Universidad de las Ciencias Informáticas

Facultad 10

Título:

Perfeccionamiento de la capa de acceso a datos del HelpDesk.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Tutor:

Ing. Efraín Losada León

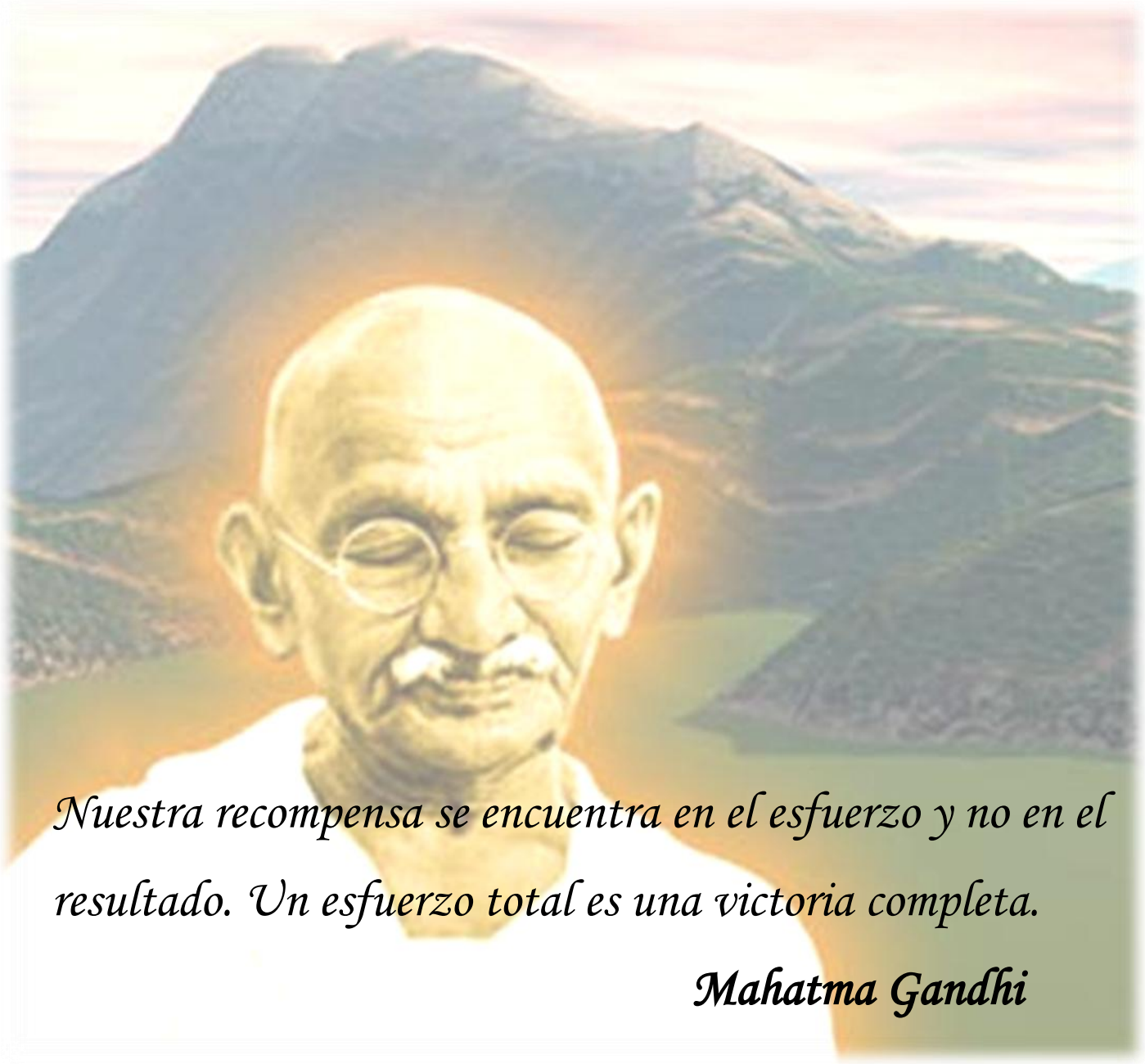
Autores:

Anier Lima Mena

Yanara Enamorado Zamora

La Habana, junio 2010

"Año 52 de la Revolución"



Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.

Mahatma Gandhi

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Tutor

_____

Ing. Tte. Efraín Losada León

Autores

_____ _____

Yanara Enamorado Zamora **Anier Lima Mena**

Datos de Contacto

Ing. Tte. Efraín Losada León

Correo electrónico: elosada@uci.cu

Curso 2008-2009.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Yanara Enamorado Zamora

Correo electrónico: yezamora@estudiantes.uci.cu

Curso 2008-2009.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Anier Lima Mena

Correo electrónico: alima@estudiantes.uci.cu

Curso 2008-2009.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Agradecimientos

Hacer realidad este sueño no hubiese sido posible sin el apoyo de muchas personas durante estos 5 años. Queremos agradecer a todos los que de una forma u otra han colaborado, entre ellos: profesores, amigos, familiares, compañeros y hasta confidentes. Al tribunal por guiarnos y darnos una nueva enseñanza en cada corte, al tutor por estar siempre dispuesto a ayudarnos, por su alegría y optimismo. A todos ellos muchas gracias. En especial Agradecemos:

Yanara: *Hay tiempos buenos y tiempos malos, pero siempre existirá una luz que nos guíe en esos momentos de oscuridad. A todas aquellas personas que han sido luz, en mis tantos momentos oscuros durante estos años.*

A mis abuelos: Enrique *por haberme guiado siempre por el buen camino, por ser tan recto y formal, por mimarme y educarme 20 años de mi vida.*

Teresa *por su apoyo, comprensión y confianza, por su orgullo al mirarme y nombrarme, por haberme regalado parte de su juventud y toda su vida dándome las fuerzas para continuar. Por ser tan linda, tan próspera y activa. A ustedes por formar lo que hoy soy a ambos gracias por haberme criado con amor y cariño donde quiera que estén.*

A mi madre querida *por estar siempre para apoyarme y darme todas las fuerzas del mundo, por enseñarme que el que persevera triunfa, por su comprensión y cariño, por aceptarme rebelde como soy, por aguantarme, consentirme y escucharme siempre. Por hacerme confiar en mí misma y por ser tan optimista.*

