOOCH, IVAR, JACOBSON y JAMES, RUMBAUGH. 2006. *El lenguaje Unificado de Modelado.* S.l.: Pearson Prentice Hall, 2006. 8478290761.

[11] Fernández Vilas, Ana. Diagramas UML. [En línea] 2001. [Citado el: 25 de enero de 2010.] http://tvdi.det.uvigo.es/~avilas/UML/node18.html .

[12] Rambla Informática. Enterprise Architect. [En línea]. [Citado el: 6 de abril de 2010.]

http://tienda.ramblainf.com/epages/tienda_ramblainf_com.sf/es_ES/

[13] Pg Admin. Características. [En línea] [Citado: 3 de marzo, 2010.] http://www.pgadmin.org/features.php .

[14] The Apache Jakarta Project. Apache Jmeter. [En línea]. [Citado: 3 de marzo, 2010.]

http://jakarta.apache.org/jmeter/

[15] Todo Programas. DBDesigner Fork 1.4. [En línea]. [Citado: 15 de marzo, 2010.]

http://www.todoprogramas.com/programa/dbdesignerfork .

[16] Tecnoretales. Introducción al ORM. [En línea]. [Citado: 15 de marzo, 2010.]

http://www.tecnoretales.com/programacion/que-es-doctrine-orm/

[17] Symfony y el libro Doctrine. Trabajando con los datos Capítulo 6 [En línea]. [Citado: 15 de marzo, 2010.]

http://www.symfony-project.org/doctrine/1_2/es/06-Working-With-Data

[18] Fernández Macías, Dayron Diseño de una base de datos para controlar la información de Unión de Jóvenes Comunistas en la UCI, 2008.

[19] Microsoft. Integridad de los datos. [En línea] [Citado: 2 de abril, 2010.]

http://msdn.microsoft.com/es-es/library/ms184276.aspx

[20] González Ballester, Lidier. Diseño de una base de datos para el control de los RRHH en los polos productivos de la facultad 9, 2009

Bibliografía

- [En línea] [Citado el: 2 de enero de 2010]

  http://www.manualesdeayuda.com/manuales/bases-de-datos/postgresql/.

- Asociación Peruana de Software Libre. Introducción a PostgreSQL. [En línea] [Citado el: 2 de abril] de 2010.]

  http://apesol.org.pe.

- Ayuda Extendida del Rational Enterprise Edition en Español 2007.

- Date C. J. Introducción a los sistemas de bases de datos, Editorial Félix Varela, La Habana, 2003.

- Dr. Conejo Muñoz, Ricardo. Tipos de Bases de Datos. [En Línea][Citado: febrero de 2010]; Disponible en:

  http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf .

- E.F. Codd. Modelo Relacional de Datos para grandes Bancos de Información, June 1970.

- Fernández Macías, Dayron. Diseño de una base de datos para controlar la información de Unión de Jóvenes Comunistas en la UCI, 2008.

- Fernández Vilas, Ana. Diagramas UML. [En línea] 2001. [Citado el: 25 de enero de 2010.] http://tvdi.det.uvigo.es/~avilas/UML/node18.html .

- Frómeta Silvente, Serguéi. 2008. Modelo lógico y físico de la base de datos del módulo de Investigaciones Forenses del proyecto CICPC. Ciudad de la Habana: s. n., 2008.

- González Ballester, Lidier. Diseño de una base de datos para el control de los RRHH en los polos productivos de la facultad 9, 2009.

- González, Lic. Roberto Acosta. Sistema de Gestión para los Colaboradores de la Salud. Ciudad de La Habana: s.n 2008.

- Hernández Orallo, José. La disciplina de los sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas. Valencia: s.n., 2002. Disponible en:

  http://palomo.usach.cl/Docs/BD/JHernandezO-La_disciplinade_los_sistemas.pdf

- GRADY, BOOCH, IVAR, JACOBSON y JAMES, RUMBAUGH. 2006. El lenguaje Unificado de Modelado. S.l.: Pearson Prentice Hall, 2006. 8478290761.
- Hansen W, Gary; Hansen V, James. Diseño y Administración de Bases de Datos. 2da Edición.
- Leyva Domínguez, Arnoldo. Modelo Lógico y Físico de la Base de Datos.
- Mato Gracia, Rosa M. Diseño de Bases de Datos, Octubre 2005.
- Microsoft. Integridad de los datos. [En línea]. [Citado el: 2 de abril de 2010.]

