Universidad de las Ciencias Informáticas
Facultad 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Automatización del proceso de Gestión de la Información de Recursos de la facultad 3. Rol de diseñador de base de dato.

Autor:

Maykel Quintana Linares.

Tutor:

Dr.C Pascual Verdecia Vicet.

Ciudad de La Habana Cuba.

Junio, 2008

DECLARACIÓN DE AUTORÍA

Declaro ser el 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 _____.

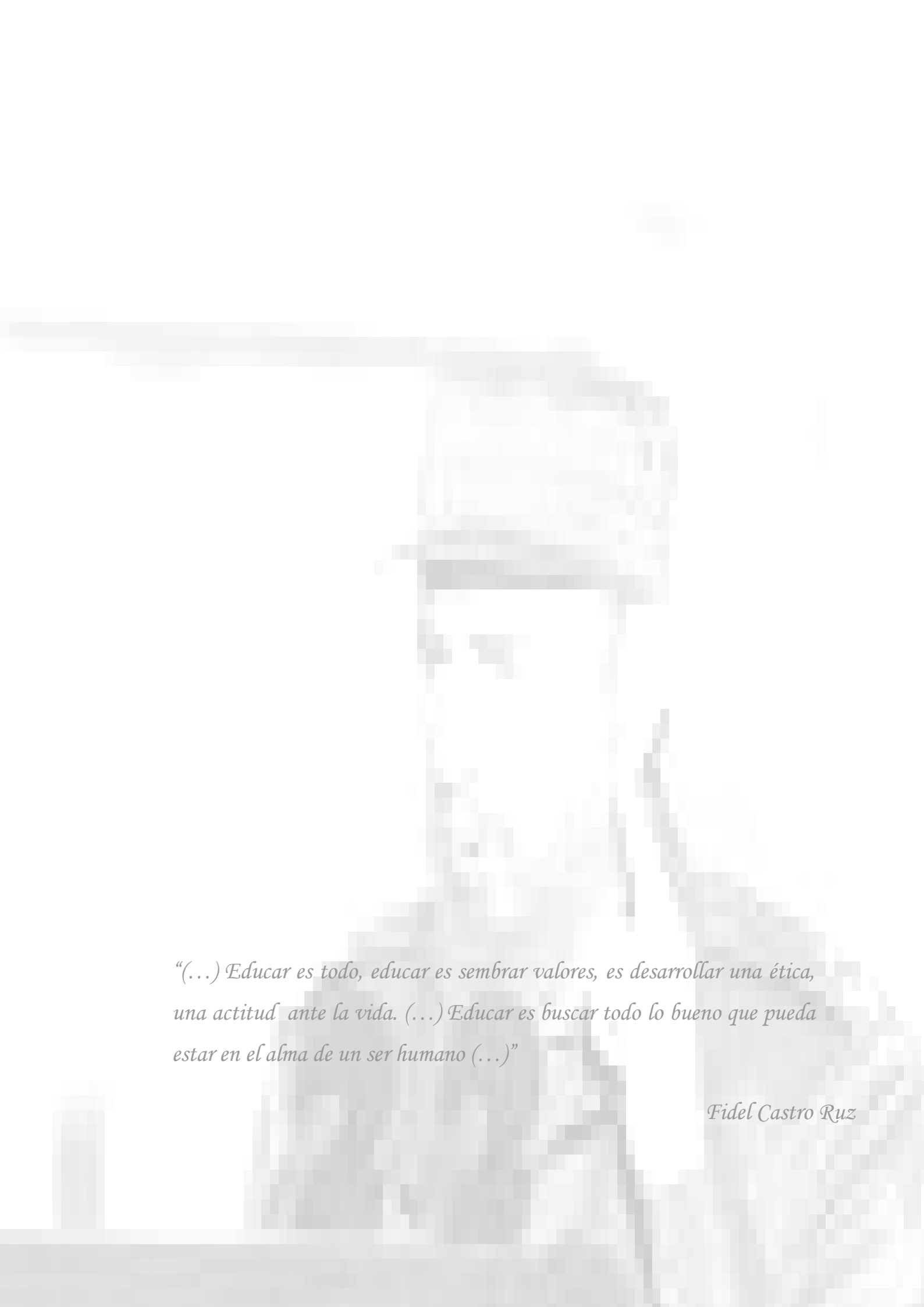
_____

Firma del autor
(**Maykel Quintana Linares**)

_____

Firma del tutor

(Pascual Verdecia Vicet)



"(…) Educar es todo, educar es sembrar valores, es desarrollar una ética, una actitud ante la vida. (…) Educar es buscar todo lo bueno que pueda estar en el alma de un ser humano (…)"

Fidel Castro Ruz

DATOS DE CONTACTO

Dr. C. Pascual Verdecia Vicet

Graduado de Ingeniero de Minas e Ingeniero Civil, Master en Voladura con Explosivos, Doctor en Ciencias Técnicas Y Profesor de Física con 20 años de experiencia, ha investigado y publicado en las ramas de los explosivos y la pedagogía, ha participado en proyectos relacionados con el empleo de explosivos industriales.

AGRADECIMIENTOS

Con este trabajo quiero agradecerle:

A mis padres y hermano por ayudarme en todo momento con sus consejos y apoyo a salir de los momentos difíciles.

Le agradezco a mi novia por estar junto a mi, comprenderme y amarme tanto.

Agradezco a mi tutor por guiarme en el desarrollo de este trabajo.

Le agradezco a mis compañeros y amigos que han compartido conmigo tantos momentos durante estos años y me han extendido su mano cuando me ha hecho falta.

A todo el claustro de profesores que de una forma u otra haya influido en mi preparación como profesional y que me haya orientado durante toda la carrera.

A Fidel Castro y a la Universidad de las Ciencias Informáticas por permitir que yo me forme como ingeniero en esta carrera durante cinco años.

A todas aquellas personas que de alguna forma ha hecho posible la realización de este trabajo.

DEDICATORIA

Dedico mi tesis, trabajo donde he puesto todo mi esfuerzo y dedicación, a:

Mis padres y hermano por haber depositado toda su confianza en mí y ser mi razón de ser.

A mi novia por ocupar un lugar tan importante en mi corazón y vida.

A mi familia.

Y a todas aquellas personas que se preocupan por mí.

RESUMEN

Actualmente la Informática constituye un factor muy importante para el desarrollo económico y social de todos los países. Como respuesta a la creciente complejidad de las aplicaciones informáticas y a las necesidades de almacenamiento, seguridad y gestión de la información que persiste han surgido en relativamente poco tiempo los Sistemas de Bases de Datos.

La Universidad de las Ciencias Informáticas es la institución académica más joven de Cuba con el objetivo de graduar profesionales de la Informática altamente calificados que repercutan positivamente en la economía del país. En esta institución se aplican las técnicas más avanzadas en la construcción de software.

En los Docentes de la Universidad de las Ciencias Informáticas se afrontan problemas debido a que la gestión de la información de recursos humanos y materiales, se realizan manualmente en su gran mayoría.

Debido a esa problemática la Facultad 3 de la UCI creó un proyecto productivo con el objetivo de implementar un Sitio Web para la misma. Es interés del cliente que la base de datos sea independiente de la aplicación.

Éste trabajo de diploma es el resultado del diseño de una base de datos en PostgreSQL para la Gestión de información de recursos.

En este trabajo de diploma se exponen las tendencias, técnicas y tecnologías que son utilizadas en la actualidad en los Sistemas de Bases de Datos. Después se realiza la descripción y el análisis de la solución propuesta.

ÍNDICE

DECLARACIÓN DE AUTORÍA II

DATOS DE CONTACTO IV

AGRADECIMIENTOS V

DEDICATORIA VI

RESUMEN VII

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA. 4

1.1 INTRODUCCIÓN… 4
1.2 PROCESOS DE DESARROLLO… 4
1.3 METODOLOGÍA DE DESARROLLO DE SOFTWARE… 4
1.3.1 Las metodologías Ágiles *5*
1.3.1.1 Extreme Programing (XP) 6
1.3.1.2 Feature Driven Development (FDD). 6
1.3.2 Las metodologías Tradicionales o Robustas *7*
1.3.2.1 Rational Unified Process (RUP). 7
1.3.2.2 Microsoft Solutions Framework (MSF) 11
1.4 SISTEMAS DE GESTIÓN DE BASE DE DATOS… 11
1.4.1 PostgreSQL *17*
1.4.2 MySQL *20*
1.4.3 SQL Server *21*
1.4.4 Access *22*
1.5 HERRAMIENTAS 24
1.51 EasyCase *24*
1.5.2 ERwin *24*

CAPITULO 2: DESCRIPCIÓN DE LA SOLUCIÓN 27

2.1 INTRODUCCIÓN 27
2.2 DESCRIPCIÓN DE LA ARQUITECTURA DE POSTGRESQL… 27
2.3 REQUISITOS NO FUNCIONALES DE LA BASE DE DATOS… 28
2.4 DISEÑO DE BASE DE DATOS 29
2.4.1 Gestión de la Información de Recursos *29*
2.4.2 Descripción de las interrelaciones. *31*
2.4.3 Descripción de las Tablas. *32*
2.5 SEGURIDAD DE LA BASE DE DATOS… 44

CAPITULO 3: VALIDACIÓN DE LOS DATOS. 43

3.1 INTRODUCCIÓN 43
3.2 INTEGRIDAD RELACIONAL… 43
3.2.1 Tipos de Restricciones de Integridad en base de datos Relacionales… *43*
3.3 NORMALIZACIÓN 55

CONCLUSIONES **56**
RECOMENDACIONES **57**
BIBLIOGRAFÍA **58**
ANEXOS **59**

INTRODUCCIÓN

Con el transcurso del tiempo han surgido grandes avances en la tecnología, eslabón fundamental en el desarrollo actual del mundo. Diferentes son las ramas de la ciencia que se han beneficiado con estos avances. Una de estas ramas beneficiadas, es la informática.

La informática ha sido una de las ramas más importantes del desarrollo socio económico de cualquier país ya que es capaz de resolver problemas existentes en empresas, dándoles rapidez y eficiencia en la gestión de procesos. Para esto se han construido diferentes formas (herramientas, metodologías…) que faciliten el desarrollo de software eficientes y robustos que tributen a dichos fines.

Cuba se ha visto envuelta en los últimos años en un proceso de rectificación y mejoramiento de sus políticas en cuanto a computación y creación de software. Para ello se han creado leyes, organismos, entidades y estructuras que favorecen el mejor desarrollo de la informática.

La magnitud de la revolución en pensamiento informático cubano ha llegado a puntos antiguamente inimaginables para nuestro pueblo, como los de plantearse una informatización del país y, aprovechando al máximo los recursos humanos de que disponemos, llegar a tener la exportación de software, como uno de nuestros renglones económicos.

Es en este periodo donde se inserta la idea de la primera ciudad digital de Cuba: la UCI, que combina la preparación académica y la producción en materia de software, que es en estos momentos la universidad más grande del país.

Esta gran ciudad digital está compuesta por diez facultades en la que cada una, tiene matriculados aproximadamente mil estudiantes vinculados a perfiles específicos. Esto implica el manejo de gran cantidad de recursos tanto humanos como materiales y de otros tipos, estos requieren de un sistema para su desarrollo y control eficiente facilitando el trabajo al personal que manejan estos recursos.

La facultad 3 no está exenta a esta estructura por lo que se propuso automatizar estas necesidades, brindando accesibilidad y optimización. El sistema resultante brindará las facilidades de administrar, controlar y contabilizar los recursos, mostrar la información requerida

acorde a niveles de acceso. Entre otras funcionalidades se pretende realizar búsquedas, que permitan la mejor y más exacta disponibilidad de los datos cuando sean requeridos.

En el mundo la gestión de recursos ha tomado un gran auge debido a la repercusión que ha tenido en la optimización de las funciones empresariales. En este tema existen tendencias actuales muy fuertes como los son Sistemas Integrales de Gestión y entre ellos los denominados ERP, que se encargan del manejo integral de todas las actividades rectoras de funcionamiento en tema de recursos en las empresas.

