UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad Regional Mártires de Artemisa

TÍTULO: *Base de datos para las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa.*

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

AUTOR: *Amed Izaguirre Fernández.*

TUTORA: *Ing. Anaibis Álvarez Morales.*

CO-TUTOR: *Lic. José Ángel Dieppa.*

Artemisa, Junio 2012.

"Año 54 de la Revolución"

Declaración de Autoría

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con el mismo.

Para que así conste firmo la presente a los ____ días del mes de _____ del 2012.

Amed Izaguirre Fernández.

_____

Firma del Autor.

Ing. Anaibis Álvarez Morales.

_____

Firma de la Tutora.

Lic. José Ángel Dieppa.

_____

Firma del Co-Tutor.

Agradecimientos

A mis amigos en especial a José Ramón (Monti), Ernesto, Leonel y Josué por recibir de ellos el apoyo en el momento y forma requeridos.

A mi familia por sobrevalorarme y confiar, sobre todo en momentos en que ya no confiaba en mi mismo.

A mi tutora y co-tutor por tener la paciencia para guiar, moldear mi rebeldía y sacar de mi lo que valgo verdaderamente.

Al tribunal por sus señalamientos precisos y oportunos, permitiéndome aprender aún en el último instante.

Dedicatoria

A mi familia y a todos mis amigos, a los que he conocido durante toda mi vida y a los que conocí en esta universidad, sobre todo a aquellos que por una razón u otra no se encuentran en este centro universitario. Para todas estas personas es este trabajo, su ofrenda.

Resumen

La provincia de Artemisa cuenta con un Órgano de Administración Provincial que esta conformado por 32 direcciones provinciales, dentro de las cuales se encuentran las Direcciones de Cultura y Deporte.

El presente trabajo expone la propuesta de diseño de la base de datos y el diseño e implementación de la capa de acceso a datos o capa de persistencia para las Direcciones de Cultura y Deporte del Órgano del Gobierno Provincial de Artemisa, teniendo en cuenta los problemas existentes en estas direcciones con la disponibilidad, confiabilidad e integridad de la información y las consecuencias que esto acarrea.

Se utilizaron las capacidades de los *frameworks* Spring e Hibernate y el Entorno de Desarrollo Integrado Netbeans para la implementación de la capa de acceso a datos. Para la realización de la propuesta de diseño de la base de datos se utilizaron las herramientas Visual Paradigm y Power Architect. Una vez confeccionada la propuesta, se sometió a evaluación mediante la realización de pruebas de unidad permitiendo la validación o evaluación de los componentes desarrollados para la persistencia de objetos en la aplicación.

La propuesta contribuye al correcto almacenamiento y recuperación de la información garantizando su integridad, confiabilidad y disponibilidad. Los componentes implementados en la capa de persistencia constituyen la base para la implementación de los componentes de las capas superiores o específicamente de la capa de negocio, posibilitándole a los desarrolladores de la misma, centrarse en la implementación de las reglas de negocio o requerimientos del sistema.

Palabras claves: Bases de datos y capa de acceso a datos.

Índice

Índice de Tablas

Introducción

La humanidad desde sus inicios ha necesitado organizar y analizar los conocimientos teniendo en cuenta la experiencia adquirida con el transcurso del tiempo. A medida que el nivel cognoscitivo y la experiencia aumentaban surgió la necesidad de administrar la información con la que se contaba, lo que constituyó una reestructuración de hábitos de vida y mecanismos diversos en la búsqueda del bienestar. Era necesario que la información construida mediante la experiencia pudiera llegar a las futuras generaciones.

El ser humano, poco a poco, fue descubriendo que la experiencia y el análisis de la información de los fenómenos podrían serle de utilidad. Administrar la información siempre le preocupó al hombre, desde el momento mismo en que se dio cuenta de que tenía que aprovechar los conocimientos de experiencias colectivas y más aún, tener estrategias de búsqueda para poder hacerse del conocimiento. Actualmente la sociedad se proyecta con disímiles características económicas y un desarrollo vertiginoso de las tecnologías de la información y las comunicaciones.

El uso cotidiano de las computadoras, los dispositivos de almacenamiento óptico, la aparición de Internet y la proliferación de las Intranets han constituido un basamento para el desarrollo de las ciencias, los negocios y de la sociedad en general. Esto ha impulsado considerablemente el hecho de que la sociedad reconozca cada día con mayor fuerza el valor de la información como recurso y la necesidad de medios adecuados para su almacenamiento y recuperación.

La administración de la información es realizada mediante el uso de diversos medios dentro de los cuales la utilización de las bases de datos constituye el más idóneo para mantener accesible y actualizada la información, pues contar con información oportuna se traduce en productos y servicios de alta calidad, mayor competitividad y una adecuada toma de decisiones.

La información no se desgasta con el uso y permite su transmisión y duplicación casi instantánea y en este sentido las bases de datos adquieren su razón de ser

permitiendo la correcta gestión de los datos. Cuba en temas de administración de la información ha acumulado experiencia en la conformación de redes y bases de datos para sistemas especializados que pueden responder a las necesidades de información de todos los ciudadanos, aun cuando existen limitaciones tecnológicas propias de un país subdesarrollado y embargado económicamente.

El conocimiento como factor fundamental para la creación de riquezas, ha incidido para que el país haya desarrollado y puesto en práctica diferentes políticas y programas con lo referente a la información. Como parte de las acciones tomadas en respuesta a esta política y con el propósito de potenciar la cultura y desarrollo de las tecnologías de la informática y las comunicaciones, así como los sistemas de información, surge en el fragor de “La Batalla de Ideas” la Universidad de Ciencias Informáticas y posteriormente sus facultades regionales.

Entre las facultades regionales se encuentra la Facultad Regional Mártires de Artemisa, la cual se encuentra inmersa en el desarrollo de proyectos informáticos tanto de interés interno como para otras instituciones estatales. Uno de esos proyectos es el sistema para el Órgano del Gobierno Provincial de la provincia de Artemisa, surgida con la división política administrativa adoptada por el país en el año 2011.

La Administración Provincial cuenta con 32 direcciones entre las cuales se encuentran la Dirección de Cultura y la de Deporte, en las cuales se tramita toda la información relacionada con el seguimiento, control y fomento de las actividades culturales y deportivas respectivamente.

Actualmente existen en las Direcciones de Cultura y Deporte un conjunto de dificultades en cuanto al manejo de la información pues los datos se encuentran en fuentes no seguras: en formato duro o de manera digital en libros de cálculo de Microsoft Excel. Esta manera de proceder con los datos genera descontrol y desorganización; se compromete la integridad y confiabilidad de los datos, ocurren modificaciones no deseadas y se descentraliza la información.

Ejemplo de como repercute esta situación en la Dirección de Cultura es la dificultad

en la planificación de actividades previstas con la calidad requerida por medio del desempeño de los promotores socio-culturales. Las transferencias de datos por correo electrónico o dentro de la red local sin cifrar pueden ser vulneradas por agentes externos. Cualquier daño que pueda producirse tendría amplías repercusiones tratándose libros de cuentas, balances de pago, disponibilidad de libros y presupuestos para festivales, actos políticos y fiestas populares.

Con respecto a la Dirección Deporte el procedimiento inadecuado de administración de la información se evidencian claramente cuando se convoca a un evento deportivo y los metodólogos deportivos no posean la información centralizada y precisa por los departamentos que conforman la Dirección, lo cual afecta la participación de los atletas al no estar informados en el tiempo establecido sobre eventos tales como: Ceremonias Deportivas, Programas de Educación y Cultura Física, siendo todos ellos vitales para intensificar la práctica sistemática del deporte, provocando así que la provincia carezca de resultados meritorios en cuanto a las diferentes actividades deportivas.

De forma general en la dinámica de trabajo de las Direcciones de Cultura y Deporte se invierte gran cantidad de tiempo y fuerza de trabajo en la búsqueda y consulta de los datos con los cuales se trabaja. Todos los departamentos no tienen acceso a la misma información que se maneja simultáneamente dentro de las Direcciones de Cultura y Deporte y el resto de las direcciones que componen el Órgano del Gobierno Provincial.

Problema Científico:

¿Cómo contribuir a la integridad, confiabilidad, centralidad y disponibilidad de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa?

Objeto de Estudio:

Procesos de gestión de la información en los sistemas gubernamentales.

Campo de Acción:

Base de datos relativa a los procesos de gestión de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.

Objetivo General:

Desarrollar una base de datos que contribuya a la integridad, confiabilidad, centralidad y disponibilidad de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.

Idea a Defender:

Con el desarrollo de una base de datos se contribuirá a la integridad, confiabilidad, centralidad y disponibilidad de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.

Objetivos Específicos:

1. Fundamentar teóricamente los elementos relacionados con la base de datos en el proceso de almacenamiento y recuperación de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.
2. Diseñar la base de datos a partir de las necesidades del proceso de almacenamiento y recuperación de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.
3. Implementar la capa de acceso a datos para el proceso de almacenamiento y recuperación de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.
4. Validar la capa de acceso a datos para perfeccionar el proceso de almacenamiento y recuperación de la información en las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.

Con el propósito de resolver el problema, dar cumplimiento a los objetivos y argumentar la idea a defender se hace uso de diferentes métodos científicos:

Métodos Científicos:

Teóricos:

1. **Analítico – Sintético**

Este método se utilizó para realizar un análisis de las tendencias actuales en cuanto al diseño de modelos de datos, así como las herramientas utilizadas y buenas costumbres para la implementación de bases de datos, capas de acceso a datos u objetos de acceso a datos.

2. **Histórico-Lógico**

Este método se utilizó en el estudio y análisis de las principales herramientas utilizadas para el manejo de datos. Permitió analizar las características de diferentes herramientas existentes que son utilizadas para la creación de capas de acceso y realizar un estudio histórico de las mismas, lo que permitió valorar y determinar lo más factible a usar para la presente investigación.

3. **Inductivo-Deductivo**

A partir del planteamiento del problema a resolver y aplicando correctamente la lógica de deducción se obtienen nuevos conocimientos que son sometidos a verificación. Se dedujo que con la realización de la base de datos y capa de acceso a datos de las Direcciones Provinciales de Cultura y Deporte, se garantizará el acceso concurrente y adecuado a la totalidad de la información referente a estas direcciones.

4. **Modelación**

Este método se utilizó en el momento de diseñar la base de datos y la capa de acceso, pues se tuvo que definir un modelo de datos que cumpla con los requisitos necesarios.

Métodos empíricos:

1. **Análisis Documental**

El análisis documental fue aplicado durante la selección de los elementos de información relevantes de los documentos resultantes del levantamiento de

información.

Variables Independientes: Integridad, confiabilidad, centralidad y disponibilidad de los datos.

Variable Dependiente: Base de datos para las Direcciones de Cultura y Deporte de la Administración Provincial de Artemisa.

Aportes Prácticos

Se garantiza el correcto almacenamiento y recuperación de la información de las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa, el acceso a la información en todo momento y de forma concurrente y la seguridad física y de acceso a datos.

Justificación Científica

La mayoría de los sistemas informáticos modernos que trabajan con un Sistema Gestor de Bases de Datos, requieren una Capa de Acceso a Datos o Capa de Persistencia, la cual es un componente fundamental en los sistemas informáticos, principalmente los sistemas de tipo aplicación web. La capa de acceso a datos es una porción de código que se encarga de realizar el acceso a los datos. Su origen se encuentra vinculado con el patrón Modelo-Vista-Controlador. Actualmente este patrón es muy utilizado en las aplicaciones que se necesiten vincularse con bases de datos. (Ruiz, 2011)

Estructura del Documento

La estructura del documento consta de tres capítulos: Fundamentación teórica, Diseño y arquitectura de la capa de acceso a datos y Adquisición y validación de los resultados del sistema.