AGRADECIMIENTOS

A mi padre *por sus consejos y afectos y por el número de hermanos.* ***A mis tíos*** *Evelio y Luisito por ayudarme, alegrarme y aconsejarme siempre.*

A mi padrastro: *Juan José por ayudarme, aconsejarme y darme cariño como a una hija más.*

A Enrique y Edilberto *por haber sido siempre como un familiar para mí, por su cariño y apoyo.*

A mi hermano *Alexander por siempre decirme las palabras exactas en el momento preciso, por ayudarme y comprenderme. Por sus presentes muchas gracias.*

A mi primo Rolando *y a su esposa por apoyarme y facilitarme su ayuda.*

A mis buenos vecinos*: Dona, Idelia y Marielena.*

A mis compañeros de brigada *del 6103 (grupo de 1er año) con los cuales compartí aquella época de inocencias y novatadas, por ser tan unidos y tan familiares a todos gracias. En especial Yaima y Milagro por enseñarme a luchar por las cosas que se quieren en la vida. Betty y Massiel por ser tan buenas e ingeniosas. Macora por ser mi primer amigo en esta universidad. Yadira, Eilen (la flaca) por haber sido las mejores compañeras de apto y Susy por ser tan noble, sincera y muy valiente por afrontar la vida como quiere.*

A Andrés *(Man)* ***mi mejor amigo****, que me ha soportado durante todos estos años de universidad en el mismo grupo, por ser mi fiel compañero gracias. A Uds. Por haber sido de los primeros y los últimos.*

A la brigada 6201 *por haberme aceptado como una más del grupo, por las fiestas y viajes a la playa gracias a todos y en especial Sungui, Alice, Yaneisi, Wilmar, Bola y Rigui.*

AGRADECIMIENTOS

A la brigada 6403: *a los Militares muchas gracias por elegirme como jefa de brigada y hacerme quedar bien, por estar siempre dispuestos. En especial. PeterD, Bismat, JDayro, Ivette, Yorker, Man,Yorlandy.*

Al apto 76103: *Yuya (patrulla), Elisa, Yanet, Haymel, la Yumi, Marisel, Aliekna (la tora) por haber estado ahí para conversar y arreglar el mundo con nuestras pláticas y por sacarme de varios aprietos, agradezco a todos su aprecio y confianza.*

A Taty *por brindarme su hombro tantas veces para llorar y reír, por ser noble, sincera y por aconsejarme tantas veces en tan poco tiempo.*

A Imara o Zukibel *por estar dispuesta tantas veces a saltar al vacío conmigo en los momentos duros, por seguirme.*

A mis amigas del PRÉ *por escucharme y aconsejarme* ***Vallejo, Nelly*** *y en especial a* ***Ariagna*** *por ser la amiga perfecta que me hace reflexionar sin criticarme.*

A mi gente fuera de las brigadas*: en especial MariselS, Lien, Lissette Lacea, Dainelis, Marlies, Raydel, Evelyn, Yasmo e Iriobe quien se ganó mi aprecio y a quien quiero mucho por alentarme siempre y Sergio la butaca por su amistad.*

Al Acro *por su paciencia y dedicación, por su esfuerzo y sus conocimientos aportados.*

Michel *por su amistad, cariño y paciencia al aportarme sus conocimientos, por ser un buen amigo y consejero.*

Richard el toro *por su amistad, aportes y revisiones en la tesis.*

AGRADECIMIENTOS

A mi gente del judo: profes José Luis *por su esfuerzo en mejorar mi cuerpo con sus ejercicios, por entrenarme tantas veces y apoyarme en los combates.*

Censei Larrude *a quien quiero mucho por su dedicación, confianza y amistad. Por estar siempre ahí para darme ánimos siempre que me deprimo. Por su cariño.*

A ambos muchas Gracias por aguantarme desde primer año y alegrarme tantos días con sus bromas.

A mi tutor *por su labor para que esta tesis se realizara al máximo por sus aportes y conocimientos.*

A mi dúo *que a pesar de que no nos conocíamos, hoy somos compañeros, por acompañarme en este viaje final gracias.*

A Enrique Y Freya *por su confianza y preocupación, por haber sido en estos 3 años como una familia para mí, por su apoyo y por haber engendrado a alguien que es y será siempre tan especial para mí.*

A Yoel Carreño *por haber sido mi alegría, por sus mimos, por haberme soportado, acompañado, ayudado y malacostumbrado en estos 3 años sin duda han sido los mejores, por haberme enseñado que la vida es una, que el orgullo es una de las cosas más destructivas que puedan tener dos personas que se quieren, que nadie es completamente malo, ni completamente bueno que todos tenemos un poco de todo. Que nada está seguro y que todo lo que tienes hoy a tu favor en un segundo se puede virar en tu contra. Por haberme hecho el favor de siendo tan mango querer a una fea como yo. Gracias mi neni.*

DEDICATORIA

Dedicatoria

Al Ministerio del Interior y la revolución por permitir esta formación.

A mis padres abuelos Teresa y Enrique quienes me dieron todo su amor, por ser mi fuente de inspiración.

A mi madre Elisa por su comprensión y apoyo.

A mis hermanos José Luis y Yaimara para que tengan un ejemplo a seguir.

A los amigos que siempre creyeron y a los que no creyeron para que vean que si se puede.

A todos los que depositaron su confianza en mí, a todos los que de una u otra forma me apoyaron quiero dedicar este trabajo de diploma.

Yanara

Resumen

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios universitarios nacido como un proyecto de la Revolución Cubana, denominado en sus inicios "Proyecto Futuro". Posee como objetivos fundamentales: la informatización del país y el desarrollo de la Industria del Software. Simultáneamente en este centro de estudios universitarios el Ministerio del Interior(MININT) cuenta con un gran número de estudiantes los cuales están vinculados a proyectos que se implementan hoy para el MININT entre ellos: el proyecto Escudo, Prisiones, Aduana, Identidad entre otros. Actualmente se desarrolla una aplicación con el objetivo de gestionar y controlar las interrupciones que se puedan presentar en los medios y servicios que brindan las tecnologías de la información y las comunicaciones (TIC).