Existe también sistemas de gestión de recursos humanos que implica el manejo del recurso más preciado de una empresa y como sistema su gestión no es responsabilidad exclusiva de una unidad organizativa especializada, sino en primer lugar implica a los administradores a todos los niveles. Ejemplo de este sistema es *Nexo Digital Huma Nex* es una aplicación de gestión de recursos humanos con una amplia y variada experiencia.

Por su parte Cuba, ha estado trabajando con algunos sistemas de gestión dirigidos principalmente en la gestión de inventarios. Se entiende por Gestión de Inventarios, todo lo relativo al control y manejo de las existencias de determinados bienes, en la cual se aplican métodos y estrategias que pueden hacer rentable y productivo la tenencia de estos bienes y a la vez sirve para evaluar los procedimientos de entradas y salidas de dichos productos. Ejemplo de esto es ASSETS (1997) que cuenta con el 90 por ciento de adaptación a la realidad económica cubana.

Este sistema es muy complejo para aplicarlo para la facultad 3 que maneja esta información de forma interna en la UCI. Esto implica que se desarrolle un software adaptado a las necesidades específicas.

Después que la facultad decide tomar esta situación o de darle solución, se enfrenta a un problema especifico.

Problema

¿Cómo organizar la información de los recursos humanos y materiales de la facultad, facilitando el acceso a los mismos de una forma automatizada?

Objeto de estudio:

Base de datos.

Campo de acción:

Diseño de la base de datos para el sistema de gestión de información de recursos.

Objetivo general:

Diseñar la base de datos que permita la gestión eficiente de la información de los recursos materiales y humanos de la facultad 3.

Se desarrollaron las siguientes tareas:

1. Realizar el estudio del arte referente a los sistemas de bases de datos.
2. Analizar la información obtenida de la investigación del tema, asumiendo posición al respecto.
3. Realizar una valoración de las herramientas y argumentar el uso de la seleccionada.
4. Instalar el software necesario.
5. Confeccionar los modelos lógico y físico de la BD.
6. Normalizar la BD.
7. Definir propuesta de gestores de base datos.

Hipótesis:

Si se obtiene un diseño de base de datos para la gestión de la información de los recursos humanos y materiales de la facultad 3, entonces existirá una organización de los mismos, facilitando el acceso a estos de una forma automatizada.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En este capítulo se abordan diversos temas que son muy importantes para dar cumplimiento de a este trabajo. Estudio y análisis de algunas de las metodologías de desarrollo del software, algunos gestores de base de datos que pueden ser utilizados para el almacenamiento de los datos, así como herramientas a utilizar por el sistema a implementar.

1.2 Procesos de Desarrollo.

En los últimos tiempos la calidad y variedad de los procesos de desarrollo ha aumentado de forma impresionante. Se podría decir que en estos últimos años se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ligeros. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan el objetivo común por medio de orden y documentación los métodos ligeros (también llamados métodos ágiles) tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso.

1.3 Metodología de desarrollo de software.

(Booch, Grady, Jacobson, Ivan y Rumbaugh, James. 2000)

Las metodologías de desarrollo de software surgieron a raíz de la necesidad de controlar, guiar y documentar proyectos cada vez más complejos, impulsadas principalmente por instituciones económicamente importantes y con requisitos de seguridad y fiabilidad en sus sistemas sumamente estrictos.

La experiencia acumulada a lo largo de años de ejecutar proyectos, indica que los proyectos exitosos son aquellos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar al proyecto. Según esta visión, los proyectos que no sigan estos

lineamientos corren un alto riesgo de fracasar. Para esto es necesario el uso de metodologías de desarrollo.

1.3.1 Las metodologías Ágiles

Lo que diferencia un proceso ágil de lo tradicional son sus características o principios, a continuación se relaciona los doce principios generales para las metodologías ágiles.

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV. Los trabajadores del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software que funciona es la medida principal de progreso.

VIII. Promover un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

X. La simplicidad es esencial.

XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Esto es de forma general para todas las metodologías ágiles, donde cada una de ella tiene características propias.

1.3.1.1 Extreme Programing (XP)

(Molpeceres, 2002)

La programación extrema o Extreme Programing (XP) es un enfoque de la ingeniería de software formulado por Kent Beck. Es un modelo de programación, se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

Los objetivos de XP son muy simples. Esta metodología trata de lograr la satisfacción del cliente, dándole el software que él necesita y cuando lo necesita. Otro objetivo de importancia es potenciar al máximo el trabajo en grupo, donde los jefes de proyecto, los clientes y desarrolladores son parte del equipo y están involucrados en el desarrollo del software.

1.3.1.2 Feature Driven Development (FDD).

(Molpeceres, 2002)

FDD es un método ágil, iterativo y adaptativo. A diferencia de otros métodos ágiles, no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción y se considera adecuado para proyectos mayores y de misión crítica.

Los principios de FDD son pocos y simples:

• Se requiere un sistema para construir sistemas si se pretende escalar a proyectos grandes.

• Un proceso simple y bien definido trabaja mejor.

• Los pasos de un proceso deben ser lógicos y su mérito inmediatamente obvio para cada miembro del equipo.

• Vanagloriarse del proceso puede impedir el trabajo real.

• Los buenos procesos van hasta el fondo del asunto, de modo que los miembros del equipo se puedan concentrar en los resultados.

• Los ciclos cortos, iterativos y orientados por features, son mejores.

FDD consiste en un conjunto de “mejores prácticas” que distan de ser nuevas pero se combinan de manera original.

La metodología de desarrollo FDD esta guiada por fases ver figura 1.1.

Fases del Proceso FDD.

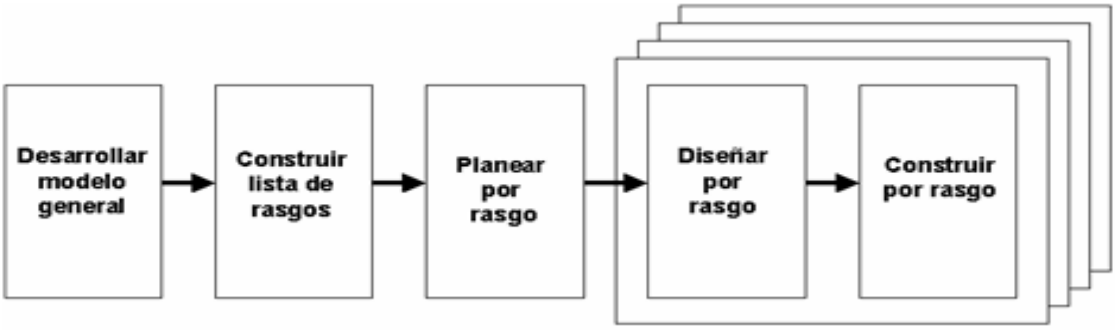


Figura 1.1. Fases del Proceso FDD.

1.3.2 Las metodologías Tradicionales o Robustas

Estas metodologías también son consideradas como fuertes o establecidas, dentro de las cuales se destaca Rational Unified Process (RUP) y Microsoft Solutions Framework (MSF).

1.3.2.1 Rational Unified Process (RUP).

(Molpeceres, 2002)

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

Un proyecto realizado siguiendo RUP se divide en cuatro fases.

1. Intercepción (puesta en marcha)
2. Elaboración (definición, análisis, diseño)
3. Construcción (implementación)
4. Transición (fin del proyecto y puesta en producción)

Las características principales del proceso son:

• Guiado por los Casos de Uso.

• Centrado en la Arquitectura.

• Iterativo e Incremental.

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo de cascada o waterfal para los flujos de trabajo que requieren las nuevas actividades anteriormente citadas.

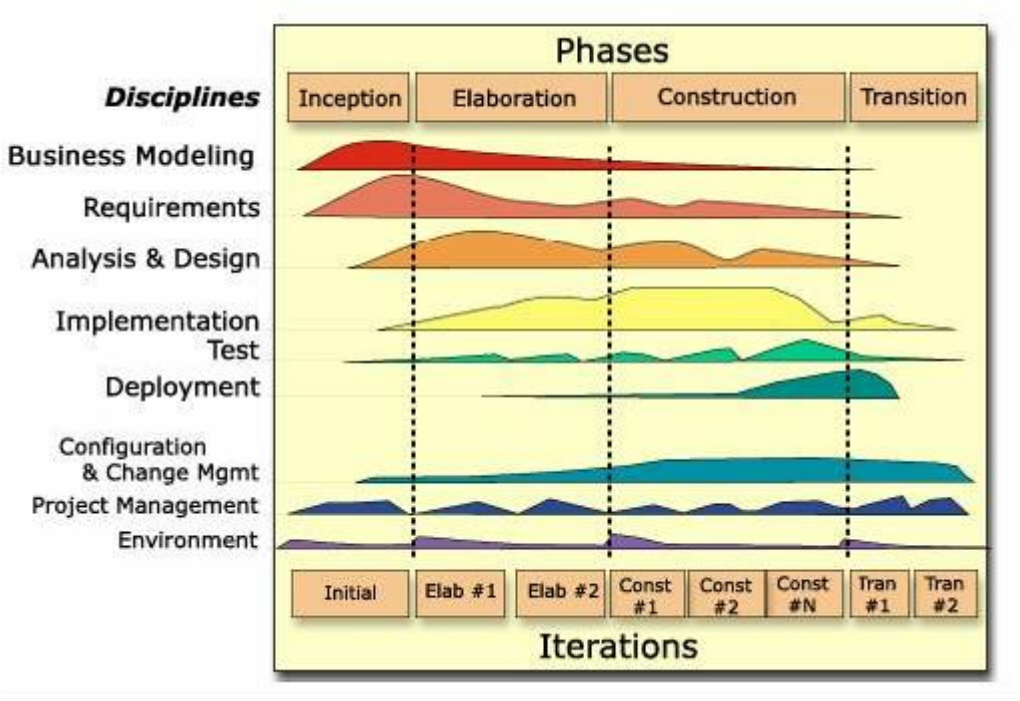


Figura 1.2.Rational Unified Process (RUP).

RUP define nuevas actividades a realizar en cada fase del proyecto

1- Modelo del negocio

2- Análisis de requisitos

3- Análisis y diseño

4- Implementación

5- Test

6- Distribución

7- Gestión de configuración y cambio

8- Gestión del proyecto

9- Gestión del entorno

y el flujo de trabajo (workflow) entre ellas en base a los llamados diagramas de actividad. El proceso define una serie de roles que se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado (artefactos en la jerga de RUP) que se espera de ellos.

RUP se basa en casos de uso para describir lo que se espera del software y esta muy orientado a la arquitectura del sistema, documentándose lo mejor posible, basándose en UML (Unified Modeling Lenguage) como herramienta principal.

En vistas de que el proyecto:

- ✓ Consta de cuatro integrantes.
- ✓ Debe ser suficiente y adecuadamente documentado para que los futuros informáticos que deban perfeccionarlo u agrandarlo tengan una base sólida por donde guiarse.
- ✓ Consta de muchos intereses involucrados (dígase usuarios de la aplicación y clientes directos).
- ✓ No tiene inmediatez en tiempo de entrega.

Y teniendo en cuenta que RUP:

- ✓ Define roles fundamentales que tienen a su cargo la generación de artefactos y documentación correctos por cada fase y flujo que tributan a un buen producto final.
- ✓ Es una metodología establecida y una de las más usadas mundialmente.
- ✓ Esta respaldada por buenos resultados en proyectos en el mundo y en la UCI.
- ✓ Esta diseñada de forma que exista un entendimiento continuo y gradual de todos los implicados en el proyecto.
- ✓ Centra su ciclo de vida en la arquitectura.

Se determinó adoptar esta metodología para guiar el desarrollo del proyecto.

1.3.2.2 Microsoft Solutions Framework (MSF)

MSF es una metodología flexible e interrelacionada con una serie de conceptos, modelos y mejores prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos.
Concretamente MSF se compone de principios, modelos y disciplinas.
Esta metodología se adapta a proyectos de cualquier dimensión y de cualquier tecnología, propone un conjunto de artefactos y entregables que ya están presentes en otras metodologías.