Capítulo 1: Fundamentación teórica.

En este capítulo se mencionan los diferentes conceptos relacionados con la gestión de información, bases de datos, las metodologías de desarrollo del *software*, el

lenguaje de modelado, las herramientas a utilizar y tendencias actuales de las capas de acceso en los sistemas de hoy en día.

Capítulo 2: Diseño y arquitectura de la base de datos.

En este capítulo se exponen los diferentes requisitos funcionales y no funcionales del sistema. Se realiza la implementación de los artefactos del diseño de la base de datos e integración con la aplicación en la capa de persistencia y se describen las características de la arquitectura y los patrones de diseño para la implementación de la capa de acceso a datos.

Capítulo 3: Adquisición y validación de los resultados del sistema.

En este capítulo se valida la capa de acceso a datos mediante pruebas funcionales al sistema de gestión de información y se describen los resultados obtenidos de dichas pruebas.

Capítulo 1: Fundamentación Teórica

Introducción del Capítulo

Considerando las características y requerimientos capturados en la fase correspondiente al levantamiento de información, se exponen los temas relacionados con el proceso de almacenamiento y recuperación de la información para su mejor compresión. También con el desarrollo de este capítulo, se presentan una serie de conceptos a los que se hará referencia en el resto del trabajo. Se abordarán diferentes puntos referentes a las bases de datos, sistemas gestores de bases de datos, capas de acceso a datos y herramientas seleccionadas para su desarrollo.

1.1 Conceptos asociados al proceso de almacenamiento y recuperación de la información

La persistencia de la información es la parte más importante en una aplicación de *software*. Teniendo en cuenta que el modelo de objetos difiere en muchos aspectos del modelo relacional y que la aplicación estará diseñada orientada a objetos, la persistencia puede lograrse mediante el almacenamiento en una base de datos. Las bases de datos más populares hoy en día son relacionales. Relacionados con el proceso de almacenamiento y recuperación de la información en este trabajo existen diversos conceptos expuestos a continuación como: gestión de la información, *framework*, ORM, capa de acceso, entre otros.

1.1.1 Gestión de la Información

La gestión o administración de la información ha estado presente desde los inicios de la humanidad hasta la actualidad y el hombre se ha valido de diferentes técnicas, métodos y procedimientos para llevarla a cabo a lo largo de toda la historia, desde las pinturas rupestres de los primitivos hasta los sistemas informáticos del hombre moderno.

La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización. (Marrero, 2003)

Se establece, entonces, como un recurso básico para cualquier organización. (Curto, 2006)

De forma general se entiende como gestión de la información al proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma.

Gestión de la información es un concepto compuesto que engloba dos términos: **gestión** e **información**.

Algunas concepciones de **información** son las siguientes:

La información es un fenómeno que aporta significado o sentido a las cosas, ya que mediante códigos y conjuntos de datos, forma los modelos de pensamiento humano. (Definicion.de, 2008)

La información es un conjunto de datos acerca de algún suceso, hecho, fenómeno o situación, que organizados en un contexto determinado tienen su significado, cuyo propósito puede ser el de reducir la incertidumbre o incrementar el conocimiento acerca de algo. (Thompson, 2008)

De forma general la información es un conjunto organizado de datos, que constituye un mensaje sobre cierto fenómeno que permite resolver problemas y tomar decisiones.

Por otra parte sobre el significado de **gestión** existen varios conceptos:

El concepto de gestión hace referencia a la acción y al efecto de gestionar o de

administrar. Gestionar es realizar diligencias conducentes al logro de un negocio o de un deseo cualquiera. Administrar, por otra parte, consiste en gobernar, dirigir, ordenar, disponer u organizar. La gestión es también la dirección o administración de una empresa o de un negocio. (Definicion.de, 2008)

De forma general el término gestión es el conjunto de trámites que se llevan a cabo para resolver un asunto o concretar un proyecto.

1.1.2 Framework y ORM

FrameWork es un concepto sumamente genérico, se refiere a "ambiente de trabajo, y ejecución", En general los *framework*s son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). (SOA-agenda, 2007)

Las ventajas principales de utilizar *frameworks* son las siguientes:

1. El programador no necesita plantearse una estructura global de la aplicación, sino que el *framework* se encarga de proporcionarle un esqueleto.

2. Facilita la colaboración, ayudando a los desarrolladores a entender el código fuente de otros. Permite todo lo que sea definir y estandarizar, lo cual contribuye al ahorro de tiempo y trabajo en desarrollos colaborativos.

3. Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al *framework* concreto para facilitar el desarrollo.

Sobre el concepto de **ORM** existen diferentes interpretaciones, como por ejemplo:

El ORM es un componente de *software* que permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos. Uno de los componentes ORM más utilizado es el **Hibernate**, surgido del ambiente Java y llevado al uso del *framework* .NET con la versión NHibernate. (Ercoli, 2007)

ORM es un método poderoso para el diseño y la consulta de los modelos de base de datos a nivel conceptual, donde se describe la aplicación en términos fácilmente comprensibles por los usuarios no técnicos. En la práctica, los datos ORM modelos suelen captar más las reglas de negocio y son más fáciles de validar y desarrollar modelos de datos en otros enfoques. (Halpin, 2011)

Object Relational Mapping, u ORM, O/RM y O/R *mapping*, es una técnica empleada en la programación, para convertir datos entre sistemas incompatibles, como lo son las bases de datos relacionales y los lenguajes de programación. Esta conversión de datos entre los sistemas crea un efecto de base de datos virtual de objetos, que puede ser usada en el programa. (Amaya, 2009)

Es importante destacar que para un mejor entendimiento sobre la investigación el último concepto está más completo que los anteriores que se exponen pues está más enfocado al tema del trabajo en específico.

Actualmente abundan *frameworks* para casi cualquier requerimiento que puedan tener los sistemas actuales. Muchos *frameworks* han sido implementados como herramientas ORM como por ejemplo: Doctrine e Hibernate.

1.1.3 Capa de Acceso a Datos

Esta capa resuelve el acceso a datos, abstrayendo a su capa superior la complejidad del acceso e interacción con los diferentes orígenes de datos. Esta capa se encarga de proveer una interfaz de programación de aplicaciones simple de usar, orientada al negocio, sin exponer complejidades propias de un repositorio de datos.

En esta capa se resuelven:

1. Cualquier acceso a la base de datos.
2. Cualquier acceso a *filesystem*.

3. Cualquier acceso a otros sistemas.

4. Cualquier acceso a un repositorio de datos en cualquier forma.

Los objetos se construyen a partir del patrón (DAO) Data Access Object (Cuartas, 2011)

Como tal, la capa de acceso a datos sirve como puente entre la capa lógica de negocio y el proveedor de datos. Tiene como objetivo encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa. Si se hace uso de alguna herramienta ORM para la implementación de la capa de acceso a datos se utilizan las librerías de dicho *framework* para que las mismas realicen el trabajo.

1.2 Bases de Datos

La base de datos constituye un elemento esencial para los sistemas de gestión de información. Una base de datos es un "almacén" que permite guardar grandes cantidades de información de forma organizada para luego poder encontrarla y utilizarla fácilmente.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una **base de datos** se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada. (basesdedatos.org, 2010)

1.2.1 Definición y Características de las Bases de Datos

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más **columnas** y **filas**. Las columnas guardan una

parte de la información sobre cada elemento que se desee guardar en la tabla, cada fila de la tabla conforma un registro. (basesdedatos.org, 2010)

Algunos de los criterios más precisos con respecto a las bases de datos son los siguientes:

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. (Valdés, 2007)

Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo. (Matos, 1999)

Entre las principales características de los sistemas de base de datos se puede mencionar: Independencia lógica y física de los datos, redundancia mínima, acceso concurrente por parte de múltiples usuarios, integridad de los datos, consultas complejas optimizadas, seguridad de acceso y auditoría, respaldo y recuperación y acceso a través de lenguajes de programación estándar. (Valdés, 2007)

1.2.2 Ventajas y Desventajas

Las bases de datos a pesar de varios inconvenientes presentan un conjunto de ventajas que le permiten ser uno de los medios más seguros y eficaces para la administración de información.

Ventajas:

1. Control sobre la redundancia de datos.
2. Consistencia de datos.
3. Compartición de datos.
4. Mantenimiento de estándares.

5. Mejora en la integridad de datos.
6. Mejora en la seguridad.
7. Mejora en la accesibilidad a los datos.
8. Mejora en la productividad.
9. Mejora en el mantenimiento.
10. Aumento de la concurrencia.
11. Mejora en los servicios de copias de seguridad. (Valdés, 2007)

Desventajas:

Algunas de las desventajas que pueden presentar las bases de datos son: la complejidad, coste del equipamiento adicional, vulnerabilidad a los fallos, entre otras.

1. Complejidad.
2. Coste del equipamiento adicional.
3. Vulnerable a los fallos. (Valdés, 2007)

1.2.3 Tipos de Campos y Cardinalidad de las Relaciones

Cada Sistema de Base de Datos posee tipos de campos que pueden ser similares o diferentes. Los más comunes son:

1. **Numérico:** entre los diferentes tipos de campos numéricos se encuentran enteros “sin decimales” y reales “decimales”.
2. **Booleanos:** poseen dos estados: Verdadero “Si” y Falso “No”.
3. **Memos:** son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados.

4. **Fechas:** almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.

5. **Alfanuméricos:** contienen cifras y letras. Presentan una longitud limitada (255 caracteres).

6. **Autoincrementables:** son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta: Servir de identificador ya que resultan exclusivos de un registro.

El diseño de relaciones o cardinalidad entre las tablas de una base de datos puede ser la siguiente:

1. **Relaciones de uno a uno:** una instancia de la entidad A se relaciona con una y solamente una de la entidad B.

2. **Relaciones de uno a muchos:** cada instancia de la entidad A se relaciona con varias instancias de la entidad B.

3. **Relaciones de muchos a muchos:** cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B. (Valdés, 2007)

1.3 Modelos de Bases de Datos

Una de las vías para la clasificación de las bases de datos, puede ser de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una “descripción” de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos. (basesdedatos.org, 2010)

Bases de datos jerárquicas

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. Son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos. (basesdedatos.org, 2010)

Bases de Datos de Red

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

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales. (basesdedatos.org, 2010)

Bases de Datos Relacional

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de “relaciones”. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados “tuplas”.

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 base de datos. 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 bases de datos relacionales es SQL, *Structured Query Language* o *Lenguaje Estructurado de Consultas*, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos. (basesdedatos.org, 2010)

Bases de Datos Orientadas a Objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

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

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. (basesdedatos.org, 2010)

Base de Datos Documental

Permiten la indexación a texto completo y en líneas generales realizar búsquedas

más potentes. (basesdedatos.org, 2010)

Bases de Datos deductivas

Un sistema de bases de datos deductivas, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas bases de datos lógicas, a raíz de que se basan en lógica matemática. (basesdedatos.org, 2010)

1.4 Sistemas Gestores de Bases de Datos

Un **Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System)** es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. (CAVSI.com, 2010)

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) 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. (R.Ernando, 2008)

1.4.1 Requisitos, Características y Ejemplos de Sistemas Gestores de Bases de Datos.

Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, entre otros. (CAVSI.com, 2010)

1. **MySQL**: es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable para grandes volúmenes de datos.

2. **PostgreSQL y Oracle**: Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos y suelen ser utilizadas en intranets y sistemas de gran calibre.

3. **Access**: Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa Access, el cual crea un archivo .mdb.

4. **Microsoft SQL Server**: es una base de datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones. (R.Ernando, 2008)

Requisitos y Características

Un Sistema Gestor de Base de Datos debe permitir: Definir una base de datos (especificar tipos, estructuras y restricciones de datos), construir la base de datos (guardar los datos en algún medio controlado por el mismo SGBD) y manipular la base de datos (realizar consultas, actualizarla, generar informes).