  http://msdn.microsoft.com/es-es/library/ms184276.aspx

- Quiñones A. Ernesto. Introducción a PostgreSQL. [En línea]. [Citado el: 23 de mayo de 2010.]
- Pérez Mora, Oscar; Gibert Ginesta, Marc. Base de Datos en Postgre.
- Pérez, Sara. Modelos de Bases de Datos. [En línea] 2001. [Citado el: 2 de abril de 2010.]

  http://www.desarrolloweb.com/articulos/modelos-base-datos.html

- Pg Admin. Características. [En línea] [Citado: 3 de marzo, 2010.]

  http://www.pgadmin.org/features.php

- Rambla Informática. Enterprise Architect. [En línea]. [Citado el: 6 de abril de 2010.]

  http://tienda.ramblainf.com/epages/tienda_ramblainf_com.sf/es_ES/

- Sitio Web Oficial PostgreSQL. [En línea]

  http://www.postgresql.org/about/press/presskit82.html.es

- Symfony y el libro Doctrine. Trabajando con los datos Capítulo 6 [En línea]. [Citado: 15 de marzo, 2010.]

  http://www.symfony-project.org/doctrine/1_2/es/06-Working-With-Data

- Tecnoretales. Introducción al ORM. [En línea]. [Citado: 15 de marzo, 2010.]

  http://www.tecnoretales.com/programacion/que-es-doctrine-orm/

- Todo Programas. DBDesigner Fork 1.4. [En línea]. [Citado: 15 de marzo, 2010.]

  http://www.todoprogramas.com/programa/dbdesignerfork .

- 26. The Apache Jakarta Project. Apache Jmeter. [En línea]. [Citado: 3 de marzo, 2010.]

  http://jakarta.apache.org/jmeter/

- 27. Zaninotto, François; Potencier Fabien. Symfony, La guía definitiva 1.2, Editorial Apress. Disponible en:

  http://www.librosweb.es/symfony/index.html .

Glosario

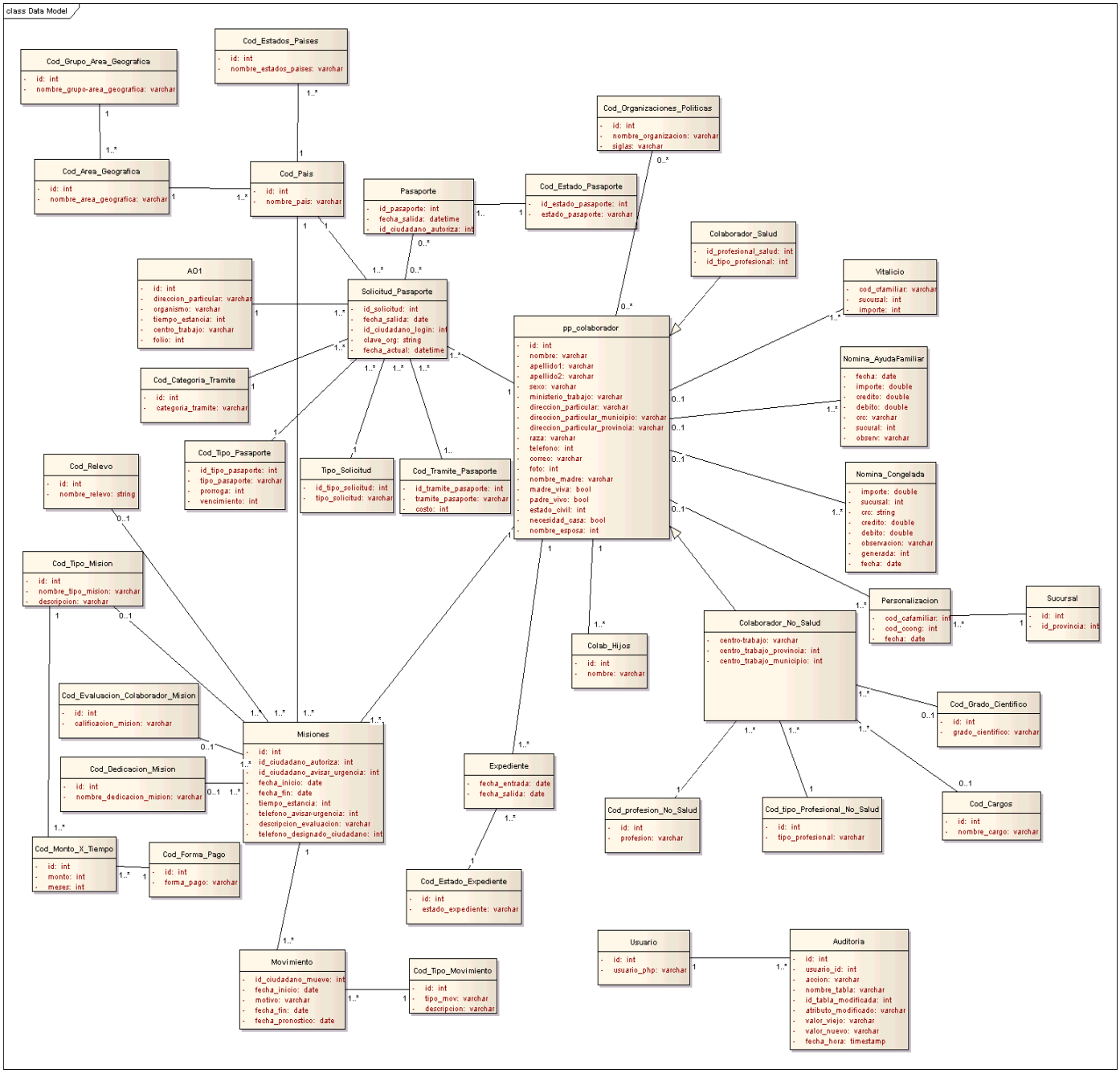
- **Doctrine:** capa de abstracción que se utiliza para el ORM. Proporciona persistencia para los objetos y un servicio de consultas optimizadas.
- **Escalabilidad:** La capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.
- **Framework:** en el desarrollo de software, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.
- **Gestor de base de datos**: Es un tipo de software específico, dedicado a servir de interfaz entre la BD, 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.
- **Multiplataforma:** Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos.
- **Normalización de bases de datos**: Consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional. Se usa para evitar la redundancia de los datos, problemas de actualización de los datos en las tablas, proteger la integridad de los datos.