Arribando a conclusiones generales en cuanto por qué usar metodologías tradicionales y dentro de estas (RUP), se puede decir que:

Su utilización es sugerida para proyectos nuevos y para actualizaciones de sistemas ya existentes. El equipo de desarrollo cuenta con 4 estudiantes, por lo que seria conveniente la definición de roles y así tener un mayor grado de control y organización en cuanto a la responsabilidad de las actividades a desarrollar. Es recomendable que el sistema se encuentre lo más documentado posible ya que el equipo de desarrollo del mismo al terminar probablemente tenga que partir a realizar otras labores, por lo que esa documentación le ayudaría muchísimo a otros informáticos que se encargarían del mantenimiento del sistema. El desarrollo del sistema no requiere inmediatez como en el caso de los sistemas que usan las metodologías ágiles.

1.4 Sistemas de gestión de base de datos.

(Maestros del Web, 2007)

Primero que todo una de las tantas definiciones de base de datos es:

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta (figura 1.3).

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

Al igual que cuando se habla de coches, no existe un único modelo, ni una sola marca, ni siquiera una sola tecnología sobre su funcionamiento, cuando se trabaja con bases de datos ocurre una cosa parecida: no existe una sola marca, sino varias, y además cada marca puede tener diferentes productos cada uno de ellos apropiado a un tipo de necesidades.
Las más utilizadas son las bases de datos relacionales, las más antiguas son las jerárquicas y en red, y las más avanzadas son las orientadas a objetos, y las declarativas. Estas se diferencian como hemos dicho, en la forma de trabajar con los datos y en la concepción o mentalidad que el usuario debe adoptar para interactuar con el sistema.

Seguidamente una breve explicación de algunos tipos de base de datos:

Modelo jerárquico.

Éstas son BD que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Las BD jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

Modelo relacional.

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de BD. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las BD relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la BD.

La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. El lenguaje más habitual para construir las consultas a BD relacionales es SQL, Structured Quero Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de BD relacionales.

Durante su diseño, una BD relacional pasa por un proceso al que se le conoce como normalización de una BD. Durante los años '80 (1980-1989) la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

Modelo orientado a objetos.

Actualmente, la creación de programas más grandes y complejos, ha hecho avanzar los métodos de programación hacia nuevas formas que permiten el trabajo en equipo de una forma más eficaz y en la que se disminuyen los problemas de coordinación. Uno de estos métodos consiste en la programación orientada a objetos (POO), que trata los problemas

desde un punto de vista realista, y modelando cada uno de ellos como si se tratase de un conjunto de elementos u objetos que interrelacionan entre sí para solucionar el problema.

Las Bases de datos orientados a objetos se propusieron con la idea de satisfacer las necesidades de las aplicaciones más complejas. El enfoque orientado a objetos ofrece la flexibilidad para cumplir con algunos de estos requerimientos sin estar limitado por los tipos de datos y los lenguajes de consulta disponibles en los sistemas de bases de datos tradicionales.

Como cualquier Bases de Datos programable, una Base de Datos Orientada a Objetos (BDOO) proporciona un ambiente para el desarrollo de aplicaciones y un depósito persistente listo para su explotación. Una BDOO almacena y manipula información que puede ser digitalizada (presentada) como objetos, además proporciona un acceso ágil y permite una gran capacidad de manipulación.

Los principales conceptos que se utilizan en las Bases de Datos Orientada a Objetos (BDOO) son las siguientes:

- Identidad de objetos
- Constructores de tipos
- Encapsulamiento
- Compatibilidad con los lenguajes de programación
- Jerarquías de tipos y herencia
- Manejo de objetos complejos
- Polimorfismo y sobrecarga de operadores y
- Creación de versiones.

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información.** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia. La independencia** de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Redundancia mínima.** Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia.** En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad.** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad.** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo y recuperación.** Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Control de la concurrencia.** En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- **Tiempo de respuesta.** Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

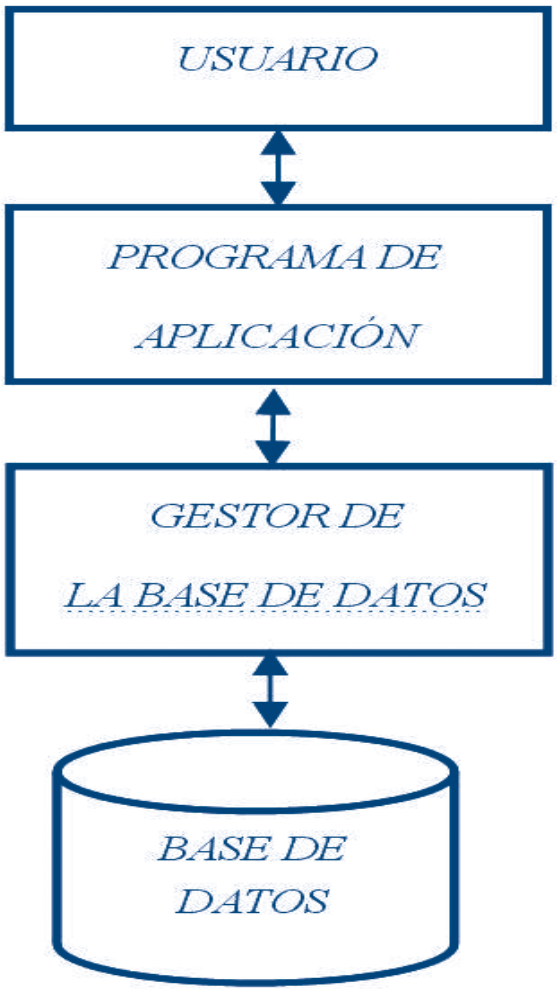


Figura 1.3. Estructura de un sistema de base de datos.

Sus principales ventajas son:

1. Facilidad de manejo de grandes volúmenes de información.
2. Gran velocidad en muy poco tiempo.
3. Independencia del tratamiento de información.
4. Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
5. No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
6. Integridad referencial el terminar los registros.

Inconvenientes:

1. El costo de actualización del hardware y software son muy elevados.
2. El Costo (salario o remuneración) del administrador de la base de datos es grande.
3. El mal diseño de esta puede originar problemas a futuro.
4. Un mal adiestramiento a los usuarios puede originar problemas a futuro.
5. Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
6. Generan campos vacíos en exceso.
7. El mal diseño de seguridad genera problemas en esta.

1.4.1 PostgreSQL

(PosrgresSQLPE, 2007)

PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. Este proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle.

PostgreSQL es un magnífico gestor de bases de datos. Tiene prácticamente todo lo que tienen los gestores comerciales. A pesar de ello, el primer encuentro con este gestor es un poco "duro", ya que la sintaxis de algunos de sus comandos no es nada intuitiva. También

resulta engorroso las pequeñas variaciones que presenta este gestor en algunos de los tipos de datos que maneja, siendo el problema más comentado el referente al tipo "serial".

Una vez nos hayamos hecho con su sintaxis y fijándonos en estos pequeños detalles (que por otro lado están totalmente documentados), PostGreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios Web que posean alrededor de 500.000 peticiones por día.

Disponibles bajo sistema operativo Windows o Linux. PostGreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. 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 es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostGreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Características:

1. Funciones.

Las funciones permiten subir bloques de código que se ejecuten en el servidor. Estas funciones pueden escribirse en una variedad de lenguajes, algunos de los más importantes son PL/pgSQL, C, C++ y Java.

Puede definirse si las funciones serán ejecutadas con los permisos del llamador o del usuario que definió la función.

2. Alta concurrencia.

Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se

le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

3. Amplia variedad de tipos nativos.

PostgreSQL provee nativamente soporte para:

- Números de de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geómetricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arreglos.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

4. Otras características.

- Cumple completamente con ACID.
- Cumple con ANSI SQL.
- Llaves Foráneas (foreign keys).
- Disparadores (triggers).
- Reglas.
- Vistas.
- Unicode.
- Secuencias.
- Outer Joins.
- Sub-selects.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

- Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica.
- Una API abierta.
- Interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby.
- Soporte nativo SSL.
- Respaldo en caliente.
- Bloqueo a nivel mejor-que-fila.
- Índices parciales y funcionales.
- Autentificación Kerberos nativa.
- Soporte para consultas con UNION, UNION ALL y EXCEPT.
- Extensiones para SHA1, MD5, XML y otras funcionalidades.
- Herramientas para generar SQL portable para compartir con otros sistemas compatibles con SQL.
- Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.

1.4.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB.

Disponibles bajo sistema operativo Windows o Linux. Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Características:

Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.

Soporta gran cantidad de tipos de datos para las columnas.

Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).

Gran portabilidad entre sistemas.

Soporta hasta 32 índices por tabla.

Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

1.4.3 SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Así de tener unas ventajas que a continuación se pueden describir.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE, PostgreSQL o MySQL.

Características de Microsoft SQL Server

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.

- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en su versión 2005 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita.

Es muy común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa una potente base de datos (Microsoft SQL Server), con un entorno de desarrollo cómodo y de alto rendimiento (VBA Access), a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows.

Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor solo está disponible para Sistemas Operativos Windows.

1.4.4 Access

Microsoft Access es un programa Sistema de gestión de base de datos relacional creado y modificado por Microsoft para uso personal de pequeñas organizaciones. Es un componente de la suite Microsoft Office aunque no se incluye en el paquete "básico". Una posibilidad adicional es la de crear ficheros con bases de datos que pueden ser consultados por otros programas. Disponibles sólo bajo sistema operativo Windows.

Para que la aplicación sobre base de datos Access no tenga problemas, es recomendable que cumpla estas condiciones:

- El volumen de datos a manejar es pequeño. (Además así será más rápida su actualización por FTP).
- El número de visitantes simultáneos no es muy alto.

La aplicación ASP no cambia la base de datos, simplemente muestra datos. Esto es consistente con el hecho de enviar periódicamente el archivo .mdb al servidor, pues si la aplicación ASP cambiase la base de datos, esos cambios se perderían al sobrescribirse con la nueva base de datos.

Criterios	Bases de datos			
	Access	SQL Server	MySQL	PostgreSQL
Plataforma			/	/
Velocidad	-	✓	✓	-
Volumen Datos	-	✓	✓	✓
Integridad	-	✓	-	✓
Potencia	-	✓	✓	✓
Coste/MB	✓	-	✓	✓

✓Positivo - Negativo

Figura 1.4. Comparativa entre Bases de datos

Luego de un análisis se determino que el PostgresSQL será el gestor de base de datos que se utilizará, por sus características y ventajas que brinda (figura 1.4):

- Instalación ilimitada.
- Extensible (el código esta disponible para todos sin costo).
- Multiplataforma (34 plataformas en la última versión estable).
- Diseñado para ambientes de alto volumen (PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes).

1.5 Herramientas

La utilización de una herramienta CASE (Computer-Aided Software Engineering) ha sido uno de los aportes más importante en el campo de la Ingeniería de software.

Aunque el concepto fue abordado desde hace tres décadas, son los productos que se encuentran hoy en el mercado los que han provocado una revolución por el aumento que propician en la eficiencia, productividad y calidad del producto final.

1.51 EasyCase

EasyCASE Profesional es un producto destinado a la generación de esquemas de base de datos e ingeniería reversa. Su objetivo es proporcionar a los usuarios una solución comprensible para el diseño, documentación y consistencia del sistema que se va a desarrollar. Brinda la posibilidad de automatizar las primeras fases del desarrollo de un software (análisis y diseño), para lograr el desarrollo de un trabajo óptimo y eficaz. Forma parte del grupo de herramientas multi-usuarios, permitiendo que varias personas trabajen en un proyecto al mismo tiempo. Esta herramienta permite la conexión de varias personas a un servidor donde está la información que necesitan los integrantes del proyecto para su trabajo. Esta conexión se hará de forma segura, ya que la herramienta controlará el nivel de acceso que podrá tener cada usuario, según el rol que desempeñe, a través del diagrama y diccionario de los datos que bloquean por niveles al registro, al archivo y al proyecto. EasyCASE soporta las siguientes base de datos: Oracle, Postgres, SQL Server, ANSI SQL, Fox Pro, SQL Base, Access, Paradox, Sybase, Clipper, entre otras.