Las características de un Sistema Gestor de Base de Datos (SGBD) son:

1. Abstracción de la información.

2. Independencia.

3. Redundancia mínima

4. Consistencia.

5. Seguridad.

6. Integridad.

7. Respaldo y recuperación.
8. Control de la concurrencia. (CAVSI.com, 2010)

1.5 Metodologías de Desarrollo de *Software*

Una metodología de desarrollo de *software* es un conjunto de pasos y procedimientos que deben seguirse para desarrollar un *software*.

1.5.1 Características deseables de una metodología

Una metodología está compuesta por: Cómo dividir un proyecto en etapas, qué tareas se llevan a cabo en cada etapa, qué restricciones deben aplicarse, qué técnicas y herramientas se emplean, y cómo se controla y gestiona un proyecto.

Algunas de las características deseables de una metodología son:

1. Existencia de reglas predefinidas.
2. Cobertura total del ciclo de desarrollo.
3. Verificaciones intermedias.
4. Planificación y control.
5. Comunicación efectiva.
6. Utilización sobre un abanico amplio de proyectos.
7. Fácil formación.
8. Herramientas CASE.
9. Actividades que mejoren el proceso de desarrollo.
10. Soporte al mantenimiento.
11. Soporte de la reutilización de *software.*

1.5.2 Clasificación de las metodologías

Las metodologías se clasifican de la siguiente forma:

1. **Estructuradas** (Orientadas a procesos, orientadas a datos y mixtas).
2. **No estructuradas** (Orientadas a objetos y sistemas de tiempo real).

Metodologías estructuradas

Se basan en la forma **top-down**.

1) Metodologías orientadas a procesos

La ingeniería del *software* se basa en el modelo básico de entrada/proceso/salida de un sistema.

Está compuesta por:

- Diagrama de flujo de datos (DFD).
- Diccionario de datos.
- Especificaciones de proceso.

Ejemplos: metodologías de DeMarco, Gene y Sarson, Yourdon.

2) Metodologías orientadas a datos

Son metodologías basadas en la información. Primero se definen las estructuras de datos y a partir de éstos, se derivan los componentes procedimentales.

Ejemplos: metodologías de Jackson, Warnier, Warnier-Orr.

Metodologías no estructuradas

1) Metodologías orientadas a objeto

La orientación a objetos unifica procesos y datos encapsulándolos en el concepto de objetos.

Tiene dos enfoques distintos:

1. Revolucionario, puro u ortodoxo. Rompen con las metodologías tradicionales. Ejemplos: metodologías OOD de Booch, CRC/RDD de Wirfs-Brock.

2. Sintetista o evolutivo. Toman como base los sistemas estructurados y conforman elementos de uno y otro tipo. Ejemplos: metodología OMT de *Rumbourgh*.

2) Sistemas de tiempo real

Procesan información orientada al control más que a los datos. Se caracterizan por concurrencia, priorización de procesos, comunicación entre tareas y acceso simultáneo a datos comunes. (R.Ernando, 2008)

1.5.3 Metodologías Tradicionales y Metodologías Ágiles

Uno de los mecanismos utilizados para diferenciar entre una metodología y otra es según su clasificación en Tradicionales y Metodologías Ágiles.

Metodologías Tradicionales

Presentan las siguientes características:

1. Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.

2. Cierta resistencia a los cambios.

3. Impuestas externamente

4. Proceso mucho más controlado, con numerosas políticas/normas

5. Existe un contrato prefijado
6. El cliente interactúa con el equipo de desarrollo mediante reuniones
7. Grupos grandes y posiblemente distribuidos
8. Más artefactos
9. Más roles
10. La arquitectura del *software* es esencial y se expresa mediante modelos

Ejemplos de Metodologías Tradicionales: WARNIER y JACKSON, MYERS y YOURDON, GANE y SARSON y DEMARCO y WEINBERG.

Metodologías Ágiles

Presentan las siguientes características:

1. Basadas en heurísticas provenientes de prácticas de producción de código.
2. Especialmente preparadas para cambios durante el proyecto.
3. Impuestas internamente (por el equipo).
4. Proceso menos controlado, con pocos principios.
5. No existe contrato tradicional o al menos es bastante flexible.
6. El cliente es parte del equipo de desarrollo.
7. Grupos pequeños (menos de 10 integrantes) y trabajando en el mismo sitio.
8. Pocos artefactos.
9. Pocos roles.
10. Menos énfasis en la arquitectura del *software*.

Ejemplos de Metodologías Ágiles: Programación Extrema (Extreme Programming, XP), SCRUM, Crystal Methodologies y Dynamic Systems Development.

1.6 Situación Actual de las Bases de Datos

El término “base de datos” se utiliza para referirse a una gran masa de datos que se encuentran relacionados entre sí. Estos datos se encuentran divididos en varias categorías que son los registros, los ficheros y las bibliotecas. Una base de datos permite la realización de consultas, reportes y filtrado de información.

Entre las bases de datos más utilizadas y conocidas en el mundo están: Oracle, Informix, Sybase, Visual Dbase y Borland Paradox.

Una base de datos es más rápida que un archivo de tarjetas, más práctica que un archivador y tiene mayor capacidad que una agenda, permitiendo reorganizar la información tan a menudo como sea necesario, adaptar los datos para nuevos usos, localizar información al instante, actualizar información y modificar o corregir datos.

El diseño de la base de datos es probablemente la clave del éxito o fracaso de la instrumentación de una aplicación, debido a que representa los cimientos de un sistema. Un incorrecto diseño puede inducir a situaciones de duplicación de los datos y falta de organización y coherencia de las salidas del sistema.

Aunque existen varios modelos de bases de datos (jerárquicos, redes, orientados a objetos), el más aceptado y popular es sin duda el relacional (Ullman 1982) debido a su potente basamento lógico-teórico soportado en el álgebra relacional. En este modelo los objetos se representan a través de tablas cuyas columnas (denominadas campos) son las propiedades o atributos. Las filas o tuplas contienen toda la información relevante a un objeto. (PgAdminIII, 2008)

En Cuba al igual que en el mundo entero, existen y se desarrollan múltiples sistemas vinculados a bases de datos, como por ejemplo:

1. Base de Datos del Sistema Bancario Cubano.

2. Base de Datos de la Biblioteca Nacional José Martí que contiene el catálogo general con toda la documentación registrada en la Biblioteca Nacional a partir del año 1998.

3. Base de Datos del Catálogo de Publicaciones Seriadas Cubanas entre los siglos dieciocho y veintiuno, donde se almacena la prensa patrimonial cubana correspondiente a los siglos dieciocho y veintiuno, publicada en Cuba y por cubanos radicados en el extranjero. Incluye periódicos y revistas de los siglos XX y XXI que se encuentran en la Biblioteca Nacional de Cuba José Martí. Esta base de datos forma parte del proyecto para la preservación de la memoria histórica: Catálogo Colectivo de Publicaciones Seriadas Cubanas.

4. Base de Datos Catálogo Colectivo del patrimonio cartográfico de Iberoamérica del Siglo XVI al XIX. Proyecto internacional que da a conocer los fondos cartográficos patrimoniales de los siglos XVI al XIX de todas las bibliotecas de Iberoamérica. Su existencia permitirá el acceso a gran cantidad de materiales cartográficos que se atesoran en las mismas. Los países participantes en este proyecto son: Argentina, Brasil, Costa Rica, Portugal, Puerto Rico y Cuba.

Ejemplos de Bases de Datos en el mundo:

1. YouTube: Con 100 millones de videos observados por día y 65.000 videos agregados cada día y el 60% de todos los videos mirados en línea.

2. Google: Con 91 millones de búsquedas por día. Es responsable del 50% de todas las búsquedas en Internet.

3. *World Data Center for Climate*: Con 220 Terabyte de datos de la Web.

Cada una de los sistemas modernos que dependan de información precisa y

actualizada o se encarguen de administrar información, necesariamente utilizan una base de datos y la capa de acceso es la responsable del vínculo operacional entre el usuario y el sistema y la correspondiente base de datos.

En la actualidad, la tendencia más aceptada para el desarrollo de aplicaciones es la aplicación de patrones de diseño de arquitectura que dividen la responsabilidad en distintas capas que interactúan unas con otras a través de sus interfaces. Se trata de los sistemas denominados multicapa o n-capas. Aplicados a los proyectos web, el modelo más básico es el de aplicaciones 3 capas, donde la capa de acceso a datos es la encargada de persistir las entidades que se manejan en negocio, el acceso a los datos almacenados y la actualización.

1.7 Metodología y Herramientas utilizadas en la creación del proyecto

Existen diferentes metodologías para el desarrollo de *software* y diversas herramientas que pueden ser empleadas para el correcto y eficiente manejo de la información.

Metodología Seleccionada

La metodología seleccionada para el desarrollo del proyecto es SXP. La misma está compuesta por las mejores prácticas de las metodologías SCRUM y XP y las mejores prácticas de Calisoft. Ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo. Esta metodología a diferencia de otras como las tradicionales, permite realizar cambios constantemente a medida que cambien los requisitos definidos del proyecto.

Metodologías que la conforman:

1. SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos.

2. XP más bien es una metodología encaminada para el desarrollo de *software*; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.

SCRUM (Metodología de gestión del trabajo) en conjunto con XP (Metodología para el proceso de desarrollo de *software*) dotan a SXP de las siguientes ventajas:

1. El líder de proyecto puede llevar un mejor control de las tareas y la planificación de las mismas.

2. Involucra a los miembros del equipo de desarrollo en las decisiones sobre el proyecto y sus vías de desarrollo.

3. El trabajo es ágil y mantiene al cliente dentro del equipo de desarrollo.

4. Integración continúa del sistema.

5. Reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación.

6. Todos los integrantes del equipo de trabajo conocen algo sobre todas las partes y muy bien aquellas en las que trabajan.

7. Todo el código se escribe en parejas.

8. Se realiza el trabajo de 1 persona en casi la mitad del tiempo y mejor.

Herramientas para el desarrollo de Bases de Datos y Capas de Acceso a Datos:

1-) pgAdmin3 versión 1.14.0

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma *wxWidgets*, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma a diferencia de la herramienta PostgreSQL Manager que aún siendo mucho mejor que PgAdmin III, se ejecuta solamente en versiones de Windows . (PgAdminIII, 2008)

2-) PostgreSQL versión 9.1

PostgreSQL es un Sistema de Gestión de Bases de Datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. PostgreSQL en comparación con otro gestor ampliamente utilizado como es el caso de MySQL, permite el manejo de gigantescas bases de datos donde un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (Martinez, 2010)

3-) Architect versión 0.9.13

Aplicación desarrollada en Java y distribuida bajo licencia GPLv3. Es multiplataforma, ya que está disponible para Linux, Windows y Mac OS X. Pero no es precisamente esta característica la que la hace destacable, es sobre todo que está diseñada para poder trabajar con diferentes Bases de datos: Oracle 8i, Oracle 9i, Oracle 10g, PostgreSQL, MySQL, SQL Server y DB2.