La capa de acceso a datos presenta algunos problemas como:

- La base de datos (BD) no está normalizada.
- Algunas conexiones se quedan abiertas.
- No existe un control adecuado de las transacciones que se realizan.
- Muchas de las herramientas empleadas no son las más actualizadas.
- Dificultad para que los desarrolladores den soporte a la BD.

El presente trabajo de diploma tiene como objetivo mejorar los módulos existentes con una herramienta de mapeo de objeto relacional (ORM) que contribuya a perfeccionar la capa de acceso a los datos del sistema HelpDesk.

PALABRAS CLAVE

Base de datos, Gestión, HelpDesk, Interrupciones, ORM

ÍNDICE

ÍNDICE

INTRODUCCIÓN

El crecimiento y desarrollo tecnológico implica un nivel de informatización social elevado, por lo que las empresas y organizaciones mundiales se han visto en la necesidad de buscar nuevas soluciones que les ayude a gestionar los recursos que poseen. Entre las soluciones utilizadas se encuentran los sistemas HelpDesk, que son aplicaciones desarrolladas con el objetivo de brindar información y asistencia para solucionar problemas que puedan presentar los medios y servicios de las Tecnologías de la Informática y las Comunicaciones (TIC).

Las corporaciones a menudo proveen soporte HelpDesk a sus consumidores vía número telefónico gratuito, website o e-mail y de forma interna para los propios empleados. Muchas de estas aplicaciones tienen en la implementación de sus capas de datos el uso de la técnica ORM lo cual facilita el trabajo con las BD.

En Cuba aunque el avance tecnológico es gradual con respecto al mundo, también existen aplicaciones con estos fines en las diversas ramas de la sociedad. En la oficina central de Copextel que radica en la UCI se utiliza un sistema HelpDesk para controlar y monitorear la calidad y cantidad de reparaciones realizadas a las computadoras y a los servicios de red que utiliza todo el personal involucrado.

El MININT cuenta con numerosos y modernos medios que facilitan la realización de su tarea diaria por lo que se hace necesario la gestión y el control de cualquier falla que pueda ocurrir en estos medios. En el módulo UCI-MININT se desarrolló una aplicación que tiene como **objetivo** gestionar y controlar las interrupciones que puedan ocurrir en los medios y servicios que brindan las TIC.

El sistema presenta algunas deficiencias en la capa de acceso a datos, estos son: la BD no está normalizada, algunas conexiones se quedan abiertas, no existe un control adecuado de las transacciones que se realizan, las tecnologías empleadas no son las más actualizadas y existe dificultad para que los desarrolladores den soporte a la BD.

A partir de la situación descrita anteriormente se tiene como problema a resolver:

¿Cómo contribuir a perfeccionar la capa de acceso a datos del sistema HelpDesk?

La presente investigación tiene como objeto de estudio: **Proceso de perfeccionamiento** de la capa de acceso a datos.

A partir del objeto de estudio se determinó el campo de acción **que enmarca:** Proceso de mejora de la capa de acceso a datos de la aplicación HelpDesk en el MININT.

Por lo antes planteado, se definió como objetivo de la investigación: Mejorar los módulos existentes con una herramienta ORM que contribuya a perfeccionar la capa de acceso a los datos del sistema HelpDesk, del cual se derivan los siguientes objetivos específicos:

- Investigar las tecnologías y herramientas ORM que existen.
- Elaborar el modelo de objetos.
- Implementar Módulos con la técnica ORM.
- Probar Módulos para verificar su buen funcionamiento.

Para desarrollar el trabajo de diploma, se proponen las siguientes tareas de la investigación:

- Análisis de las herramientas ORM para .NET.
- Selección de las herramientas adecuadas para el desarrollo de los módulos.
- Definición de los requisitos funcionales y no funcionales de los módulos.
- Identificación de los procesos existentes en el sistema.
- Confección del diseño del módulo con herramientas ORM para la capa de acceso a datos.
- Documentación de las bases de los módulos.
- Implementación y Prueba del módulo de acceso a datos.

Entre los métodos teóricos utilizados en esta investigación está el de modelación que permite la confección de los distintos modelos de objetos que serán el plano para la implementación del sistema. El histórico-lógico, se utiliza para el estudio de los distintos sistemas encargados de la gestión de servicios de las TIC y las herramientas para el perfeccionamiento de capas de acceso a datos, dando la posibilidad de analizar la trayectoria histórica de los mismos, así como su evolución y desarrollo.

Como métodos empíricos se utiliza la observación: utilizado como medio para la adquisición de conocimientos. En los intercambios realizados con el laboratorio 222 mediante este método aprendimos de una forma más fácil a utilizar el framework seleccionado, tomando de éstos algunas buenas prácticas para el posterior desempeño de la capa de acceso datos.

El presente trabajo se estructuró en los siguientes capítulos:

Capítulo 1: Fundamentación teórica

En este capítulo se realiza un estudio del estado del arte de las aplicaciones HelpDesk y el uso de ORM. Se ofrecen descripciones, definiciones, características de las tecnologías, técnicas y herramientas que serán utilizadas para el perfeccionamiento de la capa de acceso a datos de la aplicación. Además, se tratan todos los términos que dan lugar a nuestra investigación.

Capítulo 2: Características del módulo de acceso a datos HelpDesk

En el capítulo se identifican las características y comportamiento de las mejoras a realizar en la capa de acceso a datos del sistema HelpDesk. Se detallan los Requisitos funcionales y no funcionales. Se abordan detalles específicos de la herramienta seleccionada y se muestra como quedaría diseñada la BD luego de haberle incorporado el uso del framework seleccionado. Además, se muestran algunas estrategias y se detalla la que se siguió para el desarrollo de la capa de acceso a datos. Se abordan aspectos como la seguridad y el tratamiento de errores del sistema entre otras.