1.5.2 ERwin

PLATINUM ERwin es una CASE empleada en el diseño de base de datos, con la cual se puede hacer un trabajo productivo por las facilidades que brinda para la generación y mantenimiento de aplicaciones de forma sencilla para el diseñador. ERwin permite realizar el diseño del modelo lógico de los requerimientos de información, así como el diseño del modelo físico, ya con nivel mayor, refinando las características de la base de datos diseñada. Con esta herramienta CASE es posible visualizar la estructura de la base de datos diseñada, lo

cual tiene como ventaja que el diseñador pueda observar en su totalidad el trabajo realizado, para realizar un análisis y si fuera necesario hacer cambios en busca de optimizar el diseño final. Esta herramienta permite generar, de forma automática, las tablas y el código referente a los stored procedure (procedimientos almacenados) y triggers para los principales tipos de bases de datos. La migración automática garantiza la integridad referencial de la base de datos, es decir, evita la pérdida de información durante este proceso. ERwin puede hacer una conexión entre dos bases de datos, una diseñada y otra sin diseño, estableciendo una transferencia entre ellas y la aplicación de ingeniería reversa. Mediante esta conexión puede automáticamente generar índices, vistas, tablas, reglas de integridad referencial (llaves foráneas, llaves primarias), restricciones de dominios y campos y valores por defecto. ERwin básicamente soporta bases de datos del tipo relacionales SQL, y es compatible con otras como: Oracle, Microsoft SQL Server, Sybase, Microsoft Access, dBase, FoxPro, Informix, SQL Base y SQL Anyware, entre otras. Con un mismo modelo es posible generar múltiples bases de datos o hacer una conversión de una aplicación de una base de datos a otra. Es compatible con los sistemas operativos de la familia Windows, como Windows 95, Windows 98, Windows XP y Windows NT.

Luego un determinado análisis se determino que la herramienta a utilizar es el ERwin:

- En nuestro proyecto desarrollamos la base de datos en un gestor relacional, por lo que es conveniente el uso del Erwin puesto que esta herramienta modela el diseño de las bases de datos de modo relacional.
- En nuestro proyecto, para lograr un correcto diseño de base de datos, es necesario realizar un diseño lógico, y luego a partir de este se obtener un diseño físico de la base de datos. Esto te permite lograr que el diseño de la base de datos tenga en cuenta todos los detalles referentes al negocio que se esta modelando.
- La herramienta Erwin permite el diseño y construcción de bases de datos a nivel lógico y físico permitiendo la sincronización bidireccional entre ambos diseños, lo que posibilita que si se hace algún cambio en uno de los dos diseños antes mencionados, se refleje de forma automática en el otro.

- Como en nuestra base de datos se manipulan grandes volúmenes de información pues es conveniente utilizar la herramienta CASE Erwin debido a que esta permite la generación automática de procedimientos almacenados que ayudaran a agilizar el trabajo con la base de datos.

- Erwin está equipado para crear y manejar diseños de bases de datos funcionales y confiables.

- Ofrece la posibilidad de construir automáticamente bases de datos y documentación basada en HTML.
- Permite la creación de reportes de forma sencilla.
- Da la posibilidad de hacer reingeniería inversa de bases de datos.

Capitulo 2: Descripción de la Solución

2.1 Introducción

En este capítulo se abordan diferentes temas, los cuales le darán solución a la Gestión de Información, siendo este él más importante de este trabajo. Aquí se encontrará los requerimientos no funcionales, la arquitectura de PostgresSQL, modelo lógico y modelo físico de la base de datos y las descripciones de las tablas e interrelaciones entre otras cosas más.

Las herramientas de diseño utilizadas son el ER EStudio 7.1 para los modelos de base de datos y como gestor de la misma PostgreSQL 8.2.

2.2 Descripción de la Arquitectura de PostgreSQL.

La arquitectura de PostgreSQL es de 3 niveles. Esta se corresponde suficientemente bien con un gran número de sistemas, aunque no se puede asegurar que cualquier SGBD se corresponda exactamente con ella.

La arquitectura de PostgreSQL (figura 2.1) se divide en los siguientes niveles generales: interno, lógico global y externo.

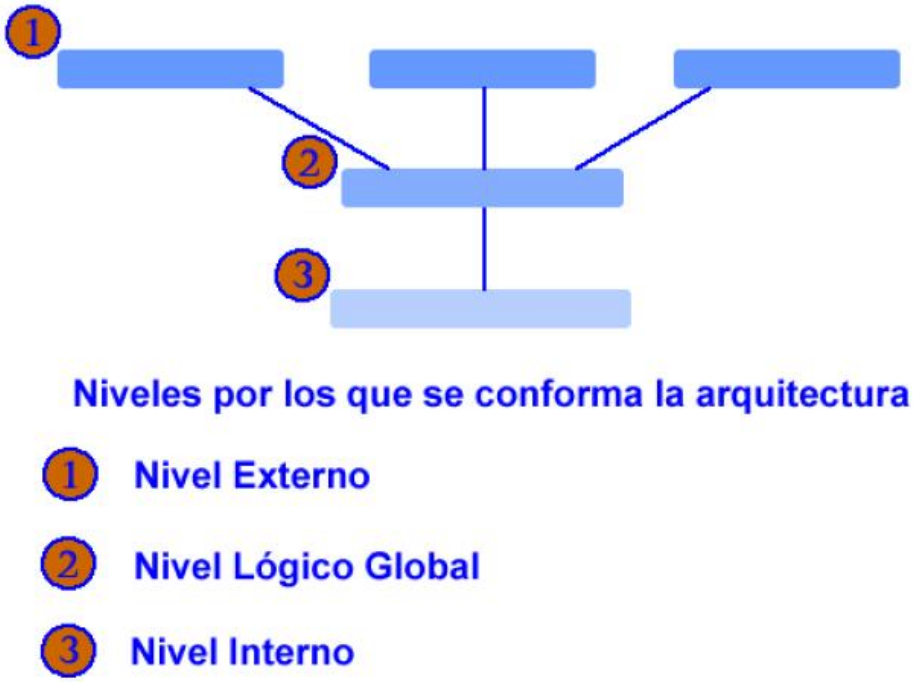


Figura 2.1. Arquitectura de PostgreSQL.

El nivel interno es el más cercano al almacenamiento físico, o sea, es el relacionado con la forma en que los datos están realmente almacenados.

El nivel externo es el más cercano a los usuarios, o sea, es el relacionado con la forma en que los datos son vistos por cada usuario individualmente.

El nivel lógico global es un nivel intermedio entre los dos anteriores.

2.3 Requisitos no Funcionales de la Base de Datos.

1. La base de datos deberá ser independiente de la aplicación.

2. El servidor será montado en una P4 con procesador de 256 MB de RAM como mínimo.

3. La BD deberá tener una gran disponibilidad y confiabilidad en cuánto a los datos que almacena.

4. Se deberá reducir el tiempo de respuesta a las peticiones por parte de los usuarios a la BD.

5. Se deberá trazar una estrategia de mantenimiento de la BD como norma de soporte.

6. La información manejada por la BD debe estar protegida de acceso no autorizado y divulgación.

7. La información manejada por la BD será objeto de cuidadosa protección contra la corrupción y estados inconsistentes.

2.4 Diseño de Base de datos

Para el Sistema de Gestión Recursos se desarrolló el diseño de un módulo. Este módulo se va a encargar del almacenamiento de los datos de los profesores, estudiantes, para docentes y de los medios que se encuentran en cada local.

A continuación podemos encontrar la descripción del diseño de la base de datos.

2.4.1 Gestión de la Información de Recursos

Este diseño de gestión de la información de recursos va tener almacenado todos los medios de la facultad, de los cuales se conoce: nombre, estado, cantidad, fecha de entrada y numero de serie, tipo de medio, fecha en la que fue ingresado, estado del medio, que a su vez, cada medio tiene reportes que de estos se conoce fecha, información del reporte. Donde los medios se encuentran en locales que de estos se conoce el nombre del local.

También va tener almacenado personas que de estos se conoce el solapín, integración revolucionaria, nombre y apellidos, fecha de ingreso y dirección particular. Las personas están divididas en tres tipos:

Para docentes: que tiene nivel de escolaridad, evaluación mensual y categoría laboral.

Profesor: que tiene categoría científica, categoría docente, dirección UCI, vinculación a proyecto, teléfono UCI, teléfono particular, cargo que ocupa, nombre de la madre y el padre.

Estudiante: que se conoce el año, grupo y facultad en que se encuentra.

Junto con esto, también se incluyen disciplinas (nombre de la disciplina), tesis (título y fecha de tesis), asignaturas (nombre de la asignatura), evolución de tesis (descripción y evaluación de tesis).

También se tendrá un historiar de personas (carnet de identidad, nombre, fecha en la que fue ingresado a la facultad, fecha en la que causo baja o fue trasladado, que fue lo que le ocurrió si fue traslado o baja, causa, integridad revolucionaria y el tipo de persona que era (estudiante, profesor, para docente)) es decir, aquí se encontraran las personas que han causado baja o que se han trasladado a otro lugar que no le pertenece la facultad y de medios (número de serie, fecha en la que ingreso el medio, fecha de la operación, tipo de operación, tipo de medio que es) que pasa lo mismo que en persona, se encontrara el historial del medio (figura 2.2).

De acuerdo a lo anterior se propone el siguiente modelo lógico de la base de datos:

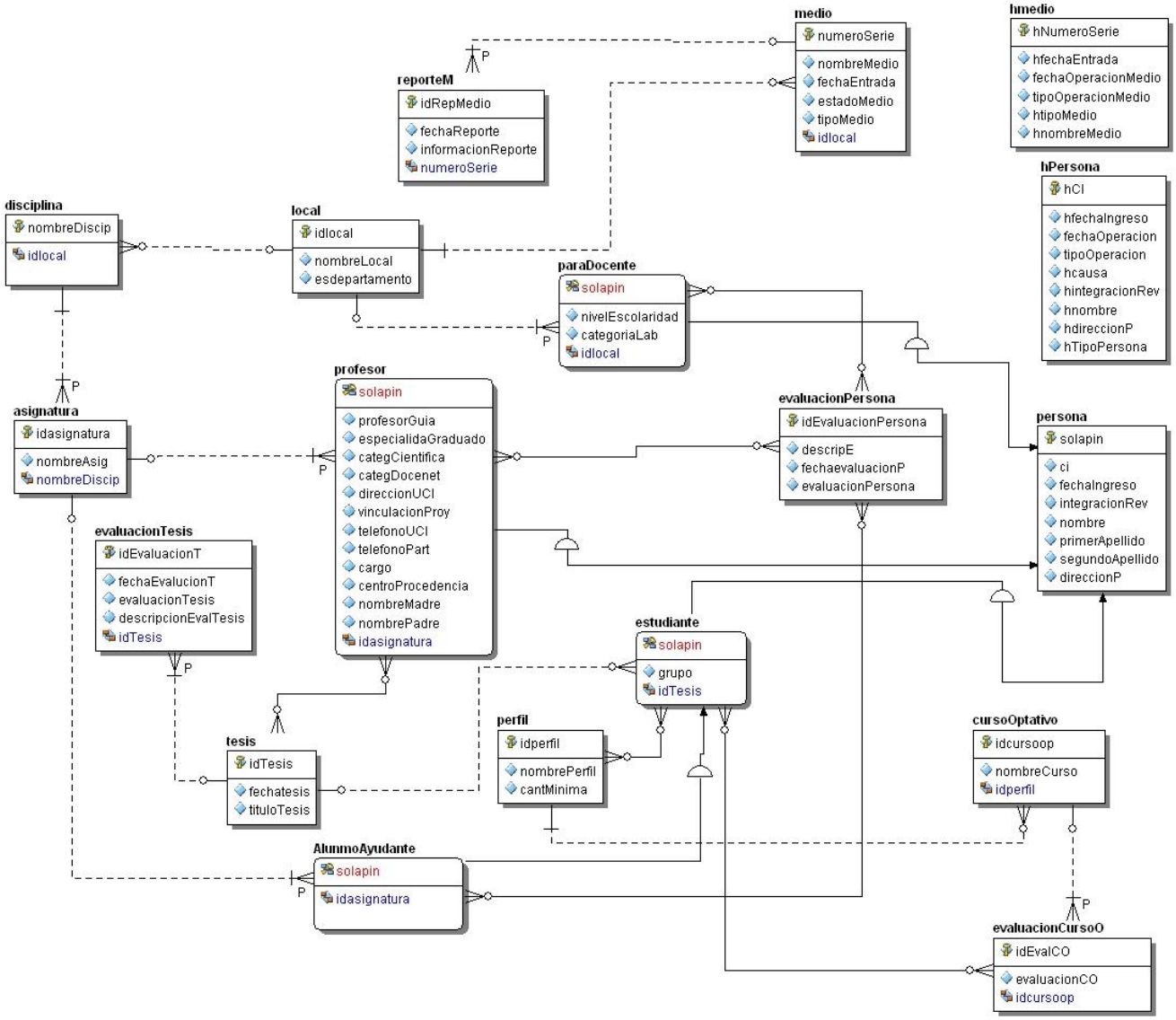


Figura 2.2. Modelo lógico para Gestión de Recursos.