Power Architect a diferencia de la herramienta DIA por ejemplo, es algo más que una simple aplicación para generar diagramas E-R, pudiendo destacar las posibilidades de: generar los scripts de creación de tablas y procedimientos para

los modelos de datos, recuperar dichos modelos de una base de datos existente, o dicho de otra forma, Ingeniería Inversa (Reverse Engineering), analizar estructuras de datos entre diversos modelos de datos para ver las diferencias y similitudes y creación de perfiles. Con lo cual, se habla de una herramienta perfecta para DBAs (Administradores de Base de Datos), analistas y diseñadores. (Solano, 2009)

4-) Visual Paradigm versión 6.4

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (visual-paradigm.com, 2007)

5-) Netbeans versión 7.0.1

El IDE NetBeans es un entorno de desarrollo integrado (IDE) modular y basado en estándares, escrito con el lenguaje de programación Java. El proyecto de NetBeans consta de un IDE de código abierto y gran variedad de funciones escrito con el lenguaje de programación Java y una plataforma para aplicaciones de cliente enriquecidas que se puede utilizar como marco genérico para crear cualquier tipo de aplicación. (netbeans.org, 2010)

6-) Hibernate versión 3.0.2

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java aunque se encuentra disponible para .Net con la versión Nhibernate. Es un *framework* que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones. Hibernate es *software* libre, distribuido bajo los términos de la licencia GNU LGPL. Hibernate está diseñado para ser flexible, tiene la funcionalidad de

crear la base de datos a partir de la información disponible y ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language). (Albiol, 2003)

7-) Spring versión 3.0.2

Spring es un *framework* de código abierto para el desarrollo de aplicaciones para la plataforma Java. Cuenta con un conjunto de librerías de las que se puede escoger aquellas que faciliten el desarrollo de acuerdo con los requerimientos del sistema a implementar. Entre sus posibilidades más potentes está su contenedor de Inversión de Control, también llamado Inyección de Dependencias, que brinda la posibilidad de retirar el *framework* sin prácticamente cambiar líneas de código. Spring tiene integrado el framework Hibernate para las operaciones con la base de datos, reduciendo aún más el código a implementar a diferencia de la utilización Hibernate solamente. (Breidenbach, 2005)

Conclusiones del Capítulo

En este capítulo se expusieron conceptos y características referentes a las bases de datos, sistemas gestores de bases de datos y capas de acceso. Se detallaron las características principales y la arquitectura de las herramientas seleccionadas. Todas estas concepciones permitieron modelar el marco teórico y el modelo conceptual sobre el cual se fundamentó la investigación.

Capítulo 2: Diseño y arquitectura de la base de datos

Introducción del Capitulo

En el presente capítulo se describen los procesos principales que tiene lugar para el diseño de la base de datos y la implementación de la capa de acceso a datos de las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa, entre los que se encuentran: la selección de los requisitos, partiendo del modelo lógico y físico de la base de datos correctamente normalizada, describiendo las clases y tablas principales obtenidas en ambos modelos y finalmente el procedimiento empleado para el diseño e implementación de la capa de acceso a datos.

2.1 Requisitos Funcionales y no Funcionales del Sistema

2.1.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, más específicamente son acciones que el sistema debe ser capaz de realizar.

Requisitos Funcionales de la Dirección Cultura

RF1	Insertar información sobre balance económico de actividades culturales.
RF2	Modificar información sobre balance económico de actividades culturales.
RF3	Eliminar información sobre balance económico de actividades culturales.
RF4	Buscar información sobre balance económico de actividades culturales.

RF5	Insertar información sobre los indicadores de actividad bibliotecaria.
RF6	Modificar información sobre los indicadores de actividad bibliotecaria.
RF7	Eliminar información sobre los indicadores de actividad bibliotecaria.
RF8	Buscar información sobre los indicadores de actividad bibliotecaria.
RF9	Insertar información sobre actividades culturales municipales.
RF10	Modificar información sobre actividades culturales municipales.
RF11	Eliminar información sobre actividades culturales municipales.
RF12	Buscar información sobre actividades culturales municipales.
RF13	Insertar información sobre unidades artísticas categorizadas y talleres por municipios.
RF14	Modificar información sobre unidades artísticas categorizadas y talleres por municipios.
RF15	Eliminar información sobre unidades artísticas categorizadas y talleres por municipios.
RF16	Buscar información sobre unidades artísticas categorizadas y talleres por municipios.
RF17	Insertar información sobre equipamiento y estado constructivo de casas de cultura a nivel municipal.
RF18	Modificar información sobre equipamiento y estado constructivo de

	casas de cultura a nivel municipal.
RF19	Eliminar información sobre equipamiento y estado constructivo de casas de cultura a nivel municipal.
RF20	Buscar información sobre equipamiento y estado constructivo de casas de cultura a nivel municipal.
RF21	Insertar información sobre la superación profesional de los trabajadores.
RF22	Modificar información sobre la superación profesional de los trabajadores.
RF23	Eliminar información sobre la superación profesional de los trabajadores.
RF24	Buscar información sobre la superación profesional de los trabajadores.
RF25	Insertar información sobre trabajadores contratados.
RF26	Modificar información sobre trabajadores contratados.
RF27	Eliminar información sobre trabajadores contratados.
RF28	Buscar información sobre trabajadores contratados.
RF29	Insertar información sobre OACES y CAP.
RF30	Modificar información sobre OACES y CAP.
RF31	Eliminar información sobre OACES y CAP.

RF32	Buscar información sobre OACES y CAP.
RF33	Insertar información sobre títulos falsos.
RF34	Modificar información sobre títulos falsos.
RF35	Eliminar información sobre títulos falsos.
RF36	Buscar información sobre títulos falsos.

Requisitos Funcionales de la Dirección Deporte

RF1	Insertar información sobre la matrícula deportiva.
RF2	Modificar información sobre la matrícula deportiva.
RF3	Eliminar información sobre la matrícula deportiva.
RF4	Buscar información sobre la matrícula deportiva.
RF5	Modificar reporte de información sobre la matrícula deportiva.
RF6	Insertar información sobre la fuerza técnica.
RF7	Modificar información sobre la fuerza técnica.
RF8	Eliminar información sobre la fuerza técnica.
RF9	Buscar información sobre la fuerza técnica.
RF10	Modificar reporte de información sobre la fuerza técnica.
RF11	Insertar información sobre los eventos.

RF12	Modificar información sobre los eventos.
RF13	Eliminar información sobre los eventos.
RF14	Buscar información sobre los eventos.
RF15	Modificar reporte de información sobre los eventos.
RF16	Insertar información sobre los actos masivos.
RF17	Modificar información sobre los actos masivos.
RF18	Eliminar información sobre los actos masivos.
RF19	Buscar información sobre los actos masivos.
RF20	Modificar reporte de información sobre los actos masivos.
RF21	Insertar información sobre la población total del plan turquino.
RF22	Modificar información sobre la población total del plan turquino.
RF23	Eliminar información sobre la población total del plan turquino.
RF24	Buscar información sobre la población total del plan turquino.
RF25	Modificar reporte de información sobre la población total del plan turquino.
RF26	Insertar información sobre los colaboradores.
RF27	Modificar información sobre los colaboradores.
RF28	Eliminar información sobre los colaboradores.

RF29	Buscar información sobre los colaboradores.
RF30	Modificar reporte de información sobre los colaboradores.
RF31	Insertar información sobre las ceremonias deportivas.
RF32	Modificar información sobre las ceremonias deportivas.
RF33	Eliminar información sobre las ceremonias deportivas.
RF34	Buscar información sobre las ceremonias deportivas.
RF35	Modificar reporte de información sobre las ceremonias deportivas.
RF36	Insertar información sobre los festivales gimnásticos.
RF37	Modificar información sobre los festivales gimnásticos.
RF38	Eliminar información sobre los festivales gimnásticos.
RF39	Buscar información sobre los festivales gimnásticos.
RF40	Modificar reporte de información sobre los festivales gimnásticos.
RF41	Insertar información sobre los practicantes sistemáticos.
RF42	Modificar información sobre los practicantes sistemáticos.
RF43	Eliminar información sobre los practicantes sistemáticos.
RF44	Buscar información sobre los practicantes sistemáticos.
RF45	Modificar reporte de información sobre los practicantes sistemáticos.

RF46	Insertar información sobre los consejos voluntarios deportivos.
RF47	Modificar información sobre los consejos voluntarios deportivos.
RF48	Eliminar información sobre los consejos voluntarios deportivos.
RF49	Buscar información sobre los consejos voluntarios deportivos.
RF50	Modificar reporte de información sobre los consejos voluntarios deportivos.
RF51	Insertar información sobre los activistas.
RF52	Modificar información sobre los activistas.
RF53	Eliminar información sobre los activistas.
RF54	Buscar información sobre los activistas.
RF55	Modificar reporte de información sobre los activistas.
RF56	Insertar información sobre las instalaciones deportivas.
RF57	Modificar información sobre las instalaciones deportivas.
RF58	Eliminar información sobre las instalaciones deportivas.
RF59	Buscar información sobre las instalaciones deportivas.
RF60	Modificar reporte de información sobre las instalaciones deportivas.
RF61	Insertar información de las reuniones provinciales.

RF62	Modificar información de las reuniones provinciales.
RF63	Eliminar información sobre las reuniones provinciales.
RF64	Buscar información sobre las reuniones provinciales.
RF65	Modificar reporte de información de las reuniones provinciales.
RF66	Insertar información sobre los cuadros deportivos.
RF67	Modificar información sobre los cuadros deportivos.
RF68	Eliminar información sobre los cuadros deportivos.
RF69	Buscar información sobre los cuadros deportivos.
RF70	Modificar reporte de información sobre los cuadros deportivos.
RF71	Insertar información sobre la estadística de superación.
RF72	Modificar información sobre la estadística de superación.
RF73	Eliminar información sobre la estadística de superación.
RF74	Buscar información sobre la estadística de superación.
RF75	Modificar reporte sobre la estadística de superación.

2.1.2 Requisitos no Funcionales (RNF)

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

RNF1 Restricciones en el diseño y la implementación	Utilizar las herramientas CASE Visual Paradigm para UML en su versión 6.4 y Power Architect en su versión 0.9.13 para el diseño del modelo físico de los datos y la generación de otros artefactos. Utilizar como Gestor de Base de Datos el PostgreSQL en su versión 9.1 o superior. Se trabajará con el IDE Netbeans en su versión 7.0.1 y las librerías del *framework* Hibernate para la implementación de las clases persistentes, archivos de mapeo y las clases controladoras para realizar las funciones de administración de datos sobre la base de datos.
RNF2 de Usabilidad	La capa de acceso brindará gran confiabilidad en cuanto a la información que se desee administrar en la base de datos.
RNF3 de Seguridad	La información será procesada primeramente por la capa de acceso a datos para prevenir errores antes de que los mismos ocurran en la base de datos.

2.2 Diseño e Implementación de la Base de Datos y Capa de Acceso a Datos

Las aplicaciones comúnmente necesitan acceder a datos y las operaciones que impliquen esto pueden suponer un problema, pues la forma de acceder a los datos depende del fabricante y del tipo de almacenamiento al que se está accediendo.

Los componentes de la aplicación deben ser transparentes en la medida de lo posible al actual sistema de persistencia o fuente de datos para permitir migraciones entre distintos fabricantes, distintos tipos de almacenamiento y

diferentes fuentes de datos. Suponiendo que en un momento dado se desee cambiar el motor de persistencia, por medio del modelo DAO será mucho más fácil.

2.2.1 Propuesta de Diseño de Base de Datos

Para la realización de la propuesta de diseño de bases de datos se deben tener en cuenta algunos elementos como: la metodología para el diseño de las bases de datos, el proceso de normalización y los estándares de nomenclatura.

Metodología para el diseño de la Base de Datos

La metodología para el diseño de la base de datos, consta de los siguientes pasos:

1. Determinación de entidades y atributos.
2. Normalización de entidades.
3. Determinación de relaciones (DER).
4. Obtención del modelo lógico global de los datos.
5. Diseño físico de la BD.

Cuando se va a realizar el diseño de la base de datos para un sistema determinado, es necesario determinar los datos que son necesarios tomar en cuenta y las dependencias funcionales existentes entre ellos. (Breidenbach, 2005)

Normalización de los Datos

La normalización se ha desarrollado para obtener estructuras de datos eficientes que eviten las anomalías de actualización, o sea, se basa en la necesidad de encontrar una representación del conjunto de relaciones para que el proceso de actualización sea más adecuado. Es la expresión formal del modo de realizar un buen diseño. Provee los medios necesarios para describir la estructura lógica de los datos en un sistema de información. De forma general presenta las siguientes ventajas:

1. Evita anomalías en la actualización.

2. Mejora la independencia de los datos, permitiendo realizar extensiones de la BD, afectando muy poco, o nada, a los programas de aplicación existentes que acceden la base de datos.