Capítulo 3: Implementación y Prueba

En este capítulo se muestra el estándar de codificación a seguir para mantener una igualdad en el desarrollo de los involucrados, facilitar la comprensión del código a desarrollar y que sea utilizado por si se necesita realizar algún cambio en el futuro. Se expone una explicación de la solución propuesta especificando en las funcionalidades más generales. Se muestran algunos diagramas como el de despliegue y el modelo de datos luego de normalizar la BD, se muestra además algunas descripciones de las tablas que presenta la BD, se mencionan las funcionalidades desarrolladas, se explica la estrategia seguida para aplicar la propuesta de solución, se detallan las especificidades de uso para las herramientas y técnicas empleadas, finalmente se tratan las pruebas a aplicar para verificar el correcto funcionamiento de la capa de acceso a datos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace referencia a los principales aspectos y definiciones que contribuirán a una mejor comprensión de los términos que se emplean a lo largo del trabajo de diploma. Se realiza una caracterización del objeto de estudio realizando un estudio del arte acerca de las tendencias actuales sobre el uso de las herramientas y tecnologías utilizadas para perfeccionar capas de acceso a datos de aplicaciones similares a nivel nacional e internacional. Además, se describen las herramientas, tecnologías y metodologías utilizadas para dar solución al problema con el fin de justificar su empleo en el mejoramiento de la capa de acceso a datos.

1.2 Conceptos asociados al dominio del problema.

Para lograr una mejor comprensión y desenvolvimiento de las áreas temáticas que se abordarán en el presente y los posteriores capítulos, se mostrarán una serie de términos identificados durante la investigación realizada que corresponden a las capas de acceso a datos.

1.2.1 Diferencia de Impedancia

Para la mayoría de las aplicaciones, almacenar y recuperar información implica alguna forma de interacción con una BD relacional. El acceso desde un lenguaje de programación orientado a objeto a una BD relacional presenta un problema fundamental para los desarrolladores porque el diseño de datos relacionales y el orientado a objetos comparten estructuras de relaciones muy diferentes dentro de sus respectivos entornos, ya que las bases de datos relacionales están estructuradas en una configuración tabular (tablas, columnas, registros) y el modelo orientado a objetos normalmente está relacionado en forma de árbol. A la disyuntiva planteada anteriormente se le denominará:”diferencia de impedancia” entre el modelo relacional y el orientado a objeto. (CEYUSA, 2006)

1.2 .2 Sistema de Bases de Datos

Los sistemas de BD constituyen una de las herramientas más difundidas en la actual sociedad de la información, son utilizadas para la recuperación y almacenamiento de la información que manejan las empresas y diversos organismos ya sean dedicado a la cultura, educación o las ciencias en general.

Surgen con el objetivo de resolver los problemas que planteaban los sistemas de ficheros, tales como la redundancia de datos y la dependencia entre programas y datos.

Una BD puede definirse como: “un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquinas accesibles en tiempo real”. (HYDE, 2002)

Para facilitar el trabajo con la BD existen los Sistemas de Gestión de Base de Datos (SGBD), estos sistemas representan: “un tipo de *software* muy específico, dedicado a servir de interfaz entre la BD, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta”. (VALDÉS, 2007)

Un SGBD es un software que permite definir, crear y mantener las BD además, proporciona un acceso controlado a estas y un lenguaje de manejo de datos para la inserción, actualización, eliminación y consulta de información en estas BD.

Los SGBD se dividen en tres generaciones. La primera generación de SGBD comprende a los sistemas jerárquicos y de red. El modelo relacional da surgimiento a la segunda generación, los sistemas de bases de datos relacionales, siendo este el más utilizado y extendido en la actualidad. La tercera generación de los SGBD se encuentra representada por el modelo relacional extendido y el modelo orientado a objetos.

Estos sistemas presentan una serie de ventajas. Algunas de estas ventajas son el control de la redundancia, la consistencia de datos, la mejora en los aspectos de seguridad y la integridad. Algunos de sus inconvenientes son su coste al requerir más capacidad de almacenamiento donde en la mayoría de los casos se dedica exclusivamente una computadora para el SGBD; al estar todo la información centralizada hace que presente vulnerabilidad ante los fallos que puedan producirse. (VALDÉS, 2007)

1.2.3 Sistema de Base de Datos Relacional

Hasta el momento se han mencionado las diferentes generaciones o tipos de sistema de gestión de BD, cada una de estas generaciones trata de resolver los problemas que planteaban los sistemas de BD predecesores. El SGBD más extendido y de mayor uso en la actualidad es el Sistema de Gestión de Base de Datos Relacional (SGBDR), como su nombre lo indica este sistema se encarga de administrar las BD relacionales. Aquellas BD que almacenan la información en forma de tablas, estas tablas se organizan en filas y columnas; cada fila representa un registro (conjuntos de datos acerca de elementos separados) y las columnas definen los campos del registro (atributos particulares de un registro).

Los SGBDR permiten realizar búsquedas utilizando los datos de columnas especificadas de una tabla para encontrar datos adicionales en otra tabla, también proveen herramientas para evitar la duplicidad de registros, garantizan que al eliminar un registro sean eliminados todos los registros relacionados dependientes y además representan un modelo más comprensible y aplicable. Los SGBDR presentan deficiencia en cuanto al almacenamiento de datos gráficos, y puede dificultar el desarrollo de ciertas aplicaciones que manejan datos de tipos multimedia como audio y video.

Los Sistemas de Gestión de Base de Datos Orientadas a Objetos (SGBDOO) se propusieron con el objetivo de satisfacer las necesidades de las aplicaciones anteriormente mencionadas. Sin embargo, este sistema de gestión de BD no logra sustituir a los SGBDR, ya que la composición de este modelo, del orientado a objeto, no está del todo claro ya que aún no existe algún modelo de datos para los SGBDOO que esté aceptado mundialmente, los modelos de datos orientado a objetos no están totalmente desarrollados ya que carecen de una teórica matemática coherente que le sirva de base, algo que no le sucede a los SGBDR. También los SGBDOO no disponen un alto nivel de desarrollo comercial y disponen de muy poca experiencia en su uso. Por tanto, actualmente los SGBDR son los más conocidos, extendidos y los que presentan mayor desarrollo comercial. (Gregorio, 2001)

1.2.4 El Modelo Relacional

El modelo relacional para la gestión de una BD es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. (Codd, 1970).

Consiste en un conjunto de objetos o relaciones, conjunto de operadores que actúan sobre los objetos y reglas para asegurar la integridad y la consistencia del propio modelo. Una relación consta de: un esquema y este a su vez de un conjunto de pares (atributo, dominio).

Este modelo permite representar la información del mundo real de una manera intuitiva, introduciendo conceptos cotidianos y fáciles de entender por cualquier inexperto. Asimismo, mantiene información sobre las propias características de la BD (metadatos), que facilitan las modificaciones, disminuyendo los problemas ocasionados en las aplicaciones ya desarrolladas. Por otro lado, incorpora mecanismos de consulta muy potente, totalmente independiente del SGBD e incluso de la organización física de los datos del propio SGBD.