- **ORM**: Acrónimo de Object-Relational Mapping (Mapeo Objeto-Relacional). Framework que utiliza técnicas de programación para convertir datos a objetos y viceversa, permitiendo el trabajo con datos persistentes como si formaran parte de una BD orientada a objetos.
- **Querys**: Consiste en una cadena de consulta, normalmente, se utilizan para: insertar, actualizar o editar valores de la BD.
- **Servidor:** Un servidor, en informática o computación, es el ordenador en el que se ejecuta un programa que realiza alguna tarea en beneficio de otras aplicaciones llamadas clientes.

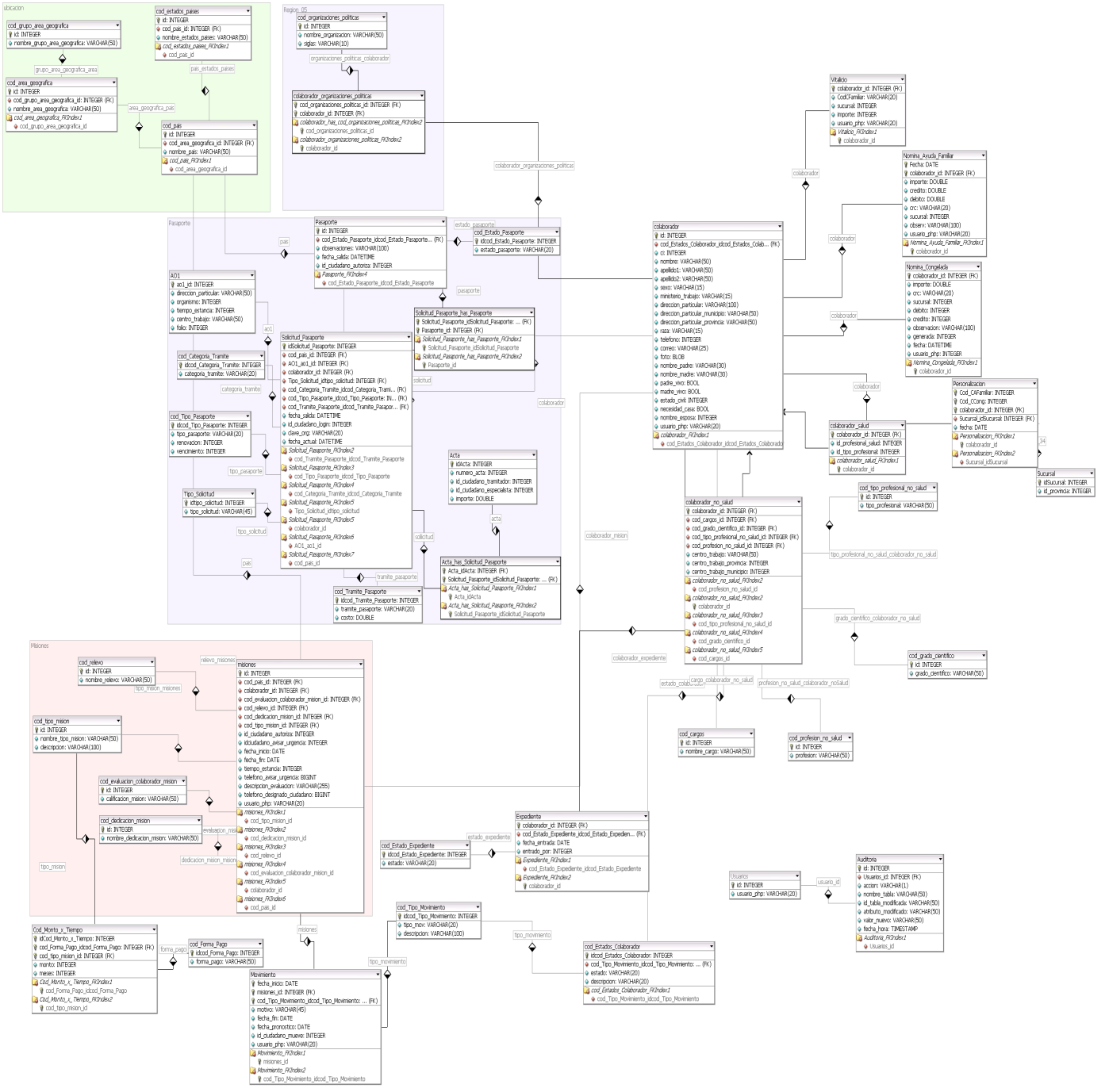
- **Triggers:** Un trigger o disparador en una BD, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).
- **Tupla:** Es la representación de una fila en una de las tablas que se está almacenando datos. Y las cuales serán llamadas por los administradores de BD en el tiempo de ejecución de un sistema.
- **Redundancia:** Repetición de una información previamente existente.