Existen diferentes niveles o fases de normalización, las cuales son: Primera Forma Normal (1FN), Segunda Forma Normal (2FN), Tercera Forma Normal (3FN), Forma Normal de Boyce-Codd (FNBC) y existen además 4FN y 5FN.Cada fase supone que se ha concluido la anterior.

Primera Forma Normal (1FN)

1. Una relación está en (1FN) si cumple la propiedad de que sus dominios no tienen elementos que, a su vez, sean conjuntos.

2. Toda relación normalizada, o sea, con valores atómicos de los atributos.

3. La relación no incluye ningún grupo repetitivo.

Segunda Forma Normal (2FN)

Una relación se dice que está en 2FN si está en 1FN y si, y únicamente si, los atributos no llaves (ni primarias, ni candidatas), son funcional y completamente dependientes de la llave primaria de dicha relación.

Tercera Forma Normal (3FN)

Una relación está en 3FN si está en 2FN y si, y únicamente si, los atributos no llaves son independientes de cualquier otro atributo no llave primaria. Esto es lo mismo que decir que se deben eliminar las dependencias transitivas de atributos no llaves respecto a la llave primaria, estando ya la relación en 2FN. (Matos, 1999)

Estándares de Nomenclatura

Se define la nomenclatura de los objetos con textos descriptivos, de modo que el nombre de los objetos muestre a que se hace referencia.

Tabla 1: Estándares de Nomenclatura.

Objeto	Prefijo	Ejemplo
Base de datos	<nombre>	sigob
Esquemas	mod_<dirección>	mod_cultura,mod_deporte
Tabla de datos	d<nombre><dirección>	dactividadescultura
Tabla de nomencladores	n<nombre><dirección>	nmanifestacionescultura

Descripción de las Entidades

A continuación se describen algunas de las entidades de las Direcciones Cultura y Deporte. Para un mayor entendimiento y ver el diagrama con las respectivas clases que lo componen dirigirse a la Plantilla Modelo Canónico de la dirección correspondiente.

Descripción de Entidades de la Dirección Cultura

Tabla 2: Descripción de la entidad dsaludocuplicotrascultura.

Entidad:	dsaludocuplicotrascultura		
Descripción:	Modelo de Información del MTSS		
Relaciones:	dmodelooacescapcultura		
Campos	Tipo de Dato	Tamaño	Descripción
fecha	Identificador date	10	Indica la fecha del reporte
informante	Identificador varchar	80	Indica el centro informante
periodo	Identificador varchar	80	Indica el periodo del año
indicadores	Identificador varchar	80	Representa diferentes tipos de indicadores
enferm_prof	Integer	10	Indica la cantidad de enfermedades
entidades	Integer	10	Indica la cantidad de entidades
trabaj_expuestos	Integer	10	Indica la cantidad de trabajadores expuestos

Tabla 3: Descripción de la entidad dprocedtrabsegucultura.

Entidad:	dprocedtrabsegucultura		
Descripción:	Modelo de Información del MTSS		
Relaciones:	dmodelooacescapcultura		
Campos	Tipo de Dato	Tamaño	Descripción
fecha	Identificador date	10	Indica la fecha del reporte
informante	Identificador varchar	80	Indica el centro informante
periodo	Identificador varchar	80	Indica el periodo del año
indicadores	Identificador varchar	80	Representa diferentes tipos de indicadores
cantidad	Integer	10	Indica la cantidad por conceptos anteriores

Descripción de Entidades de la Dirección Deporte

Tabla 4: Descripción de la entidad dplanreunionesdeporte.

Entidad:	dplanreunionesdeporte		
Descripción:	Tabla con la información referente a los planes de reuniones		
Relaciones:	dplanesdeporte		
Campos	**Tipo de Dato**	**Tamaño**	**Descripción**
fecha	Identificador date	10	Indica la fecha del plan
cantidad	Integer	10	Indica la cantidad de acuerdo a cada deporte

Tabla 5: Descripción de la entidad dplaneventosdeporte.

Entidad:	dplaneventosdeporte		
Descripción:	Tabla con la información referente a los planes de eventos		
Relaciones:	dplanesdeporte		
Campos	**Tipo de Dato**	**Tamaño**	**Descripción**
fecha	Identificador date	10	Indica la fecha del plan
deportes	Identificador varchar	80	Indica el tipo de deporte
categoría	Identificador varchar	80	Indica la categoría deportiva
tipo	Identificador varchar	80	Indica si es un evento Nacional o Provincial

evento	varchar	80	Indica el tipo de evento
sede	varchar	80	Indica el lugar evento

2.2.2 Solución Propuesta

Desarrollo de la Base de Datos de las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa. Realización de componentes utilizando el patrón CRUD y el patrón DAO.

El patrón CRUD (Create-Read-Update-Delete), conocido como el padre de todos los patrones de capa de acceso. Describe que cada objeto debe ser creado en la base de datos para que sea persistente. Una vez creado, la capa de acceso debe tener una forma de leerlo para poder actualizarlo o simplemente borrarlo.

El patrón DAO (Data Access Object) viene a resolver el problema usando un Objeto de Acceso a Datos para abstraer y encapsular el acceso a los datos. DAO es un método muy simple de mapear objetos a bases de datos. Un DAO maneja la conexión con la fuente de datos para obtener y guardar los datos, siempre realiza operaciones atómicas contra la base de datos y nunca son necesarias las transacciones. Algunos ejemplos de operaciones serían: búsquedas por una clave, creación, actualización y borrado de un registro, obtener todos los registros y cualquier otra operación que se vaya a realizar a menudo.

Normalmente se crea un DAO por cada Objeto que se use en la aplicación. Un objeto puede ser muchas cosas, lo mismo una tabla o una *Entity* de Hibernate. Para generar un DAO el desarrollador podría escribir una clase que contiene un atributo para cada campo de una tabla y una clase Dao para esa tabla que contiene los métodos para la inserción, actualización, selección y eliminación de filas.

2.2.3 Técnicas para crear Capas de Acceso a Datos

Existen distintas formas de encarar la creación de una capa de acceso a datos:

1. Una forma es crear una clase por cada tabla en la base de datos y mapearlas una a una. Esta es la forma más sencilla y es muy válida cuando se tiene una estructura de base de datos que sea muy parecida al modelo de dominio (que no se haya optimizado con particiones horizontales o uniones).
2. Otra forma es respetar un modelo orientado a objetos que represente los objetos del dominio del negocio. Esta es una forma más correcta de trabajar pero se hace más complejo mapear esas clases a sus tablas correspondientes.

Para realizar esta tarea existen herramientas que se llaman ORM (*Object to Relational Mapper*). (Cisneros, 2007)

2.2.4 Ventajas y Desventajas del Data Access Object

Ventajas del DAO:

1. Cualquier objeto no requiere conocimiento directo del destino final de la información que se manipula.
2. Se baja el nivel de acoplamiento entre clases, reduciendo la complejidad de realizar cambios.
3. Se aísla las conexiones a la fuente de datos en una capa fácilmente identificable.
4. Se oculta los detalles de implementación a la fuente de datos.
5. Simple ya puede ser entendido por la mayoría de los desarrolladores.
6. No requiere tiempo de ejecución de contenedores (código DAO puede ser una unidad de prueba en el cliente).
7. El código es muy eficiente, si el DAO está diseñado para aprovechar las capacidades de base de datos (procedimientos almacenados o funciones, *JOINs*, entre otras.)

Desventajas del DAO

1. Muchas veces es necesario acceder a datos que están ubicados en diferentes repositorios de datos.

2. En mucho de los repositorios es necesario tener en la cuenta sus diferentes particularidades.

Las aplicaciones deben poder acceder de forma transparente a estos repositorios. (Cuartas, 2011)

2.2.5 Descripción de las Clases Persistentes y Diagrama de Paquetes

A continuación se describen algunas de las clases persistentes de las Direcciones Cultura y Deporte. Para un mayor entendimiento y ver el diagrama con las respectivas clases que lo componen, dirigirse a la Plantilla Modelo Canónico de la dirección correspondiente.

Descripción de Clases Persistentes de la Dirección Cultura

Tabla 6: Descripción de la clase persistente dsaludocuplicotrascultura.

Clase	dsaludocuplicotrascultura			
Descripción:	Modelo de Información del MTSS			
Tipo	Subclase			
Roles o Propiedades	**Descripción**	**Valor requerido**	**Llave**	**Relación**
fecha	Indica la fecha del reporte	date	X	dmodelooac escapcultura
informante	Indica el centro informante	varchar	X	dmodelooac escapcultura
periodo	Indica el periodo del año	varchar	X	dmodelooac escapcultura
indicadores	Representa diferentes tipos de indicadores	varchar	X	
enferm_prof	Indica la cantidad de enfermedades	Integer	------	------
entidades	Indica la cantidad de entidades	Integer	------	------
trabaj_expue	Indica la cantidad de	Integer	------	------

stos	trabajadores expuestos			

Tabla 7: Descripción de la clase persistente dprocedtrabsegucultura.

Clase	dprocedtrabsegucultura			
Descripción:	Modelo de Información del MTSS			
Tipo	Subclase			
Roles o Propiedades	**Descripción**	**Valor requerido**	**Llave**	**Relación**
fecha	Indica la fecha del reporte	date	X	dmodelooacescapcultura
informante	Indica el centro informante	varchar	X	dmodelooacescapcultura
periodo	Indica el periodo del año	varchar	X	dmodelooacescapcultura
indicadores	Representa diferentes tipos de indicadores	varchar	------	------
cantidad	Indica la cantidad por conceptos anteriores	Integer	------	------

Descripción de Clases Persistentes de la Dirección Deporte

Tabla 8: Descripción de la clase persistente dplanreunionesdeporte.

Clase	dplanreunionesdeporte			
Descripción:	Información referente a los planes de reuniones			
Tipo	Subclase			
Roles o Propiedades	**Descripción**	**Valor requerido**	**Llave**	**Relación**
fecha	Indica la fecha del plan	date	X	dplanesdeporte
cantidad	Indica la cantidad de acuerdo a cada deporte	Integer	------	------

Tabla 9: Descripción de la clase persistente dplaneventosdeporte.

Clase	dplaneventosdeporte
Descripción:	Información referente a los planes de eventos
Tipo	Subclase

Roles o propiedades	Descripción	Valor requerido	Llave	Relación
fecha	Indica la fecha del plan	date	X	dplanesdeporte
deportes	Indica el tipo de deporte	varchar	X	------
categoría	Indica la categoría deportiva	varchar	X	------
tipo	Indica si es un evento Nacional o Provincial	varchar	X	------
evento	Indica el tipo de evento	varchar	------	------
sede	Indica el lugar evento	varchar	------	------

Diagrama de Paquetes

Tanto la dirección de Cultura como la de Deporte forman un módulo compuesto por varias carpetas y subcarpetas entre las cuales está la carpeta "*model*". La misma a su vez contiene la carpeta "*entity*" donde están todas las clases u objetos a persistir y la carpeta "dao" donde se encuentran de uno a varios Objetos de Acceso a Datos necesarios con sus funcionalidades.

En el diagrama siguiente se describe la arquitectura de ambos módulos.