1.2.5 Normalización de una Base de Datos

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de BD a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la BD, era ineficiente y conducía a errores de lógica cuando se trataban de manipular los datos.

Los seres humanos tienen la tendencia de simplificar las cosas al máximo. Lo hacen con casi todo, desde los animales hasta con los automóviles. Ven una imagen de gran tamaño y la hacen más simple agrupando cosas similares juntas. Las guías que la normalización provee, crean el marco de referencia para simplificar una estructura de datos compleja. Otra ventaja de la normalización de BD es el consumo de espacio. Una BD normalizada ocupa menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco. (EDUARDO, 2003)

Consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad relacional al relacional. Las BD relacionales se normalizan para:

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

Las formas normales son aplicadas a las tablas de una BD. Decir que una BD está en la forma normal **N** es decir que todas sus tablas están en la forma normal **N**. En general, las primeras tres formas normales son suficientes para cubrir las necesidades de la mayoría de las BD. El creador de estas 3 primeras formas normales (o reglas) fue Edgar F. Codd. Las formas normales establecen los siguientes planteamientos:

Primera Forma Normal (1FN)

Una tabla está en Primera Forma Normal si:

Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos. La tabla contiene una clave primaria, la clave primaria no contiene atributos nulos, no posee ciclos repetitivos, no debe de existir variación en el número de columnas, una columna no puede tener múltiples valores. Los datos son atómicos (Si a cada valor de X le pertenece un valor de Y, entonces a cada valor de Y le pertenece un valor de X). Esta forma normal elimina los valores repetidos dentro de una BD.

Segunda Forma Normal (2FN)

Independencia Funcional. Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales.

Tercera Forma Normal (3FN)

La tabla se encuentra en 3FN si es 2FN y si no existe ninguna dependencia funcional transitiva entre los atributos que no son clave.

1.2.6 Transacción

Una transacción: es una unidad atómica de ejecución, es decir, un grupo de instrucciones que se ejecutan con éxito en su totalidad, o no se ejecuta ninguna. La transacción una vez iniciada, puede terminar de dos formas: transacción aceptada, cuando las operaciones asociadas a una transacción se ejecutan con éxito y se hacen persistentes los cambios realizados, en la BD o transacción rechazada, cuando ocurre una intervención en la ejecución de las operaciones asociadas a una transacción y se "vuelve atrás" todos los cambios que puedan haberse realizado, imposibilitando de esta manera la inconsistencia en los datos de la BD. (MSDN, 2005)

En un SGBD, es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica. Cada operación de acceso a datos, como crear, actualizar o borrar datos está asociada a una transacción de BD.

1.2.7 Capa de Acceso a datos

Una capa de acceso a datos (DAL) es una capa de un programa de ordenador que proporciona simplificado el acceso a los datos almacenados en el almacenamiento persistente de algún tipo, como una

entidad relacional de bases de datos. Además, es esta capa la que vincula directamente a la aplicación con la BD y es la base de todas las demás capas superiores. (Luque, 2005)

1.2.8 Motores de persistencia

La filosofía de imponer un único modelo teórico (un único formato de datos) a toda la aplicación sufre un gran número de inconvenientes. En el caso de que toda la aplicación siga el modelo relacional, se perderían las ventajas de la orientación a objetos. En el caso de que toda la aplicación siga el modelo orientado a objetos, implica que las BD sean inmaduras y tengan un bajo nivel de estandarización. Por otro lado, si la aplicación sigue la lógica orientada a objetos y la BD es relacional, lo que, en principio, constituye la opción más natural, plantea el problema de cómo conseguir que dos componentes con formatos de datos muy diferentes puedan comunicarse y trabajar conjuntamente. La solución se basa en encontrar un elemento intermedio que permita comunicar aplicación y BD, a este elemento se le conoce como “capa de persistencia” o “motor de persistencia”.

El motor de persistencia traduce entre los dos formatos de datos: de registros a objetos y de objetos a registros. La situación se ejemplifica en la Figura #1. Cuando el programa quiere grabar un objeto llama al motor de persistencia, que traduce el objeto a registros y llama a la BD para que guarde estos registros. De la misma manera, cuando el programa quiere recuperar un objeto, la BD recupera los registros correspondientes, los cuales son traducidos en formato de objeto por el motor de persistencia.

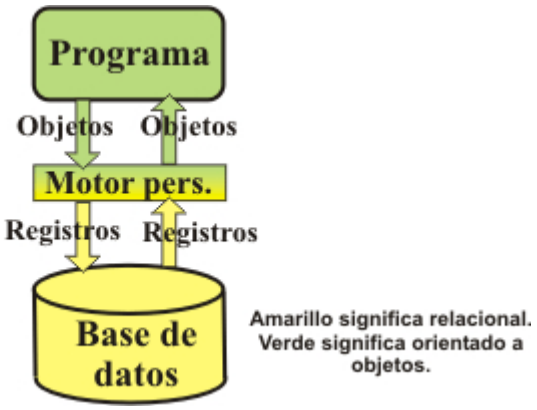


Figura 1: Motores de persistencia

El programa sólo ve que puede guardar objetos y recuperar objetos, como si estuviera programado para una BD orientada a objetos. La BD sólo ve que guarda registros y recupera registros, como si el programa estuviera dirigiéndose a ella de forma relacional. Es decir, cada uno de los dos componentes trabaja con el formato de datos que le resulta más natural y es el motor de persistencia el que actúa de traductor entre los dos modelos, permitiendo que los dos componentes se comuniquen y trabajen conjuntamente.

1.3 Detalles asociados con la aplicación a perfeccionar

A continuación se exponen las características de la aplicación a la cual se le perfeccionará la capa de acceso a datos. Detalles y conceptos que ayudan a comprender las necesidades de la misma.

1.3.1 ¿Qué es un Sistema HelpDesk?

HelpDesk es una aplicación web que maneja y coordina las incidencias de la manera más rápida, ayudando a incrementar la productividad y aumentando la satisfacción de los usuarios internos y externos. Se encarga solamente de gestionar y solucionar todas las posibles incidencias como una interrupción o una solicitud relacionadas con las TIC.