2.4.2 Descripción de las interrelaciones.

Las interrelaciones del modelo anterior obedecen a que una persona puede ser definida de tres tipos: estudiante, profesor y para docente. Donde el para docente, el profesor y el alumno ayudante van a tener una o más evaluaciones y el profesor va a tener tutoría en una o muchas tesis. También un local tendrá muchos medios donde radicara un grupo de personal

para docente, que un departamento estará formado por disciplinas y a su vez por asignaturas y estas las impartirán los profesores. También un perfil tendrá un grupo de cursos optativos y a su vez estos tendrán evaluaciones (figura 2.3).

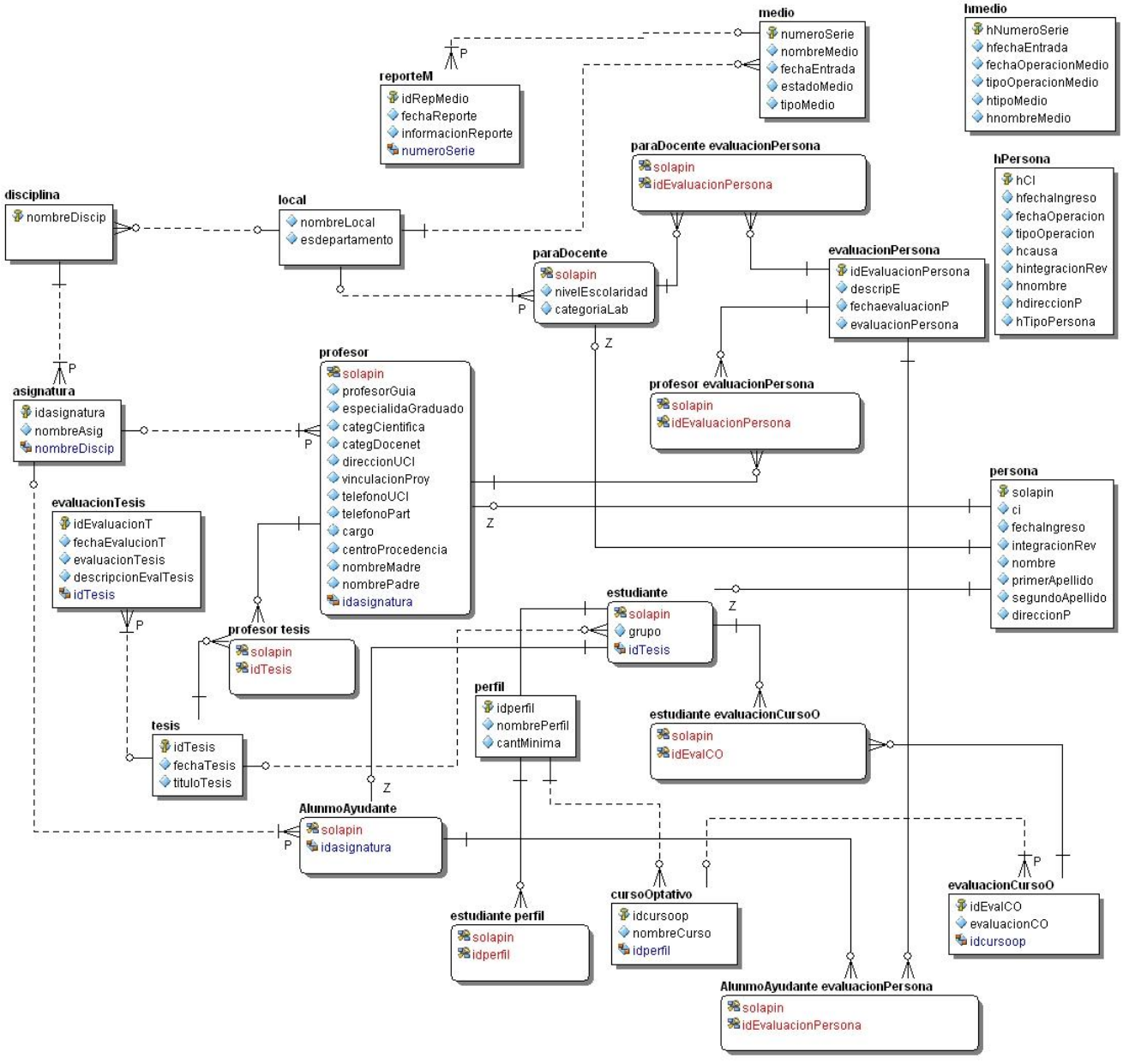


Figura 2.3. Modelo físico para Gestión de Recursos.

2.4.3 Descripción de las Tablas.

Tabla 1 Descripción de la tabla persona.

Nombre	persona			
Descripción	Almacena los datos de la persona			
Llave	Columna	Dato	Requerido	Descripción
PK	solapin	char(7)	Not Null	Número que identifica a la persona
	integracionRev	nvarchar(20)	Null	Integración revolucionaria
	nombre	nvarchar(25)	Not Null	Nombre da la persona
	primerApellido	nvarchar(25)	Not Null	Primer apellido de la persona
	segundoApellido	nvarchar(25)	Not Null	Segundo apellido de la persona
	direccionP	nvarchar(150)	Null	Dirección particular de la persona
	fechaIngreso	date	Null	Fecha de ingreso al centro
	CI	nvarchar (11)	Not Null	Carné de identidad

Tabla 2 Descripción de la tabla paradocente.

Nombre	paraDocente			
Descripción	Almacena los datos del para docente			
Llave	Columna	Dato	Requerido	Descripción
PK	solapin	char(7)	Not Null	Número de solapín que lo identifica
	nivelEscolaridad	nvarchar(50)	Null	Nivel de estudio que tiene
	categoriaLab	nvarchar(100)	Null	Categoría laboral
FK	nombreLocal	nvarchar(50)	Null	Llave foránea de ¨local¨

Tabla 3 Descripción de la tabla profesor.

Nombre	profesor			
Descripción	Almacena los datos de profesor			
Llave	Columna	Dato	Requerido	Descripción
PK	solapin	char(7)	Not Null	Número de solapín que lo identifica
	especialidadGraduado	nvarchar(50)	Null	Especialidad de graduado
	centroProcedencia	nvarchar(50)	Null	Centro de procedencia
	profesorGuia	char(2)	Null	Si es profesor

				guía
	categCientifica	nvarchar(30)	Null	Categoría científica
	categDocenet	nvarchar(50)	Null	Categoría docente
	direccionUCI	nvarchar(255)	Null	Dirección UCI
	vinculacionProy	char(1)	Null	Si esta vinculado a proyecto
	telefonoUCI	nvarchar (30)	Null	Teléfono UCI
	telefonoPart	integer	Null	Teléfono particular
	cargo	nvarchar(100)	Null	Cargo que ocupa
	nombreMadre	nvarchar(50)	Null	Nombre de la madre
	nombrePadre	nvarchar(50)	Null	Nombre del padre
FK	nombreAsig	nvarchar(50)	Null	Lave foránea de ¨asignatura¨

Tabla 4 Descripción de la tabla estudiante.

Nombre	estudiante			
Descripción	Estudiante de la facultad			
Llave	Columna	Dato	Requerido	Descripción
PK	solapin	char(7)	Not Null	Número de solapín que lo identifica
	grupo	nvarchar(5)	Not Null	Grupo donde se encuentra el estudiante
FK	idTesis	nvarchar(10)	Null	Llave foránea de ¨tesis¨

Tabla 5 Descripción de la tabla medio.

Nombre	medio			
Descripción	Almacena los datos del medio			
Llave	Columna	Dato	Requerido	Descripción
PK	numeroSerie	nvarchar(20)	Not Null	Identificador de medio
	fechaEntrada	date	Not Null	Fecha de entrada del medio
	nombreMedio	nvarchar(50)	Not Null	Nombre del medio
	estadoMedio	nvarchar(50)	Not Null	Estado del medio
	tipoMedio	nvarchar(10)	Null	El tipo de medio
FK	nombreLocal	nvarchar(50)	Not Null	Llave foránea de ¨local¨

Tabla 6 Descripción de la tabla reportem.

Nombre	reportem			
Descripción	Almacena los reportes del medio			
Llave	Columna	Dato	Requerido	Descripción
PK	idrepmedio	serial	Identity	Identificador de la clase reporte
	fechareporte	date	Not Null	Fecha en que se hiso el reporte
	informacionreporte	nvarchar	Not Null	Descripción del reporte
FK	numeroserie	nvarchar	Not Null	Llave foránea de ¨medio¨

Tabla 7 Descripción de la tabla evaluacionTesis.

Nombre	evaluacionTesis			
Descripción	Almacena la evaluación que tiene la tesis			
Llave	Columna	Dato	Requerido	Descripción
PK	idEvaluacionT	serial	Identity	Identificador de evaluación de tesis
	evaluacionTessi	integer	Not Null	Evaluación de tesis
	descriccionEvalTesis	nvarchar(255)	Null	Descripción de la evaluación de tesis
	fechaEvaluacionT	date	Not Null	Fecha de evaluación
FK	idTesis	nvarchar(10)	Null	Llave foránea de ¨tesis¨

Tabla 9 Descripción de la tabla local

Nombre	local			
Descripción	Almacena los datos del local			
Llave	Columna	Dato	Requerido	Descripción
PK	idlocal	serial	Identity	Identificador de local
	nombreLocal	nvarchar(50)	Not Null	Nombre del local
	esdepartamento	char(2)	Not Null	Si es departamento o no

Tabla 10 Descripción de la tabla disciplina.

Nombre	disciplina			
Descripción	Almacena los datos de disciplina			
Llave	Columna	Dato	Requerido	Descripción
PK	iddiscip	serial	Identity	Identificador de disciplina
	nombreDisip	nvarchar(50)	Not Null	Nombre de disciplina e identificador
FK	nombreLocal	nvarchar(50)	Not Null	Lave foránea de ¨local ¨

Tabla 11 Descripción de la tabla evaluacionPersona.

Nombre	evalucionPersona			
Descripción	Almacena la evaluación que tiene la persona			
Llave	Columna	Dato	Requerido	Descripción
PK	idEvalucionPersona	serial	Identity	Identificador de evaluación de persona
	evaluacionPersona	char(10)	Not Null	Evaluación que obtiene la persona
	fechaEvaluacionP	date	Not Null	Fecha de la evaluación
	descripcionEval	nvarchar(255)	Null	Descripción de evaluación
FK	solapin	char(7)	Not Null	Llave foránea de ¨ profesor ¨ y ¨ paraDocente ¨

Tabla 12 Descripción de la tabla perfil.