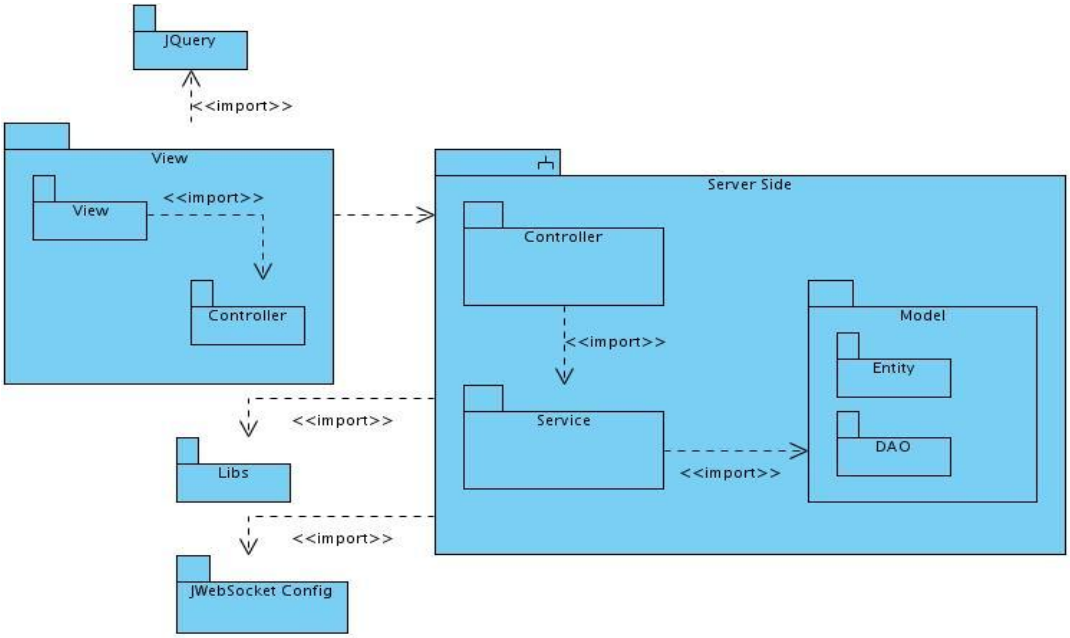
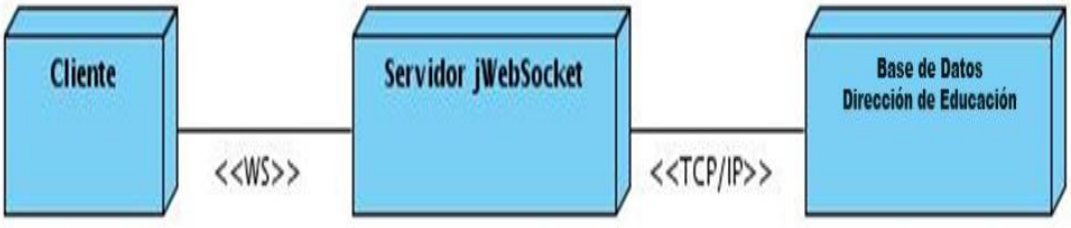


Diagrama de Despliegue

El diagrama de despliegue es utilizado para describir la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Está formado por nodos y las relaciones que se establecen entre ellos. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP y microondas.



Arquitectura de software

Se decidió para el desarrollo de la aplicación, aplicar la arquitectura en n-capas (cuatro capas) debido a que la programación por capas es una arquitectura cliente-servidor que tiene como objetivo principal separar la lógica de negocios de la lógica de diseño. Dicha arquitectura tiene como ventaja que el desarrollo se puede llevar a cabo en varios niveles y en caso de que se requiera algún cambio, se ataca al nivel requerido. Además permite que cada grupo de trabajo esté totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

2.2.6 Implementación de la capa de Acceso

Patrones Utilizados: Patrón n-capas, patrón DAO y patrón CRUD.

Trabajar con *software* orientado a objetos y bases de datos relacionales puede acarrear invertir mucho tiempo en los entornos actuales. Hibernate es una herramienta que realiza el *mapping* entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java y en cuanto a Spring, también ofrece varias opciones para la persistencia de datos.

Por las ventajosas prestaciones que proporcionan, Hibernate y Spring son los *frameworks* seleccionados para la creación de la capa de acceso a datos de las Direcciones Cultura y Deporte del CAP de Artemisa.

Cada dirección tendrá un módulo independiente para la capa de acceso. Cada módulo estará conformado por una serie de ficheros como son: el hibernate-persistence, los *Plain Old Java Objects* (POJOs), ficheros hbm.xml, entre otros.

Configuración de Hibernate

Los datos de configuración de la base de datos son registrados en un fichero hibernate.properties o hibernate-persistence que debe estar en el path de la aplicación. En este fichero se indican los parámetros de conexión de la base de datos como la base de datos a conectar, usuario y *password*.

Plain Old Java Object (POJO)

Hibernate funciona asociando a cada tabla de la base de datos un Plain Old Java Object (POJO, a veces llamado Plain Ordinary Java Object). Un POJO es similar a una Java Bean, con propiedades accesibles mediante métodos setter y getter

Fichero hbm.xml

Para poder asociar el POJO a su tabla correspondiente en la base de datos, Hibernate usa los ficheros hbm.xml. En este tipo de fichero se declaran las propiedades del POJO y sus correspondientes nombres de columna en la base de datos, asociación de tipos de datos, referencias, relaciones con otras tablas, entre otras posibilidades. También otro recurso válido es mapear mediante el método denominado notación, siendo éste el más factible y eficaz método de mapeo.

Hibernate Query Languaje HQL

Hibernate proporciona un lenguaje para realizar consultas a la base de datos. Es similar a SQL y es utilizado para obtener objetos de la base de datos según las

condiciones especificadas en el HQL. El uso de HQL permite usar un lenguaje intermedio que según la base de datos que se use y el dialecto especificado será traducido al SQL dependiente de cada base de datos de forma automática y transparente.

Implementación de clases de control

Una vez que se tiene el archivo de configuración, los POJOs y los ficheros hbm.xml (en caso de no usar notación), solo resta la implementación de las clases de control para el manejo y administración de los datos.

La administración de la información integra básicamente las funcionalidades de creación, eliminación, búsqueda y actualización de registros y ya que cada entidad de la base de datos requiere al menos de estas funcionalidades para su manejo y control, podría ser de gran utilidad el uso del recurso brindado por la genericidad. Utilizando una clase genérica el código ha generar se reduce considerablemente

Conclusiones del Capítulo

En el presente capítulo se plantearon diferentes aspectos relacionados con el diseño de bases de datos e implementación de la capa de acceso a datos, así como la descripción de las técnicas para crear capas de acceso, principales requisitos funcionales del sistema y el uso de los *frameworks* Spring e Hibernate, los cuales proporcionan capacidades para la obtención y almacenamiento de datos reduciendo el tiempo de desarrollo.

Capítulo 3: Adquisición y validación de los resultados del sistema

Introducción del Capítulo

En el presente capítulo se realiza la prueba y validación del diseño e implementación de la capa de acceso a datos de las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa teniendo en cuenta ciertos aspectos como el duplicado e integridad de la información gestionada.

3.1 Integridad, confiabilidad y duplicado de los datos

Integridad de los Datos

La integridad de los datos hace alusión a la corrección y exactitud de la información gestionada. La integridad puede verse afectada de diferentes maneras al realizar modificaciones al contenido de los datos con operaciones de inserción, actualización y eliminación de información, a través de adiciones de datos inválidos y de modificaciones de datos existentes haciendo que tomen valores incorrectos o eliminándolos.

La integridad de datos se refiere a los valores reales que se almacenan y se utilizan en las estructuras de datos de la aplicación. La aplicación debe ejercer un control deliberado sobre todos los procesos que utilicen los datos para garantizar la corrección permanente de la información. En la mayoría de los sistemas actuales, la verificación de la integridad se realiza mediante códigos de procedimientos escritos por los usuarios. (Juan, 2009)

Reglas de Integridad

Una vez definida la estructura de datos del modelo relacional se pasa a estudiar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina *restricciones de dominio*. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias. Estas reglas son la de integridad de entidades y la de integridad referencial.

Reglas de Integridad - Dominio

Un Dominio de valores posibles puede estar asociado a cada atributo. Los límites de Dominio son la forma más elemental de restricciones de Integridad. Son fáciles de probar en el sistema siempre que se introduce un nuevo dato en el sistema. Una definición bien adecuada de restricciones de dominio no solamente permite probar valores insertados en la base de datos. También permite probar consultas para asegurarse de que las comparaciones que se hacen tienen sentido.

Reglas de Integridad - Relación

Las reglas de Integridad de relación son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias, es decir, se deben cumplir todo el tiempo. Existen básicamente dos reglas de Integridad asociadas con el modelo relacional: la *Integridad de Entidad* y la *Integridad Referencial*. Estas dos reglas son generales y tienen relación con las llaves primarias y foráneas.

Reglas de Integridad – Relación (Integridad de Entidad)

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

Las razones de esta regla son:

1. Las tuplas en las relaciones base representan entidades en la realidad.
2. Las entidades en la realidad son identificables por definición.
3. Sus contrapartes en la base de datos también deben ser identificables.
4. Los valores de la clave primaria sirven como identificadores en la base de datos.
5. Los valores de clave primaria no pueden ser nulos.

Reglas de Integridad – Relación (Integridad Referencial)

La regla de Integridad referencial define que la base de datos no debe contener valores de claves foráneas sin concordancia. Esta regla se aplica a las claves foráneas. Si en una relación hay alguna clave foránea, entonces sus valores deben coincidir con los valores de la clave primaria a la que hace referencia, o bien, debe ser completamente nulo.

Así que cuando se realiza una operación ilegal, existen dos opciones: rechazar la operación ilegal o bien aceptar la operación y realizar operaciones adicionales compensatorias que conduzcan a volverla legal. Por lo tanto, para cada clave foránea en la base de datos habrá que contestar a dos preguntas:

1. Regla de los nulos: ¿tiene sentido que la clave foránea acepte nulos?

2. Regla de borrado: ¿Qué ocurre si se intenta borrar la tupla referenciada por la clave foránea?

    1. Restringir: no se permite borrar la tupla referenciada.

    2. Propagar: se borra la tupla referenciada y se propaga el borrado a las tuplas que hacen referencia mediante la clave foránea.

    3. Anular: se borra la tupla referenciada y las tuplas que la referenciaban indicando un valor nulo a la clave foránea (únicamente si acepta nulos).

La Integridad referencial también vela por el cumplimiento de las siguientes reglas:

1. No se podrá introducir un valor en la tabla relacionada si antes no ha sido introducida en la tabla principal.

2. No se puede eliminar un registro de una tabla principal si existen registros coincidentes en la tabla relacionada.

3. No se puede cambiar un valor de la clave primaria en la tabla principal si el registro tiene registros relacionados. (Juan, 2009)

Confiabilidad de los Datos.

La confiabilidad se puede interpretar de varias formas. La confiabilidad se puede ver como una medida con la cual un sistema conforma su comportamiento a alguna especificación. También se puede interpretar como la probabilidad de que un sistema no haya experimentado ninguna falla dentro de un periodo de tiempo dado. La confiabilidad se utiliza típicamente como un criterio para describir sistemas que no pueden ser reparados o donde la operación continua del sistema es crítica.

Duplicado de los datos

El duplicado de datos no es más que el almacenamiento repetido de los mismos. Esto puede significar un inconveniente al realizar modificaciones en los datos, siendo el elemento más frecuente de inconsistencias. Necesita además, mayor espacio de almacenamiento lo cual puede aumentar los costes de almacenamiento y acceso a los datos.

3.2 Ejecución de pruebas

Para garantizar el manejo y persistencia de la información de forma correcta se realizaron un conjunto de pruebas.

Verificación de la Conexión

Se realizaron comprobaciones al archivo de configuración (hibérnate-persistence) que necesita los *frameworks* Spring e Hibernate para la conexión con la base de datos.