1.3.2 Interrupción

Una interrupción es el evento que afecta el buen funcionamiento de los servicios de las TIC. La misma puede ocurrir mediante la caída total o parcial de una red o por un desperfecto en un sistema operativo o teléfono, y puede ser provocada por muchos factores asociados principalmente con la protección y las condiciones de explotación, así como la atención y seguimiento que se le dé al equipamiento utilizado para un mejor manejo de la información.

1.3.3 Servicio.

Un servicio es el resultado de llevar a cabo necesariamente al menos una actividad en la interfaz entre el proveedor y el cliente la cual generalmente es intangible. La prestación de un servicio puede implicar, por ejemplo:

- Una actividad realizada sobre un producto tangible suministrado por el cliente.
- Una actividad realizada sobre un producto intangible suministrado por el cliente.
- La entrega de un producto intangible.
- La creación de una ambientación para el cliente.

Se puede afirmar que la característica básica de los servicios que los diferencia de los productos es que son intangibles, es decir, no son palpables, ni perceptivos, ni materiales. Los servicios que brindan las TIC no difieren de ninguna manera de lo enunciado anteriormente porque son soluciones tecnológicas que se les ofrecen a determinadas organizaciones, empresas o cualquier cliente que lo solicite, en el ámbito de la informática y las comunicaciones.

1.3.4 Aplicación a perfeccionar capa de acceso.

Se presenta una aplicación Web denominada HelpDesk la cual permitirá la automatización de los procesos de gestión y control de los servicios que brindan las TIC en el MININT, permitiendo que se cumplan los requerimientos planteados por los clientes. El sistema será capaz de realizar de manera eficiente y amigable todos los procesos relacionados con la gestión de servicios que se llevan a cabo actualmente de forma manual, contribuyendo a mejorar y agilizar la calidad de estos procesos.

La aplicación brindará la posibilidad de almacenar todas las solicitudes de servicios realizadas por los usuarios y a la vez ellos podrán obtener información sobre el estado de sus solicitudes. También permitirá llevar un control riguroso de todas las solicitudes de servicios por parte de todo el interesado a través de informes estadísticos de los reportes realizados por los usuarios. Esta aplicación es configurable y personalizada y puede ser utilizada en cualquier organización o empresa. (Losada, 2009)

1.3.5 Herramientas del sistema HelpDesk

- Utiliza Plataforma.NET.
- Lenguaje de programación: C#.NET.
- IDE Visual Studio 2008.
- Gestor de Base de Datos: Oracle.

1.3.6 Deficiencias en el Sistema HelpDesk

HelpDesk se diseñó en tres capas con el estilo arquitectónico de llamada-retorno, específicamente el patrón Modelo Vista Controlador (MVC). El sistema presenta algunos problemas con la capa de acceso a datos como:

- La BD no está normalizada.
- Algunas conexiones se quedan abiertas.
- No existe un control adecuado de las transacciones que se realizan.
- Muchas de las herramientas empleadas no son las más actualizadas.
- Dificultad para que los desarrolladores den soporte a la BD.

1.4 Estado del Arte en el mundo

Existen diversas empresas, organizaciones y compañías que utilizan esta nueva tecnología Web que ha surgido para darle solución a los diversos problemas en cuanto al soporte de servicios.

En la investigación realizada se encontró información relacionada con la empresa mayorista de software Ireo la cual es especialista en herramientas de soporte técnico y administración. Las soluciones Service Desk propuestas por Ireo permiten gestionar la mayoría de los servicios que presta un departamento de informática. Con una base instalada de más de 200 clientes en España y Portugal, Ireo cuenta con una larga experiencia en el mercado de Service Desk y un sólido equipo de consultores especializados.

Dentro de sus productos Service Desk, se encuentran BMC Service Desk Express, BMC Service Desk Express SBE, Service Desk Plus y Support Center Plus.

1.4.1 BMC Service Desk Express

BMC Service Desk Express fue diseñado para grandes y medianas empresas. Se caracteriza por su amplia funcionalidad y fácil personalización. Se dice que es uno de los líderes del mercado a nivel mundial y cuenta con un amplio número de clientes en todo el mundo.

1.4.2 BMC Service Desk Express SBE

BMC Service Desk Express SBE es una nueva versión del BMC Service Desk Express. SBE significa "Small Business Edition" ("Pyme"). Esta versión ofrece toda la eficacia de la versión padre, pero a

diferencia de esta utiliza solamente los módulos típicos para el desarrollo de pequeñas y medinas empresas. Es más rápido a la hora de su implementación y de menor coste que la versión a la que sucede.

1.4.3 Service Desk Plus

Service Desk Plus es una aplicación cien por ciento Web, que además de ofrecer una amplia funcionalidad a bajo coste, incluye un módulo de inventario automático. Según Ireo SPD (Service Desk Plus) es idóneo para empresas que buscan un sencillo sistema Service Desk para uso interno.

1.4.4 Support Center Plus

Support Center Plus es un sistema HelpDesk y de atención a clientes, diseñado para organizaciones que dan soporte a clientes externos. Incluye la gestión de clientes, un catálogo de productos y servicios, gestión de contratos con clientes y mucho más.

Estos Service Desk fueron concebidos e implementados para gestionar los diferentes procesos que se desarrollan en un departamento de Tecnologías de la información (TI), tal como se plantea en los estándares de Infraestructura de Tecnologías de Información (ITIL) es un conjunto de normas creadas para ayudar a mejorar la calidad y eficiencia de los procesos del negocio en cualquier tipo de organización. Todos son aplicaciones Web, con un alto nivel de personalización desde cualquiera de sus ambientes de trabajo, permitiendo que el sistema sea implementado en dependencia de las necesidades de la empresa y a su vez cada usuario pueda configurarlo para su puesto de trabajo. Estos software(s) se licencian y para ello existen dos alternativas:

- La suscripción anual es la más económica. Se puede adquirir una suscripción de 1, 2 ó 3 años, lo que implica un considerable ahorro fiscal ya que la licencia es un gasto, no una inversión.
- Para clientes que quieren adquirir el producto, existen licencias perpetuas.