Nombre	perfil			
Descripción	Perfil que la facultad			
Llave	Columna	Dato	Requerido	Descripción
PK	idperfil	serial	Identity	Identificador de perfil
	nombrePerfil	nvarchar(50)	Not Null	Nombre del perfil e identificador de perfil
	cantMinima	integer	Not Null	Cantidad mínima

Tabla 13 Descripción de la tabla cursiOptativo.

Nombre	cursoOptativo			
Descripción	Curso que ha pasado el estudiante			
Llave	Columna	Dato	Requerido	Descripción
PK	idcursoop	serial	Identity	Identificador de curso optativo
	nombreCurso	nvarchar(50)	Not Null	Nombre del curso optativo e identificador del curso optativo
FK	nombrePerfil	nvarchar(50)	Not Null	Llave foránea de ¨perfil¨

Tabla 14 Descripción de la tabla hmedio.

Nombre	Hmedio			
Descripción	Almacena el historial de los medios			
Llave	Columna	Dato	Requerido	Descripción
PK	hidMedio	serial	Identity	Identificador de historial de medio
	hfechaEntrada	date	Not Null	Fecha de entrada del medio
	fechaOperacionMedio	date	Not Null	
	tipoOperacionMedio	nvarchar(18)	Not Null	
	htipoMedio	nvarchar(10)	Null	El tipo de medio
	hnombreMedio	char(50)	Not Null	Nombre del medio

Tabla 15 Descripción de la tabla alumnoayudante.

Nombre	Alumnoayudante			
Descripción	Almacena los datos del reporte del medio			
Llave	Columna	Dato	Requerido	Descripción
PK	solapin	char(7)	Not Null	Número de solapín que lo identifica
	asignaturaimparte	nvarchar(25)	Not Null	Asignatura que imparte

Tabla 16 Descripción de la tabla evaluacionCursoO.

Nombre	evaluacionCursoO			
Descripción	Evaluación del curso optativo			
Llave	Columna	Dato	Requerido	Descripción
PK	idEvalCO	serial	Identity	Identificador de curso optativo
	evaluacionCO	char(10)	Not Null	Evaluación del curso optativo
FK	idCursoOptativo	integer	Null	Llave foránea de ¨ cursoOptativo¨

Tabla 17 Descripción de la tabla hpersona.

Nombre	Hpersona			
Descripción	Almacena el historial de las personas			
Llave	Columna	Dato	Requerido	Descripción
PK	idPersona	serial	Identity	Identificador de historial de persona
	hfechaIngreso	date	Not Null	
	fechaOperacion	date	Not Null	
	tipoOperacion	nvarchar(18)	Null	
	hcausa	nvarchar(255)	Not Null	
	hintegracionRev	nvarchar(20)	Null	
	hnombre	nvarchar(50)	Not Null	
	hdireccionP	nvarchar(150)	Null	
	hTipoPersona	nvarchar(50)	Not Null	

Tabla 18 Descripción de la tabla asignatura.

Nombre	Asignatura			
Descripción	Evaluación del curso optativo			
Llave	Columna	Dato	Requerido	Descripción
PK	nombreAsig	nvarchar(50)	Not Null	Nombre de la asignatura e Identificador de la asignatura
FK	nombreDiscip	nvarchar(50)	Not Null	Llave foránea de ¨ disciplina¨

Tabla 19 Descripción de la tabla tesis.

Nombre	Tesis			
Descripción	Almacena los datos de la tesis			
Llave	Columna	Dato	Requerido	Descripción
PK	idTesis	nvarchar(10)	Not Null	Número de serie de e identificador de la tesis
	fechaTesis	date	Not Null	Fecha de la tesis
	tituloTesis	nvarchar(100)	Not Null	Titulo de la tesis

2.5 Seguridad de la Base de Datos.

(hospedajeydominios.com, 2005)

Hoy en día, las bases de datos son componentes cardinales de cualquier aplicación basada en web, permitiendo que los sitios web provean contenido dinámico. Debido a que información considerablemente sensible o secreta puede ser almacenada en tales bases de datos, se debe considerar seriamente la forma de protegerlas.

La aplicación nunca debería conectarse a la base de datos bajo el usuario correspondiente a su dueño, o como un super_usuario (usuario con acceso total a la base de datos), ya que estos usuarios pueden, por ejemplo, ejecutar cualquier consulta a su antojo, modificando el esquema (ejemplo: eliminando tablas) o borrando su contenido completo.

Para esto se crearon diferentes usuarios de la base de datos para cada aspecto de la aplicación con derechos muy limitados sobre los objetos de la base de datos. A estos usuarios se les otorgaron privilegios estrictamente necesarios, evitando que el mismo usuario pueda interactuar con la base de datos en diferentes casos de uso. Esto quiere decir que si un intruso gana acceso a la base de datos usando una de éstas credenciales, él solo puede efectuar tantos cambios como la aplicación se lo permita.

Para garantizar la disponibilidad de la información que se encuentra almacenada en la base de datos, se creó una tarea programada del sistema, la cual permitirá un respaldo (backup) automático de la base de datos cada 24 horas.

Capitulo 3: Validación de los Datos.

3.1 Introducción

En éste capítulo se realiza la descripción de la integridad relacional de la base de datos que se propone, así como un análisis sobre la normalización y redundancia de la información que la conforma. También se hace referencia a aspectos que están relacionados en la validación funcional de los modelos mediante el llenado de la base de datos y la aplicación adecuada de índices para lograr incrementar el rendimiento del SGBD en la ejecución de las consultas SQL más frecuentes.

3.2 Integridad Relacional.

La implementación de la integridad relacional en una BD consiste en establecer las reglas de consistencia correspondientes a los datos requeridos de las tablas, el chequeo de los valores válidos y únicos, así como la integridad de las llaves primarias y foráneas.

3.2.1 Tipos de Restricciones de Integridad en base de datos Relacionales.

Datos Requeridos: Establece que una columna tenga un valor no nulo. Se define efectuando la declaración de una columna es NOT NULL cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.

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

Integridad de entidad: Establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla, sino la base de datos perderá su integridad. Se especifica en la sentencia CREATE TABLE. El SGBD comprueba automáticamente la unicidad del valor de la

clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente fallará.

Integridad referencial: Asegura la integridad entre las claves ajenas y primarias (relaciones padre/hijo).

A continuación se realiza la descripción de dichas reglas de integridad para los diseños de BD que se propusieron.

Tabla 20 Validación de la tabla persona.

Tabla	Persona				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
solapin	✓	PK			
integracionRev					
nombre	✓				
primerApellido	✓				
segundoApellido	✓				
direccionP					
fechaIngreso	✓				
ci	✓				
Columnas Unitarias	solapin, ci				
Chequeo			Descripción		
fechaIngreso <= CURRENT_DATE			La fecha ingreso de una persona tiene que ser menor o igual que la fecha actual.		

Tabla 21 Validación de la tabla profesor.

Tabla	profesor				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
solapin	✓	PK			
especialidadGraduado					
centroProcedencia					
profesorGuia					
categCientifica					
categDocenet					
direccionUCI					
vinculacionProy					
telefonoUCI					
telefonoPart					
cargo					
nombreMadre					
nombrePadre					
nombreAsig		FK	asignatura	Cascade	Restrict
Columnas Unitarias	Solapin				
	nombreAsig				

Chequeo	Descripción
(profesorguia = 'No') OR (profesorguia = 'Si')	El profesor debe ser guía si o no
((vinculacionproy = 'No') OR (vinculacionproy = 'Si')	El profesor puede estar vinculado a proyecto si o no

Tabla 22 Validación de la tabla paradocente.

Tabla	Paradocente				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
solapin	✓	PK			
nivelEscolaridad					
categoriaLab					
nombreLocal		FK	local	Cascade	Restrict
Columnas Unitarias	Solapin				
	nombreLocal				

Tabla 23 Validación de la tabla asignatura.

Tabla	Asignatura				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
nombreAsig	✓	PK			
nombreDiscip	✓	FK	disciplina	Cascade	Restrict
Columnas Unitarias	nombreAsig				
	nombreDiscip				

Tabla 24 Validación de la tabla evaluacionCursoO.

Tabla	evaluacionCursoO				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idEvalCO	✓	PK			
evaluacionCO					
idCursoOptativo	✓	FK	cursoOptativo	Cascade	Restrict
Columnas Unitarias	idEvalCO				
	idCursoOptativo				
Chequeo			Descripción		
(evaluacionCO>=0) and (evaluacionCO<=5)			La evolución de un curso optativo no debe ser menor que cero ni mayor que cinco		

Tabla 25 Validación de la tabla reportem.

Tabla	Reportem				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idRepMedio	✓	PK			
fechaReporte	✓				
informacionReporte	✓				
numeroSerie		FK	Medio	Cascade	Cascade
Columnas Unitarias	idRepMedio				
	numeroSerie				
Chequeo			Descripción		
fechaReporte <= CURRENT_DATE			La fecha del reporte del medio tiene que ser menor o igual que la fecha actual.		

Tabla 26 Validación de la tabla hpersona.

Tabla	hpersona				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idPersona	✓	PK			
hfechaIngreso	✓				
fechaOperacion	✓				
tipoOperacion	✓				
hcausa	✓				
hintegracionRev	✓				
hnombre	✓				
hdireccionP	✓				
hTipoPersona	✓				
Columnas Unitarias	idPersona				

Tabla 27 Validación de la tabla hmedio.

Tabla	Hmedio				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
hidMedio	✓	PK			
hfechaEntrada	✓				
fechaOperacionMedio	✓				
tipoOperacionMedio	✓				
htipoMedio	✓				
hnombreMedio	✓				
Columnas Unitarias	hidMedio				

Tabla 28 Validación de la tabla cursoOptativo.

Tabla	cursoOptativo				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idCursoOptativo	✓	PK			
nombreCurso	✓				
nombrePerfil	✓	FK	perfil	Cascade	Restrict
Columnas Unitarias	idCursoOptativo				
	nombrePerfil				

Tabla 29 Validación de la tabla perfil.

Tabla	Perfil				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idperfil	✓	PK			
nombrePerfil	✓				
cantMinima	✓				
Columnas Unitarias	nombrePerfil, idperfil				
Chequeo		Descripción			
(cantminima >= 0)		El número de cantidad mínima es un valor entero y positivo.			

Tabla 30 Validación de la tabla evaluacionpersona.

Tabla	Evalucionpersona				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idEvalucionPersona	✓	PK			
evaluacionPersona	✓				
fechaEvaluacionP	✓				
descripcionEval					
Columnas Unitarias	idEvalucionPersona				
Chequeo		Descripción			
fechaEvaluacionP <= CURRENT_DATE		La fecha evaluación de persona tiene que ser menor o igual que la fecha actual.			

Tabla 31 Validación de la tabla disiplina.

Tabla	Disciplina				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
iddiscip	✓	PK			
nombreDisip	✓				
nombrelocal	✓	FK	departamento	Cascade	Restrict
Columnas Unitarias	nombreDisip, iddiscip				
	Nombrelocal				

Tabla 33 Validación de la tabla local.

Tabla	Local				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idlocal	✓	PK			
nombrelocal	✓				
Columnas Unitarias	Nombrelocal, idlocal				

Tabla 34 Validación de la tabla evaluacionTesis.

Tabla	evaluacionTesis				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idEvaluacionT	✓	PK			
evaluacionTesi	✓				
descriccionevaltesis					
fechaevaluacionT	✓				
idTesis		FK	tesis	Cascade	Restrict
Columnas Unitarias	idEvaluacionT				
	idTesis				

Chequeo	Descripción
(evaluacionTesi >=0) and (evaluacionTesi <=5)	La evolución de un curso optativo no debe ser menor que cero ni mayor que cinco
fechaEvaluacionT <= CURRENT_DATE	La fecha de la evaluación de la tesis tiene que ser menor o igual que la fecha actual.