Realización de las Pruebas

Se pusieron en práctica pruebas para comprobar la capacidad funcional de los métodos declarados en la capa de acceso a datos verificando de esta forma el cumplimiento de los requisitos funcionales. Para la comprobación se utilizó una clase principal en java, óptima para este propósito. En la misma fueron probadas las diferentes operaciones de los métodos de la capa de acceso como: inserción, modificación y eliminación de registros.

Las siguientes tablas muestran los aspectos que se tuvieron en cuenta en la realización de las pruebas a los métodos para la persistencia de objetos.

Tabla 10: Caso de Prueba 1.

Método:	save
Condiciones:	Debe existir un objeto de la clase entidad especificada y/u otros objetos que estén relacionados con dicha entidad.

Objetivo	Clases Válidas	Resultado Esperado	Resultado de la Prueba
Insertar un objeto de la clase entidad especificada correspondiente a la tabla o entidad persistente en la base de datos.	Cada una de las Propiedades del objeto u objetos.	Se espera que el objeto sea insertado en la tabla y columnas correspondientes en la base de datos.	El objeto fue insertado satisfactoriamente. Se insertaron cada uno de los valores definidos a las propiedades del objeto, en cada una de las columnas de la tabla correspondiente a la clase entidad en la base de datos.

Tabla 11: Caso de Prueba 2.

Método:	delete		
Condiciones:	Debe existir un objeto de la clase entidad especificada.		
Objetivo	**Clases Válidas**	**Resultado Esperado**	**Resultado de la Prueba**
Eliminar un objeto de la clase entidad especificada correspondiente a la tabla o entidad persistente en la base de datos.	Objeto de la clase entidad especificada.	Se espera que el objeto sea eliminado de la tabla correspondiente a la clase entidad especificada, en la base de datos.	1. El objeto fue eliminado satisfactoriamente. 2. Se elimino la tupla especificada por el objeto en la tabla de la base de datos correspondiente a la clase entidad. 3. Para el caso de las entidades con instancias asociadas a otras entidades a las cuales se le especificó el atributo cascade

			fueron eliminadas de igual manera de cada una de las tablas o entidades persistentes correspondientes a éstas. 4. Para el caso en que el objeto no existe en la base de datos, el método lanzó

Tabla 12: Caso de Prueba 3.

Método:	findAll		
Condiciones:	Debe existir una Lista con los objetos de la clase entidad especificada.		
Objetivo	**Clases Válidas**	**Resultado Esperado**	**Resultado de la Prueba**
Obtener un listado de la clase entidad especificada correspondiente a la tabla o entidad persistente en la base de datos.	Clase entidad especificada.	Se espera obtener un listado con todas las tuplas de la clase entidad especificada que se encuentren almacenadas en la tabla correspondiente a ésta, en la base de datos.	1. Se obtuvo el listado satisfactoriamente. 2. Se listaron todas las tuplas existentes de la clase entidad especificada, almacenadas en la tabla correspondiente a ésta, en la base de datos.

Tabla 13: Caso de Prueba 4.

Método:	findById

Condiciones:	Debe existir un objeto de la clase entidad especificada, con un identificador.		
Objetivo	**Clases Válidas**	**Resultado Esperado**	**Resultado de la Prueba**
Buscar un objeto por el identificado (Integer) de clase entidad especificada correspondiente a la tabla o entidad persistente en la base de datos.	1. Clase entidad especificada. 2. Identificador (Integer)	1. Se espera obtener el objeto con el identificador de la clase entidad especificada. 2. En caso de que no se encuentre el identificador, se espera que devuelva Null.	1. Se obtuvo el objeto con el identificador dado satisfactoriamente. 2. No encontró el identificador. 3. El método devolvió el valor Null (la prueba continúa siendo satisfactoria).

Tabla 14: Caso de Prueba 5.

Método:	findById		
Condiciones:	Debe existir un objeto de la clase entidad especificada, con un identificador.		
Objetivo	**Clases Válidas**	**Resultado Esperado**	**Resultado de la Prueba**
Buscar un objeto por el identificador (Object) de clase entidad especificada correspondiente a la tabla o entidad persistente en la base de datos.	1. Clase entidad especificada. 2. Identificador (Object)	1. Se espera obtener el objeto con el identificador de la clase entidad especificada. 2. En caso de que no se encuentre el identificador, se espera que devuelva Null.	1. Se obtuvo el objeto con el identificador dado satisfactoriamente. 2. No encontró el identificador. 3. El método devolvió el valor Null (la prueba continúa siendo satisfactoria).

Tabla 15: Caso de Prueba 6.

Método:	findByParamsAndValues		
Condiciones:	Debe existir un objeto de la clase entidad especificada y una lista de parámetros de búsqueda.		
Objetivo	**Clases Válidas**	**Resultado Esperado**	**Resultado de la Prueba**
Buscar los objetos persistentes por los valores o subcadenas de los valores para los parámetros de búsqueda.	1. Parámetros. 2. Valores o subcadenas de los valores.	Se espera recuperar los objetos persistentes para los valores o subcadenas de estos valores definidos para cada parámetro de búsqueda.	1. La búsqueda resultó satisfactoria. 2. Devolvió los objetos persistentes con los valores o subcadena del valor para los parámetros de búsqueda definidos.

Tabla 16: Caso de Prueba 7.

Método:	findByParamAndValue		
Condiciones:	Debe existir un objeto de la clase entidad especificada y un parámetro de búsqueda.		
Objetivo	**Clases Válidas**	**Resultado Esperado**	**Resultado de la Prueba**
Buscar los objetos persistentes por para el parámetro de búsqueda.	1. Parámetro. 2. Valor.	Se espera recuperar los objetos persistentes para el valor definido para el parámetro de búsqueda.	1. La búsqueda resultó satisfactoria. 2. Devolvió los objetos persistentes para el valor definido para el parámetro de búsqueda.

Los métodos de prueba fueron realizados para cada entidad de la base de datos. Algunas de las pruebas realizadas a la capa de acceso correspondiente al módulo de la Dirección Cultura de la Administración Provincial de Artemisa pasándole valores arbitrarios fueron las siguientes:

```
public class prueba {

    private static GenericDao obj;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
       obj=GenericDao.GetIntance();
     Addaammaotrasmanifest ();//Se le hace un llamado a las
funciones con la intención de probar su eficacia

    }
    public static void Addaammaotrasmanifest()//Agregar
registros de tipo Daammaotrasmanifestcultura
    {
         Date fecha = new Date(2012, 2, 20);
        DcategorizacionculturaId iduno = new
DcategorizacionculturaId(2, fecha);
        Ddepartamentocasaculturacultura dpto = new
Ddepartamentocasaculturacultura(2);
        Dcategorizacioncultura uno = new
Dcategorizacioncultura(iduno, dpto);
        DuammaculturaId iddos = new DuammaculturaId("1", 2,
fecha);
        Duammacultura dos = new Duammacultura(iddos);
        DaammaotrasmanifestculturaId idm=new
DaammaotrasmanifestculturaId("1", 2, fecha, "musica");
        Daammaotrasmanifestcultura x=new
Daammaotrasmanifestcultura(idm, 3, 4);
        obj.saveOrUpdate(dpto);
        obj.saveOrUpdate(uno);
        obj.saveOrUpdate(dos);
        obj.saveOrUpdate(x);

}
Public void Deleteaammaotrasmanifest()//Borrar registros de
tipo daammaotrasmanifestcultura
{
 Date fecha = new Date(2012, 2, 20);
 DaammaotrasmanifestculturaId idm=new
DaammaotrasmanifestculturaId("1", 2,     fecha, "musica");
 Daammaotrasmanifestcultura x=new
```

```
Daammaotrasmanifestcultura(idm, 3, 4);
Obj.delete(x);
}
Public void Deleteuamma()//Borrar registros de tipo
duammacultura
{
Date fecha = new Date(2012, 2, 20);
DuammaculturaId iddos = new DuammaculturaId("1", 2, fecha);
Duammacultura dos = new Duammacultura(iddos);
Obj.delete(dos);
}
Public
<Daammaotrasmanifestcultura>Showaammaotrasmanifest()//Mostrar
elementos de tipo Daammaotrasmanifestcultura
{
List<Daammaotrasmanifestcultura>
result=dao.find(Daammaotrasmanifestcultura.class);
Return result;
}
```

A manera de explicación se describe el siguiente fragmento de código:

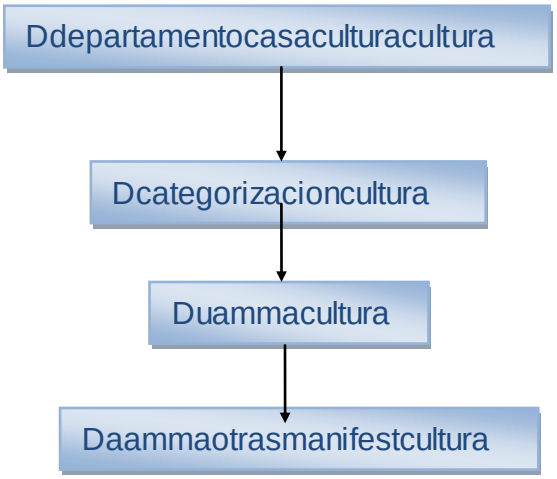
```
public static void main(String[] args)
{
  obj=GenericDao.GetIntance();
  Addaammaotrasmanifest ();//Se le hace un llamado a las funciones con la
intención de probar su eficacia
}
```

Este constituye el método principal de una clase java en consola. En su cuerpo se inicializa una instancia del GenericDao para acceder a sus funcionalidades. Por último se le hace un llamado a una función de nombre Addaammaotrasmanifest que es la contenedora de alguna operación que se desea someter a prueba.

La función Addaammaotrasmanifest se describe a continuación:

```
public static void Addaammaotrasmanifest
{
Date fecha = new Date(2012, 2, 20);
DcategorizacionculturaId iduno = new DcategorizacionculturaId(2, fecha);
Ddepartamentocasaculturacultura dpto = new
Ddepartamentocasaculturacultura(2);
Dcategorizacioncultura uno = new Dcategorizacioncultura(iduno, dpto);
DuammaculturaId iddos = new DuammaculturaId("1", 2, fecha);
Duammacultura dos = new Duammacultura(iddos);
DaammaotrasmanifestculturaId idm=new DaammaotrasmanifestculturaId("1", 2,
fecha, "musica");
Daammaotrasmanifestcultura x=new Daammaotrasmanifestcultura(idm, 3, 4);
obj.saveOrUpdate(dpto);
obj.saveOrUpdate(uno);
obj.saveOrUpdate(dos);
obj.saveOrUpdate(x);
}
```

Esta función comprobará si realmente puede añadirse un registro a la tabla de nombre daammaotrasmanifestcultura. Primeramente se debe tener claro el factor de la herencia para que no ocurran violaciones de llaves. Antes de agregar algo en daammaotrasmanifestcultura debe verificarse si los campos de tipo llave foránea existan en su tabla padre de nombre duammacultura y con esta a su vez se debe realizar la misma operación. La jerarquía seria la siguiente:



Por esta razón se crean objetos de cada tipo y se le hace una llamada al método de la clase GenericDAO llamado saveOrUpdate el cual verifica la existencia del objeto y de no existir la agrega y en el caso de que exista la modifica.

3.3 Resultados y Funcionalidades Obtenidas y Aporte Social y Económico

Resultados y Funcionalidades Obtenidas

A partir de la Propuesta de Solución queda disponible la Base de Datos para las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa, con su respectiva capa de acceso, la cual tiene como funcionalidades principales insertar, buscar, modificar y eliminar cualquier tipo de información referente a dichas direcciones, garantizando la integridad, confiabilidad, centralidad y disponibilidad en cuanto al manejo de la información.