1.4.5 Tecnología Service Desk Libre

OneOrZero TMS (Task Management System) es un sistema de ticket por vía Web, basada en la tecnología Help Desk, que proporciona características ideales para el manejo de las tareas del negocio. Es robusto, rápido, altamente personalizable y compatible con cualquier plataforma virtual. Se rige por la Licencia Pública General (GPL, por sus siglas en inglés) de código abierto (por su nombre en inglés, Open

Source) permitiendo a cada empresa personalizar el producto libremente. Los desarrolladores de este sistema ofertan servicios técnicos y personalizados a los clientes basados en la experiencia y muchas habilidades con que cuentan en el trabajo en equipo.

OneOrZero a pesar de ser totalmente libre, las últimas versiones y mejoras del producto son reservadas para la asociación de miembros de desarrollo de este Service Desk. Para hacerse miembro se necesita abonar cierta cantidad de dinero que es usado para costear el desarrollo de OneOrZero y en dependencia de esta cantidad son los privilegios otorgados. Además aunque está basado en la tecnología Help Desk, no tiene una interfaz Web específica para gestionar reportes mediante llamadas telefónicas.

Mundialmente ya es un hecho irrefutable el que las compañías a la hora de desarrollar aplicaciones con grandes volúmenes de datos hagan uso de las herramientas ORM, estas son una de las armas más poderosas que emplean los programadores para ahorrar tiempo y ganar en eficacia, dado a que facilitan un código limpio y estricto disminuyendo así la probabilidad de errores. Un ejemplo de aplicación donde se evidencia el uso de estas herramientas es el TEGZ - Sistema para el control de Trabajos Especiales de Grado del IUETLV.

1.4.6 Estado del Arte en Cuba

En Cuba también se hace uso de los Call Center siendo ejemplo de ello ETECSA que es una organización cubana de capital mixto que tiene como objeto social la prestación de servicios públicos de telecomunicaciones, mediante la proyección, explotación, operación, instalación, comercialización y mantenimiento de redes públicas de telecomunicaciones en todo el territorio cubano. Se ha caracterizado por llevar la prestación de modernos servicios de beneficio popular, tanto a las ciudades como a las zonas rurales de difícil acceso, respetando la naturaleza y la ecología en sus procesos inversionistas.

La empresa de Telecomunicaciones de Cuba brinda diversos servicios y cuenta con un personal de ejecutivos calificados y la más avanzada tecnología. Brinda una excelente calidad en la comunicación y en los servicios de asistencia ofrecido por operadoras de esta empresa.

Como todo Service Desk, los Call Center de ETECSA son evaluados por el grado de satisfacción de usuarios y clientes con el servicio prestado.

En Cuba existen ejemplos de aplicaciones que son de gran ayuda para las organizaciones a las que pertenecen y en las que se utilizan herramientas ORM para la implementación de la capa de acceso a datos. En la empresa Aerovaradero S.A se utiliza un sistema denominado Sistema para la reserva de

guías aéreas el cual se encarga de gestionar las reservaciones de guías aéreas para la exportación de carga, el mismo presenta una estrecha relación con el sistema SICSA (Sistema Integral de Servicios de Aerovaradero) el cual controla toda la actividad de manejo de carga en Aerovaradero S.A y que se encuentra en explotación desde el año 2002. Este sistema esta implementado sobre plataforma .NET, utilizando arquitectura tres capas y ORM para gestionar la BD.

En la UCI existen algunos proyectos que han empleado esta forma de trabajo entre ellos se encuentran: Sinapsis de la facultad#3 el cual esta implementado en java y como ORM emplea Spring e Hibernate en este caso también se encuentra el proyecto de Mapeo cerebral Humano(MCHC) de la facultad #6 entre otros.

1.5 Mecanismos, técnicas y patrones de acceso a datos

El nivel de evolución de los mecanismos de acceso a datos es notable en la actualidad. Antes del surgimiento de mecanismos para acceder a BD, el acceso a un servidor, requería el conocimiento puntual de librerías específicas de cada servidor, lo que originaba que una aplicación desarrollada con estas librerías, fuera inútil si se cambiaban los datos del servidor. Dado este problema surgen mecanismos para acceder a la BD; a continuación se muestran los mecanismo encontrados durante la investigación

1.5.1 ODBC. Conectividad abierta a base de datos

ODBC es la abreviatura en inglés que indica la conectividad abierta de BD. Es un estándar desarrollado por Microsoft que se utiliza para acceder a una BD a través de consultas SQL, posibilita el acceso a cualquier dato desde cualquier aplicación, sin importar qué sistema de BD almacena la información. ODBC ofrece gran independencia entre las aplicaciones y las BD, ya que inserta un manejador de estas que traduce las consultas de datos enviadas por la aplicación en comandos que el sistema entienda. Para utilizar este mecanismo la aplicación y el sistema deben ser compatibles con ODBC, es decir, que la aplicación sea capaz de producir comandos ODBC y que el sistema de BD sea capaz de responder ante ellos, esto se cumple en su totalidad ya que actualmente la mayoría de los sistemas de BD soportan comandos ODBC.

Este es un mecanismo de gran utilidad ya que ofrece una interfaz de acceso universal que permite no tener que aprender a usar múltiples interfaces de programación de aplicación. Sin embargo, ODBC tiene sus inconvenientes producto de que es una interfaz escrita en el lenguaje C y al no ser este un lenguaje

portable le resta escalabilidad a la aplicación desarrollada con ODBC, haciéndola dependiente del sistema operativo. Y además, ODBC tiene el inconveniente de que se debe instalar manualmente en cada máquina donde se ejecute la aplicación que utilice comandos ODBC para acceder a la BD. (SKINNER, 2002)

1.5.2 ADO.Net

ADO.Net es un conjunto de componentes que brindan servicios de acceso a datos. Todos estos componentes forman parte de la biblioteca de clases básicas incluidas en el Framework .NET. Generalmente es utilizado para acceder y modificar la información guardada en un SGBDR, aunque también puede ser usado para acceder a fuentes de datos no relacionales, tales como XML, Excel y otros.