Tabla 35 Validación de la tabla medio.

Tabla	Medio				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
numeroRerie	✓	PK			
fechaEntrada	✓				
nombreMedio	✓				
estadoMedio	✓				
tipoMedio	✓				
nombreLocal	✓	FK	local	Cascade	Restrict
Columnas Unitarias	numeroSerie				
	nombreLocal				
Chequeo			Descripción		
fechaEntrada <= CURRENT_DATE			La fecha de entrada del medio tiene que ser menor o igual que la fecha actual.		

Tabla 36 Validación de la tabla estudiante.

Tabla	estudiante				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
solapin	✓	PK			
grupo	✓				
idTesis	✓	FK	tesis	Cascade	Restrict
Columnas Unitarias	solapin				
	idTesis				

Tabla 37 Validación de la tabla alumnoayudante.

Tabla	alumnoayudante				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
solapin	✓	PK			
asignaturaimparte	✓				
Columnas Unitarias	solapin				

Tabla 38 Validación de la tabla tesis.

Tabla	tesis				
Columna	No Nulo	Llave	Tabla Foránea	OnUpdate	OnDelete
idTesis	✓	PK			
fechaTesis	✓				
tituloTesis	✓				
Columnas Unitarias	idTesis				

Chequeo	Descripción
fechaTesis <= CURRENT_DATE	La fecha de tesis tiene que ser menor o igual que la fecha actual.

3.3 Normalización

Cuando se va a realizar el diseño de una base de datos relacional, se tiene en cuenta que las tablas que la componen cumplan con las reglas de normalización, que no es más que el proceso que permite eliminar la redundancia de los datos y evitar los problemas al insertar, eliminar y actualizar los datos, es decir, optimizar el trabajo con la información almacenada. En este proceso existen varios niveles de normalización, pero los tres primeros son los más usados. Las reglas de normalización se relacionan entre ellas de forma dependiente, lo que indica que para que una base de datos se encuentre en una determinada forma normal, primeramente tiene que cumplir con las reglas de los niveles inferiores de normalización. Esto quiere decir que una base de datos esta en 2da Forma Normal si previamente cumple con las reglas de normalización del 1er nivel. Cada regla que se cumple aumenta el grado de normalización del esquema de relación. Si una regla no se cumple, el esquema se debe descomponer en varios esquemas de relación que si la cumplan por separado. Se dice que una base de datos se encuentra en determinada forma normal si cumple con las restricciones correspondientes a dicha forma normal.

Se puede afirmar que el diseño del módulo de la base de datos propuesto se encuentra normalizado hasta Tercera Forma Normal. Se puede plantear que el esquema de relación del módulo está en 1ra Forma Normal puesto que podemos asegurar que cada tupla contiene exactamente un valor para cada atributo de las tablas de la base de datos, es decir, no existen campos multivaluados. También se cumple que el esquema de relación está en 2da Forma Normal, porque primeramente se encuentran en 1ra Forma Normal, y además todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria. Finalmente se puede plantear que los esquemas de relación se encuentran en 3ra Forma Normal, porque primeramente se encuentran en 2da Forma Normal, y además que no existen dependencias transitivas entre llaves candidatas y atributos no primos.

Conclusiones

En el desarrollo del presente trabajo de diploma se dio cumplimiento al objetivo general, diseñando la base de datos para la gestión de la información de los recursos materiales y humanos de la facultad, para una mayor organización de estos recursos.

La Base de datos diseñada permite al sistema el control y la información de los recursos así como el almacenamiento y la búsqueda de otras informaciones.

Los privilegios que fueron asignados en la base de datos permiten el acceso a los mismos de una forma segura y diferenciada.

Recomendaciones

Luego de haber realizado el presente trabajo, se recomienda:

- Aplicar la Gestión de Información de los Recursos de la facultad a otras facultades de acurdo a sus necesidades.
- Proporcionar mayor funcionalidad al sistema atendiendo a las necesidades del cliente.

Bibliografía

Alburquerque Arias, Amado. 2007. *Sistema Integrado de Gestión Estadística (SIGE). Rol diseñador de la base de datos.* 2007.

Donatien Goliath Karenia, Rodriguez Martínez Yudermis. 2007. *Documentación imprescindible para los flujos de trabajo de diseño e implementación de software de gestión.* 2007.

Kuroki, Christian. 2005. *Migración a PostgreSQL desde otras bases de datos.* 2005.

Leonel de Jesús Castillo Santanal,Yaniel Alvarez Sánchez. 2007. *Base de Datos para la Residencia de la UCI.* 2007.

M. Piquer, José. 2002. dcc.uchile.c. *dcc.uchile.c.* [En línea] 2002. http://www.dcc.uchile.cl/~jpiquer/Extension/Informatica/uoct/uoct.html.

Maestros del Web. 2007. maestrosdelweb.com. *maestrosdelweb.com.* [En línea] 2007. http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

Molpeceres, Alberto. 2002. *Prosesos de desarrollo: RUP, XP, FDD.* 2002.

PosrgresSQLPE. 2007. postgresql.org.pe. *postgresql.org.pe.* [En línea] 2007. http://postgresql.org.pe.

Proyecto S.O.B.L. 2000. sobl.org. *sobl.org.* [En línea] 2000. http://sobl.org.

Tienda Linux. 2003. tiendalinux.com. *tiendalinux.com.* [En línea] 2003. http://soporte.tiendalinux.com/portal/Portfolio/postgresql_html.

Universidad de Magala. 2007. lcc.uma.es. *lcc.uma.es.* [En línea] 2007. http://www.lcc.uma.es/%7Egalvez/ftp/bdst/Tema2.pdf.

W. Hansen Gary, V. Hansen James. *Diseño y Administracion de Bases de Ddatos.*

wikipedia. wikipedia.org. *wikipedia.org.* [En línea] http://es.wikipedia.org/wiki/PostgreSQL.

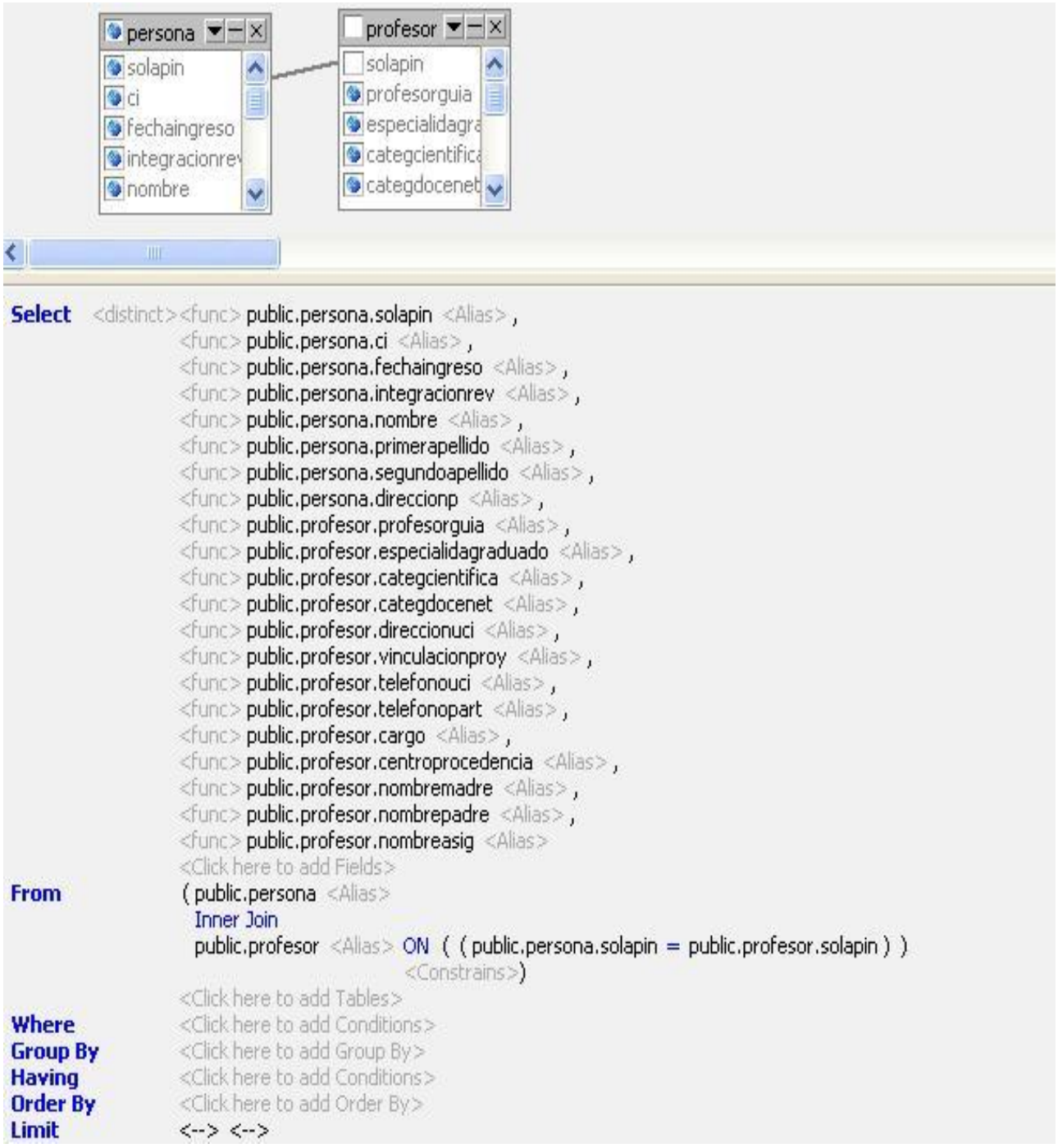
Terry, C. E., & Morales, M. M. (s.f.). Recuperado el 9 de noviembre de 2007, de Los Sistemas de Información de Gestión y la eficiencia empresarial: http://www.congreso-info.cu/UserFiles/File/Info/Info2006/Ponencias/104.pdf

Booch, Grady, Jacobson, Ivan y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación. S.A., 2000.

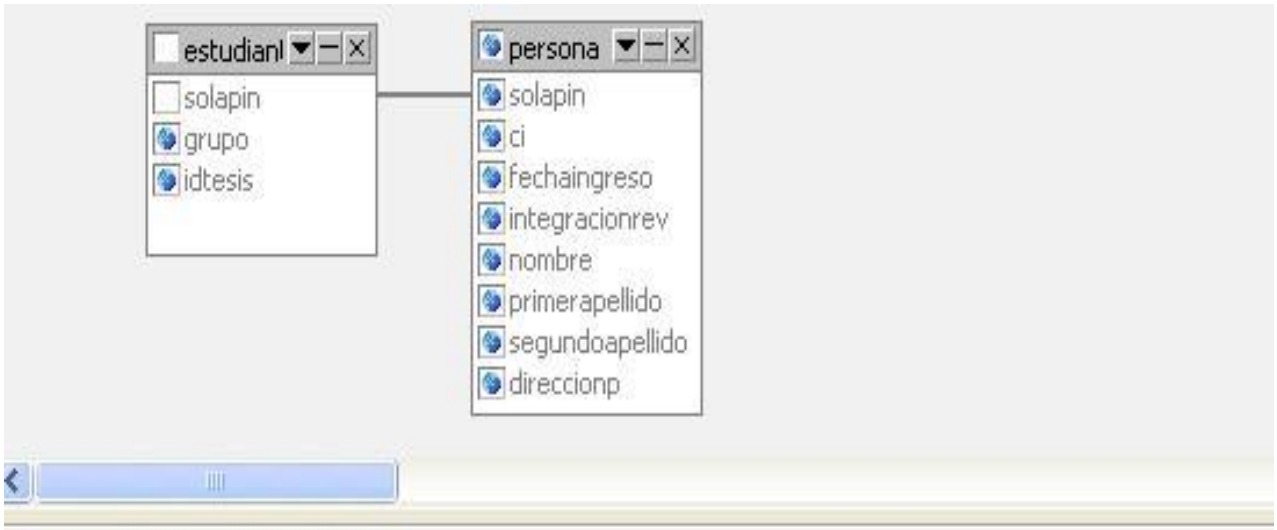
hospedajeydominios.com. (2005). *hospedajeydominios.com*. Obtenido de hospedajeydominios.com: http://www.hospedajeydominios.com/mambo/documentacion-manual_php-pagina-security_database.html

Anexos

Anexo I Vista del caso de uso obtener profesores.