ANEXOS

Anexo1. Diagrama de Clases Persistentes



Anexo2. Modelo Físico



Anexo 3.Consultas para las Pruebas

Consultas de Selección.

1.

```
SELECT
  public.colaborador.nombre,
  public.colaborador.apellido1,
  public.colaborador.apellido2
FROM
  public.colaborador
  INNER JOIN public.colaborador__no__salud ON (public.colaborador.id =
public.colaborador__no__salud.colaborador_id)
  where public.colaborador__no__salud.colaborador_id =1
```

2.

```
SELECT
  public.colaborador.id,
  public.colaborador__salud.colaborador_id,
  public.colaborador__salud.id_profesional_salud,
  public.colaborador__salud.id_tipo_profesional,
  public.colaborador.ci,
  public.colaborador.nombre,
  public.colaborador.apellido1,
  public.colaborador.apellido2,
  public.colaborador.sexo,
  public.colaborador.ministerio_trabajo,
  public.colaborador.dirección_particular,
  public.colaborador.dirección_particular_municipio,
  public.colaborador.dirección_particular_provincia,
  public.colaborador.raza,
  public.colaborador.telefono,
  public.colaborador.correo,
  public.colaborador.foto,
  public.colaborador.nombre_padre,
  public.colaborador.nombre_madre,
  public.colaborador.padre_vivo,
  public.colaborador.madre_vivo,
  public.colaborador.estado_civil_id,
  public.colaborador.necesidad_casa,
  public.colaborador.nombre_esposa,
  public.colaborador.cod_estados_colaborador_id,
  public.colaborador.created_at,
  public.colaborador.updated_at,
  public.colaborador.usuario_php
FROM
```

```
  public.colaborador
  INNER JOIN public.colaborador__salud ON (public.colaborador.id =
public.colaborador__salud.colaborador_id)
WHERE
  public.colaborador__salud.colaborador_id = 5001
```

3.

```
SELECT
  count (public.colaborador.id) AS cantidad
FROM
  public.colaborador
  INNER JOIN public.misiones ON (public.colaborador.id = public.misiones.colaborador_id)
  INNER JOIN public.cod__país ON (public.misiones.cod_país_id = public.cod__país.id)
  INNER JOIN public.cod__estados__países ON (public.cod__país.id = public.cod__estados__países.id)
WHERE
  public.cod__país.nombre_país = 'Venezuela Republica Bolivariana'
```

4.

```
SELECT
  public.colaborador.ci,
  public.colaborador.nombre,
  public.colaborador.apellido1,
  public.colaborador.apellido2
FROM
  public.colaborador
  INNER JOIN public.misiones ON (public.colaborador.id = public.misiones.colaborador_id)
  INNER JOIN public.cod__dedicación__misión ON (public.misiones.cod_dedicación_ misión _id =
public.cod__dedicación__misión.id)
WHERE
  public.cod__dedicación__mision.nombre_dedicación_ misión = 'Informático'
```