Aporte Social y Económico

Por medio de la Base de Datos para las Direcciones Cultura y Deporte de la Administración Provincial de Artemisa toda la información que se maneja en dichas direcciones estará centralizada permitiendo un correcto almacenamiento de la información. Esta base de datos tiene la ventaja económica que es realizada en Cuba, es decir no usa un *software* propietario que haya que pagar para su uso, ni es necesario invertir gran cantidad de dinero contratando una serie de desarrolladores para la realización de la misma. Esta puede ser implantada en cualquier provincia o municipio que estén necesitando de ella sin necesidad de pagar para ser utilizada.

Conclusiones del Capítulo

En este capítulo se presentaron un conjunto de aspectos para realizar la validación práctica de la solución propuesta. Con la validación de la capa de acceso se pusieron a prueba una serie de elementos como el control ante duplicados de

información, el manejo de datos y tipos de datos y el nivel o capacidad de recuperación frente a fallos o excepciones. Por medio de las pruebas realizadas se pudo tener conocimiento de cómo respondería la capa de acceso a datos ante determinadas situaciones. El comportamiento fue normal y esperado obteniendo resultados satisfactorios.

Conclusiones

Durante el desarrollo de la base de datos y su correspondiente capa de acceso se adquirieron un conjunto de conocimientos que posibilitaron su avance. Concluido el presente trabajo de investigación se puede afirmar que se ha cumplido con el objetivo del mismo a través de la propuesta de diseño de base de datos y la implementación de la capa de acceso a datos para las Direcciones de Cultura y Deporte, lo cual permitió centralizar la información y asegurar su confiabilidad e integridad. Para darle solución al problema planteado en la investigación, se realizaron cada una de las tareas planteadas. Terminada la investigación se tienen las siguientes conclusiones:

1. Durante la investigación se realizó un estudio acerca de los elementos teóricos principales relacionados con el diseño de bases de datos que favoreció el desarrollo del trabajo permitiendo sentar las bases para un correcto diseño de bases de datos.

2. Se realizó el diseño de la base de datos, el cual cumple con los requerimientos especificados.

3. Se implementó la capa de acceso a datos, lo cual permitió establecer la estructura y operaciones adecuadas de almacenamiento y recuperación de la información.

4. Se analizó el comportamiento de la capa de acceso a datos y de la base de datos ante situaciones determinadas a través de diferentes pruebas, a las cuales respondió satisfactoriamente.

De manera general se obtuvo una base de datos que permite el almacenamiento y recuperación de la información requerida, permitiendo mayor organización y control de la misma.

Recomendaciones

El presente trabajo de diploma propone una solución que contribuye al almacenamiento y recuperación de la información perteneciente a las Direcciones Cultura y Deporte, garantizando la disponibilidad, integridad y confiabilidad de los datos, por lo que se recomienda:

1. Implementar los restantes servicios para los módulos de las Direcciones Cultura y Deporte, utilizando los componentes desarrollados para la persistencia de objetos, y las funciones de optimización definidas.

2. Refinar o incluir nuevas funcionalidades para la persistencia de objetos.

3. Aplicar las concepciones adoptadas para el desarrollo de la capa de persistencia en otros proyectos que utilicen las mismas tecnologías e incluso en otras plataformas de desarrollo.

Referencias Bibliográficas

Albiol, Francesc Rosés. 2003. *Introducción a Hibernate.* 2003.
Alfredo. 2011. Illinois. [Online] 2011. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.
Amaya, Rodrigo. 2009. Sr.Byte. [Online] 09 02, 2009. www.srbyte.com/2009/09/que-es-orm.html.
basesdedatos.org. 2010. Base de Datos. [Online] 2010. http://www.basesdedatos.org/.
Breidenbach, Craig Walls Ryan. 2005. *Spring in Action.* 2005.
CAVSI.com. 2010. cavsi.com. [Online] 2010. http://www.cavsi.com/preguntasrespuestas/.
Cisneros, Jonathan. 2007. Cisneros. [Online] 09 24, 2007. http://cisneros.wordpress.com/2007/09/24/creando-una-capa-de-acceso-a-datos/.
Cuartas, Jose. 2011. Patrones de arquitectura Software. [Online] 06 06, 2011. http://www.slideshare.net/josecuartas/patrones-de-arquitectura-softwarecapa-de-datos.
Curto, Josep. 2006. Information Management. [Online] 11 28, 2006. http://informationmanagement.wordpress.com/.
Definicion.de. 2008. definicion.de. [Online] 2008. http://definicion.de/informacion/.
—. 2008. definicion.de. [Online] 2008. http://definicion.de/gestion/.
—. 2009. definicion.de. [Online] 2009. http://definicion.de/sql/.
Ercoli, Jorge. 2007. Arquitectura de Sistemas Informáticos. [Online] 10 31, 2007. http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.
Halpin, Terry. 2011. Object Role Modeling. [Online] 10 01, 2011. http://www.orm.net.
http://www.efdeportes.com/efd62/redes.htm. [Online]
http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5017/msoft08.htm. [Online]
Juan. 2009. Integridad de las Bases de Datos. *Integridad de las Bases de Datos.* [Online] 09 19, 2009. [Cited: 02 12, 2012.] http://juanin.bligoo.com/view/606209/Integridad de las Bases de Datos.html.
Marrero, Belina Capote. 2003. Revistas. [Online] 01 16, 2003. http://www.bvs.sld.cu/revistas/aci/vol11_2_03/aci030203.htm.
Martinez, Rafael. 2010. PostgreSQL-es. [Online] 10 02, 2010. http://www.postgresql.org.es/sobre_postgresql.
Matos, Lic. Rosa María. 1999. *Diseño de Bases de Datos.* 1999.
—. 1999. *Sistemas de Bases de Datos.* 1999.
netbeans.org. 2010. netbeans.org. [Online] 07 28, 2010. http://netbeans.org/community/releases/69/relnotes_es.html.
PgAdminIII. 2008. PgAdmin III. [Online] 03 10, 2008. http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.
R.Ernando. 2008. Metodologías de desarrollo de software. [Online] 2008. http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html.
SOA-agenda. 2007. SOA agenda. [Online] 2007. http://soaagenda.com/journal/articulos/.
Solano. 2009. Modulado 3.2. [Online] 02 20, 2009. http://modulado.wordpress.com/2009/02/20/modelos-entidad-relacion-con-power-architect/.
Thompson, Ivan. 2008. Mercadotecnia. [Online] 10 2008. http://www.promonegocios.net/mercadotecnia/que-es-informacion.html.

Valdés. 2007. Maestros del Web. [Online] 10 26, 2007. http://www.maestrosdelweb.com/principiantes/.
visual-paradigm.com. 2007. visual-paradigm.com. [Online] 2007. http://www.visual-paradigm.com.

Bibliografía

1. **Albiol, Francesc Rosés. 2003.** *Introducción a Hibernate.* 2003.
2. **Alfredo. 2011.** Illinois. [Online] 2011. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.
3. **Amaya, Rodrigo. 2009.** Sr.Byte. [Online] 09 02, 2009. www.srbyte.com/2009/09/que-es-orm.html.
4. **basesdedatos.org. 2010.** Base de Datos. [Online] 2010. http://www.basesdedatos.org/.
5. **Beck,K.Extreme.** Programming Explained. Embrace Change. Pearson Education. 1999. (Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio". Addison Wesley. 2000)
6. **Breidenbach, Craig Walls Ryan. 2005.** *Spring in Action.* 2005.
7. **CAVSI.com. 2010.** cavsi.com. [Online] 2010. http://www.cavsi.com/preguntasrespuestas/.
8. **Cisneros, Jonathan. 2007.** Cisneros. [Online] 09 24, 2007. http://cisneros.wordpress.com/2007/09/24/creando-una-capa-de-acceso-a-datos/.
9. **Controlchaos**. http://www.controlchaos.com [Online]
10. **Cuartas, Jose. 2011.** Patrones de arquitectura *Software*. [Online] 06 06, 2011. http://www.slideshare.net/josecuartas/patrones-de-arquitectura-softwarecapa-de-datos.
11. **Curto, Josep. 2006.** Information Management. [Online] 11 28, 2006. http://informationmanagement.wordpress.com/.
12. **Definicion.de. 2008.** definicion.de. [Online] 2008. http://definicion.de/informacion/.
13. **—. 2008.** definicion.de. [Online] 2008. http://definicion.de/gestion/.
14. **—. 2009.** definicion.de. [Online] 2009. http://definicion.de/sql/.
15. **Ercoli, Jorge. 2007.** Arquitectura de Sistemas Informáticos. [Online] 10 31, 2007. http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.
16. **Extremeprogramming.** http://www.extremeprogramming.org [Online]
17. **Halpin, Terry. 2011.** Object Role Modeling. [Online] 10 01, 2011. http://www.orm.net.
18. **http://www.efdeportes.com/efd62/redes.htm.** [Online]
19. **http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5017/msoft08.htm.** [Online]
20. **Juan. 2009.** Integridad de las Bases de Datos. *Integridad de las Bases de Datos.* [Online] 09 19, 2009. [Cited: 02 12, 2012.] http://juanin.bligoo.com/view/606209/Integridad de las Bases de Datos.html.
21. **Kioskea**. http://es.kioskea.net/contents/bdd/bddintro.php3 [Online]
22. **Lawebdelprogramador.** http://www.lawebdelprogramador.com/ [Online]
23. **Marrero, Belina Capote. 2003.** Revistas. [Online] 01 16, 2003. http://www.bvs.sld.cu/revistas/aci/vol11_2_03/aci030203.htm.
24. **Martinez, Rafael. 2010.** PostgreSQL-es. [Online] 10 02, 2010. http://www.postgresql.org.es/sobre_postgresql.

25. **Mat Raible**. Spring in Live.2004
26. **Matos, Lic. Rosa María. 1999.** *Diseño de Bases de Datos.* 1999.
27. **—. 1999.** *Sistemas de Bases de Datos.* 1999.
28. **Monografias.com.** http://www.monografias.com/ [Online]
29. **netbeans.org. 2010.** netbeans.org. [Online] 07 28, 2010. http://netbeans.org/community/releases/69/relnotes_es.html.
30. **PgAdminIII. 2008.** PgAdmin III. [Online] 03 10, 2008. http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.
31. **Programming.** http://www.xprogramming.com [Online]
32. **R.Ernando. 2008.** Metodologías de desarrollo de software. [Online] 2008. http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html.
33. **Ramón Ruiz** .*El Método Científico y sus Etapas*. México 2007.
34. **Rincondelvago.**http://www.rincondelvago.com/ [Online]
35. **SOA-agenda. 2007.** SOA agenda. [Online] 2007. http://soaagenda.com/journal/articulos/.
36. **Solano. 2009.** Modulado 3.2. [Online] 02 20, 2009. http://modulado.wordpress.com/2009/02/20/modelos-entidad-relacion-con-power-architect/.
37. **Thompson, Ivan. 2008.** Mercadotecnia. [Online] 10 2008. http://www.promonegocios.net/mercadotecnia/que-es-informacion.html.
38. **Valdés. 2007.** Maestros del Web. [Online] 10 26, 2007. http://www.maestrosdelweb.com/principiantes/.
39. **visual-paradigm.com. 2007.** visual-paradigm.com. [Online] 2007. http://www.visual-paradigm.com.
40. **Wikipedia.** http:// es.wikipedia.org/ [Online]

Glosario de Términos

1. **Java:** Es un lenguaje de programación orientado a objetos.
2. **PostgreSQL:** Es un sistema de gestión de base de datos relacional orientada a objetos y libre.
3. **SGBD** (Sistema de Gestor de Base de Datos): Conjunto coordinado de programas, procedimientos, lenguajes. Herramienta destinada a la creación y manipulación de bases de datos.
4. **ORM:** Mapeador de Objetos Relacionales.
5. **BD:** Base de Datos