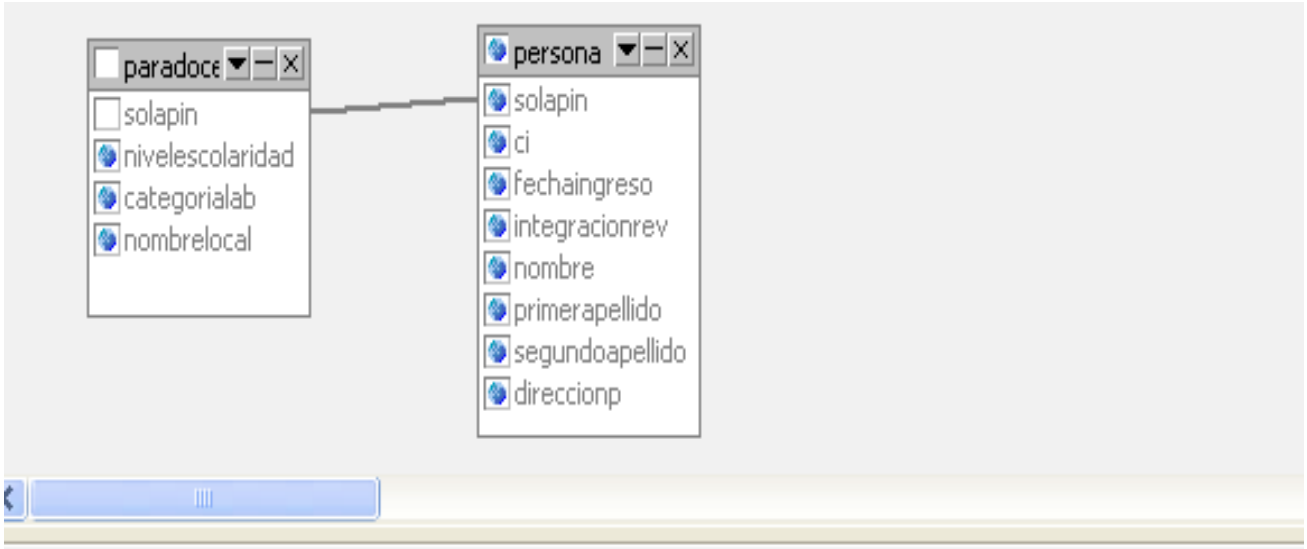


Anexo II Vista del caso de uso obtener estudiante.



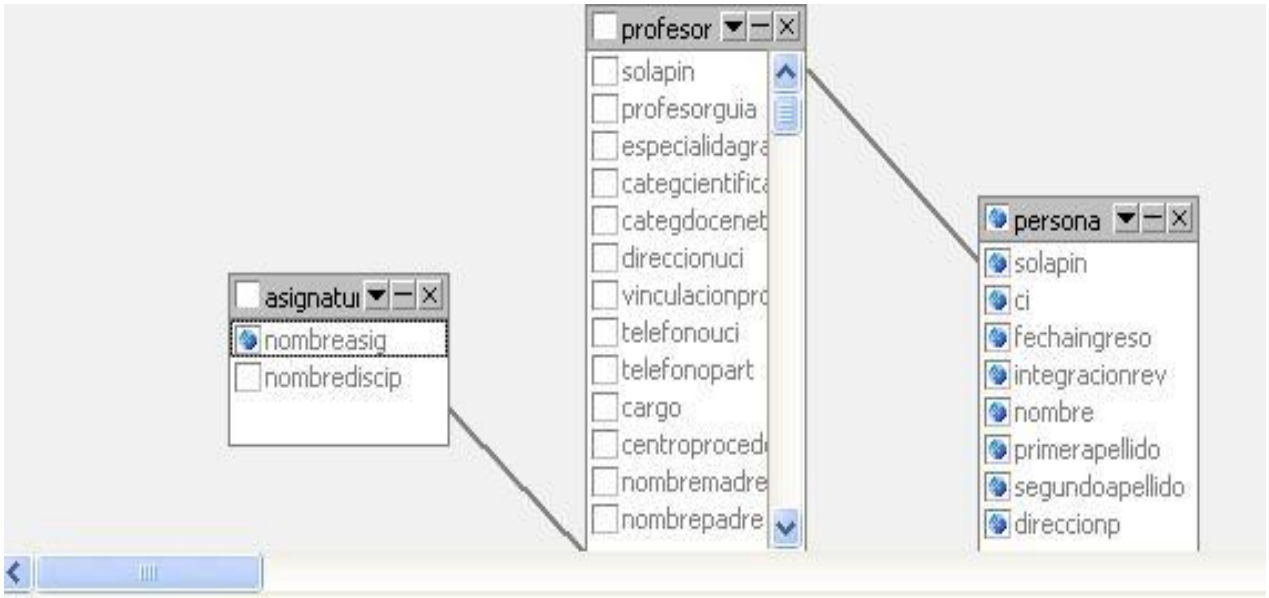

```
Select   <distinct><func> public.estudiante.grupo <Alias> ,
                  <func> public.estudiante.idtesis <Alias> ,
                  <func> public.persona.solapin <Alias> ,
                  <func> public.persona.ci <Alias> ,
                  <func> public.persona.fechaingreso <Alias> ,
                  <func> public.persona.integracionrev <Alias> ,
                  <func> public.persona.nombre <Alias> ,
                  <func> public.persona.primerapellido <Alias> ,
                  <func> public.persona.segundoapellido <Alias> ,
                  <func> public.persona.direccionp <Alias>
                  <Click here to add Fields>
From              ( public.estudiante <Alias>
                    Inner Join
                    public.persona <Alias> ON ( ( public.persona.solapin = public.estudiante.solapin ) )
                                              <Constrains>)
                  <Click here to add Tables>
Where             <Click here to add Conditions>
Group By          <Click here to add Group By>
Having            <Click here to add Conditions>
Order By          <Click here to add Order By>
Limit             <--> <-->
```

Anexo III Vista del caso de uso obtener paradocentes.



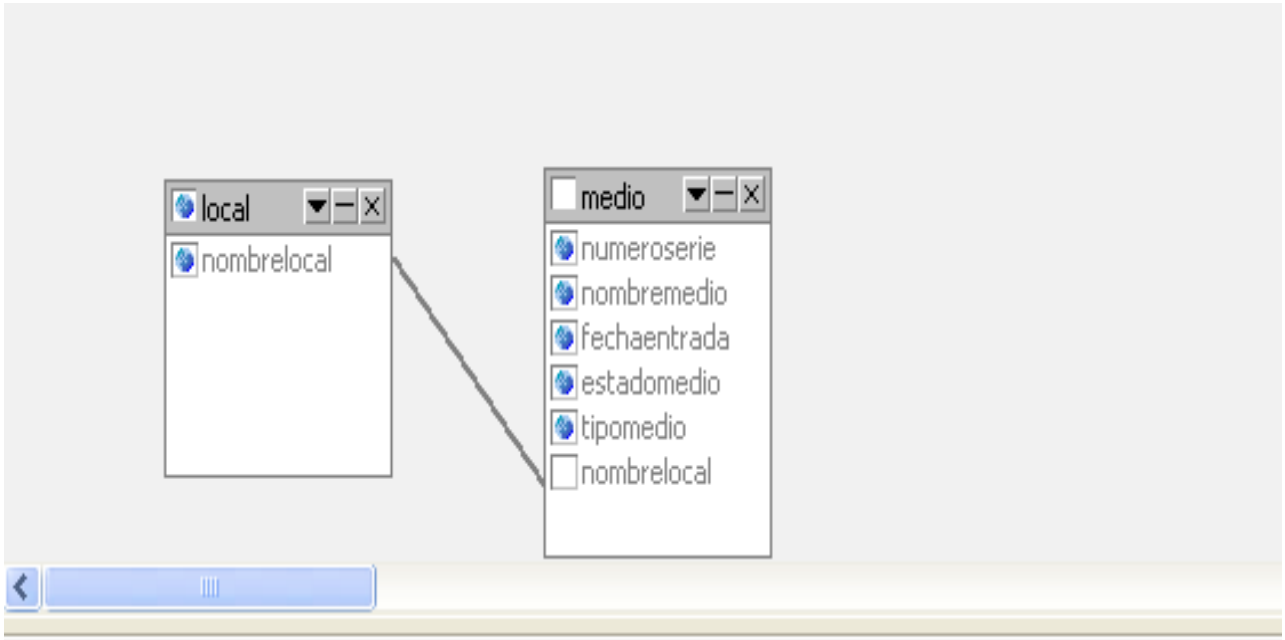

```
Select   <distinct><func> public.paradocente.nivelescolaridad <Alias> ,
                  <func> public.paradocente.categorialab <Alias> ,
                  <func> public.paradocente.nombrelocal <Alias> ,
                  <func> public.persona.solapin <Alias> ,
                  <func> public.persona.ci <Alias> ,
                  <func> public.persona.fechaingreso <Alias> ,
                  <func> public.persona.integracionrev <Alias> ,
                  <func> public.persona.nombre <Alias> ,
                  <func> public.persona.primerapellido <Alias> ,
                  <func> public.persona.segundoapellido <Alias> ,
                  <func> public.persona.direccionp <Alias>
                  <Click here to add Fields>
From              ( public.paradocente <Alias>
                    Inner Join
                    public.persona <Alias> ON ( ( public.persona.solapin = public.paradocente.solapin ) )
                                              <Constrains>)
                  <Click here to add Tables>
Where             <Click here to add Conditions>
Group By          <Click here to add Group By>
Having            <Click here to add Conditions>
Order By          <Click here to add Order By>
Limit             <--> <-->
```

Anexo IV Vista del caso de uso obtener profesores de una asignatura.




```
Select   <distinct> <func> public.persona.solapin <Alias> ,
                    <func> public.persona.ci <Alias> ,
                    <func> public.persona.fechaingreso <Alias> ,
                    <func> public.persona.nombre <Alias> ,
                    <func> public.persona.integracionrev <Alias> ,
                    <func> public.persona.primerapellido <Alias> ,
                    <func> public.persona.segundoapellido <Alias> ,
                    <func> public.persona.direccionp <Alias> ,
                    <func> public.asignatura.nombreasig <Alias>
                    <Click here to add Fields>
From                public.profesor <Alias>
                    Inner Join
                    public.persona <Alias> ON public.profesor.solapin = public.persona.solapin
                                              <Constrains>
                    Inner Join
                    public.asignatura <Alias> ON public.asignatura.nombreasig = public.profesor.nombreasig
                                                 <Constrains>
                    <Click here to add Tables>
Where               <Click here to add Conditions>
Group By            <Click here to add Group By>
Having              <Click here to add Conditions>
Order By            <Click here to add Order By>
```

Anexo V Vista del caso de uso obtener medios de un local.




```
Select   <distinct><func> public.medio.numeroserie <Alias>,
                  <func> public.medio.nombremedio <Alias>,
                  <func> public.medio.fechaentrada <Alias>,
                  <func> public.medio.estadomedio <Alias>,
                  <func> public.medio.tipomedio <Alias>,
                  <func> public.local.nombrelocal <Alias>
                  <Click here to add Fields>
From              ( public.local <Alias>
                    Inner Join
                    public.medio <Alias> ON ( ( ( public.local.nombrelocal = public.medio.nombrelocal ) ) )
                                            <Constrains>)
                  <Click here to add Tables>
Where             <Click here to add Conditions>
Group By          <Click here to add Group By>
Having            <Click here to add Conditions>
Order By          <Click here to add Order By>
Limit             <--> <-->
```