5.

```
SELECT
  public.expediente.colaborador_id
FROM
  public.colaborador
  INNER JOIN public.misiones ON (public.colaborador.id = public.misiones.colaborador_id)
  INNER JOIN public.colaborador__salud ON (public.colaborador.id =
public.colaborador__salud.colaborador_id)
  INNER JOIN public.expediente ON (public.colaborador.id = public.expediente.colaborador_id)
WHERE
  public.misiones.fecha_inicio = '6/10/2010'
ORDER BY
  public.expediente.colaborador_id DESC
```

6.

```
SELECT
  public.colaborador.nombre,
  public.colaborador.apellido1,
```

```
  public.colaborador.apellido2,
  public.colaborador.telefono,
  public.colaborador.dirección_particular,
  public.colaborador.necesidad_casa,
  public.colaborador__salud.id_profesional_salud
FROM
  public.colaborador
  INNER JOIN public.misiones ON (public.colaborador.id = public.misiones.colaborador_id)
  INNER JOIN public.cod__evaluación__colaborador__mision ON
(public.misiones.cod_evaluación_colaborador_misión_id =
public.cod__evaluación__colaborador__mision.id)
  INNER JOIN public.colaborador__salud ON (public.colaborador.id =
public.colaborador__salud.colaborador_id)
WHERE
  public.cod__evaluación__colaborador__misión.calificación_misión = 'buena'
```

7.

```
SELECT
  public.misiones.fecha_inicio,
  public.misiones.fecha_fin
FROM
  public.colaborador
  INNER JOIN public.misiones ON (public.colaborador.id = public.misiones.colaborador_id)
  INNER JOIN public.cod__tipo__misión ON (public.misiones.cod_tipo_misión _id =
public.cod__tipo__misión.id)
WHERE
  public.cod__tipo__misión.nombre_tipo_misión ='PIS'
```

Consultas de Actualización

1.

```
UPDATE
  public.movimiento
SET
  idciudadano_mueve = 32,
  motivo = 'vacaciones'
WHERE
  public.movimiento.misiones_id = 54609
```

2.

```
UPDATE
  public.colaborador
SET
  nombre = 'alejandro',
  apellido1 = 'dsfsd',
  apellido2 = 'sdff',
  sexo =' m',
```

```
  ministerio_trabajo = 3,
  dirección_particular = 3,
  dirección_particular_municipio = 4,
  dirección_particular_provincia = 'regdfgfd',
  raza = 'blanca',
  telefono = 565466,
  correo = 'ddfg@fdfd.com',
  nombre_padre = 'aassad',
  nombre_madre = 'ffffd',
  estado_civil_id = 1,
  nombre_esposa = 'elena',
  cod_estados_colaborador_id = 1,
  usuario_php = 'pepe'
WHERE
  public.colaborador.id = 1
```

3.

```
UPDATE
  public.cod__tipo__movimiento
SET
  descripcion = 'EnfermoGrave',
  tipo_mov = 'Asma'
WHERE
  public.cod__tipo__movimiento.id = 2
```

4.

```
UPDATE
misiones
SET
 cod_país_id = '16',
descripción_evaluación ='buena conducta'
 WHERE
 public.misiones.id=54730
```

Consultas de Inserción

1.

```
INSERT INTO
  public.misiones(
  Id,
  colaborador_id,
  cod_evaluación_colaborador_misión_id,
  cod_relevo_id,
  cod_dedicación_misión_id,
  cod_tipo_misión_id,
  cod_país_id,
```

```
  id_ciudadano_avisar_urgencia,
  id_especialidad_salida,
  id_designado_ciudadano,
  tiempo_estancia,
  telefono_avisar_urgencia,
  descripción_evaluación,
  teléfono_designado_ciudadano,
  usuario_php)
VALUES (
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  1,
  24,
  34543,
  'hola',
  43555,
 ' pepe')
```

2.

```
INSERT INTO
  public.colaborador__salud
(
  colaborador_id,
  id_profesional_salud,
  id_tipo_profesional
)
VALUES (
 1,
4,
4
)
```

3.

```
NSERT INTO
  public.cod__evaluación__colaborador__misión
(
  Id,
  calificación_misión
```

```
)
VALUES (
5,
'pésima'
)
```

4.

```
INSERT INTO
  public.vitalicio(
  colaborador_id,
  cod_cfamiliar,
  sucursal,
  importe,
  usuario_php)
VALUES (
  1,
  234,
  43,
  50,
  'alfredo')
```