ADO.Net esta divido en dos partes primarias: Un proveedor de datos (DataProvider), que es un conjunto de clases que proporcionan acceso a una fuente de datos específicos. Mediante un proveedor se pueden realizar conexiones con la fuente de datos, enviar consultas SQL y almacenar datos en un objeto Data Set. Cada proveedor presenta cuatros objetos básicos que son:

- Conexión (Connection), para conectar y administrar las transacciones en una BD.
- Comandos (Command), para emitir sentencias SQL a una BD.
- Adaptador de datos(DataAdapter), para insertar datos en un objeto DataSet y reconciliar datos de la BD
- Lector de datos (DataReader), proporciona una forma de leer una secuencia de registros de datos sólo hacia delante desde un origen de datos. (SKINNER, 2002)
- Un DataSet, que representa en memoria una estructura análoga a una BD relacional, permite almacenar un conjunto de datos obtenidos mediante un objeto DataAdapter. Un DataSet es un objeto independiente que contiene un conjunto de registros de forma desconectada a la fuente de datos por lo que el DataSet no conoce en absoluto el origen y destino de los datos que contiene. En él no solo se pueden colocar datos provenientes de una BD relacional, permite manejar datos provenientes de un XML, código o información escrita explícitamente por un usuario. (SKINNER, 2002)

Una característica particular de ADO.Net es que se puede desarrollar el acceso a BD relacionales manteniendo una conexión física permanente con la BD durante todo el proceso de consultas o actualizaciones sobre los datos, pero también permite desarrollar el acceso a BD relacionales de manera desconectada debido a la existencia del DataSet, ya que en este objeto se copian los datos obtenidos de la colección de datos y luego se pueden consultar y actualizar sin contar con una conexión abierta. Luego si se desea se puede establecer una conexión con la misma, mediante el objeto DataAdapter, para sincronizar los cambios efectuados por el DataSet y actualizar los datos en la BD. Es válido mencionar que no es obligatorio usar el objeto DataSet para insertar, actualizar o eliminar datos en una BD relacional, se puede ejecutar comandos SQL directamente en la BD para realizar inserciones, actualizaciones y eliminaciones.

1.5.3 Patrón de diseño DAO

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Los patrones de diseño pretenden evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, proporcionando un catálogo de elementos reusables en el diseño de sistemas de software. Formaliza un lenguaje común entre diseñadores, estandarizando también el modo en que se realiza el diseño.

En la tarea de desarrollar el acceso a datos de un sistema informático existe un patrón de diseño que su utilización se ha hecho indispensable, este es el patrón de diseño DAO (Data Access Object, por sus siglas en inglés) es uno de los patrones de diseño de mayor uso en el desarrollo del acceso a datos de un sistema informático.

Se utiliza para separar las operaciones de bajo nivel de acceso a datos, de las operaciones de alto nivel de la lógica del negocio. Una implementación típica de DAO contiene:

- Una interfaz DAO. Que expone operaciones de acceso a datos, estas operaciones no están vinculadas con alguna tecnología de persistencia. Esta característica permite la abstracción del mecanismo que se emplee para acceder a los datos, permitiendo desarrollar diversas implementaciones a una misma interfaz DAO.

- Una clase concreta que implemente la interfaz DAO. Esta clase contiene la lógica de acceso a datos a una fuente específica de información.
- Una clase fabricadora. Responsable de instanciar las interfaces DAO.
- Objetos de transferencias de datos. Posibilitan transferir la información que se obtiene desde una fuente determinada.

Este patrón permite implementar las operaciones definidas por una interfaz DAO de diversas maneras utilizando diferentes tecnologías, sin producir cambios en el código que utiliza las operaciones expuestas por esta interfaz, ya que cada implementación concreta de una interfaz DAO es transparente a quien utilice dicha interfaz. (LAGO, 2007)

1.5.4 Patrón de Acceso a Datos CRUD

Acrónimo de Create-Read-Update-Delete. Conocido como el padre de todos los patrones de capa de acceso. Es usado para referirse a las funciones básicas en BD y a la capa de persistencia en un sistema de software. Describe que cada objeto debe ser creado en la BD para que sea persistente. Una vez creado, la capa de acceso debe tener una forma de leerlo para poder actualizarlo o simplemente borrarlo. Teóricamente el borrado de objetos debería quedar a cargo de la misma BD. Pero un recolector de objetos “basura” (garbage collector) en una BD gigante afecta en gran medida el rendimiento. Por ello es que la tarea de borrado queda delegada al programador. (PIZARRO, 2005)

1.6 Técnica Mapeo Objeto-Relacional

El mapeo objeto-relacional, conocido por su nombre en inglés: Object-Relational- Mapping surgió a mediado de la década del 90, su uso se hizo masivo a partir de la masificación de Java. Por esa razón, los framework más populares hoy en día en .Net son adaptaciones del modelo pensado para Java. (PIZARRO, 2005) Consiste en una técnica de acceso a datos que trata de disminuir la diferencia de impedancia entre el modelo relacional y el modelo orientado a objetos, transformando las representaciones de los datos de un modelo de objetos en un modelo de datos con un esquema relacional y viceversa, donde esta transformación se realiza transparentemente al programador. Una solución de este tipo está compuesta de cuatro partes:

- Una API encargada de realizar las operaciones de creación, recuperación, actualización y eliminación (operaciones CRUD) para los objetos de las clases que se corresponden con las tablas de la BD. Dichas operaciones se corresponden con instrucciones de lenguajes de manipulación de datos en el SGBD. (QUINTERO, 2008)
- Un lenguaje de consulta a la BD, generalmente orientado a objetos, o API para especificar consultas que se refieran a las clases y atributos de estas clases. Estos lenguajes en su mayoría son similares al SQL, con la diferencia principalmente de que es orientado a objeto como se ha comentado anteriormente. (QUINTERO, 2008)
- Una utilidad para especificar metadatos de mapeo. Generalmente mediantes archivos XML se realizan la correspondencia entre las tablas de la BD y las clases que representan estas tablas. También se hacen corresponder los atributos de las clases con los campos de la tabla. (QUINTERO, 2008)
- Técnicas de optimización de transacciones como el chequeo sucio automático, que permite detectar cambios en los objetos y actualizar su estado en la BD sin necesidad de escribir alguna sentencia de actualización; técnicas estratégicas para la recuperación de las colecciones que presentan los objetos (producto de las asociaciones uno-muchos, muchos-uno), tales como:
  - Carga perezosa (Lazy Loading), donde se cargan primero el objeto de la tabla maestra y posteriormente, si se necesitan, sus colecciones de objeto que representan los datos de la tabla detalle.
  - Carga proactiva (Eager Loading). Se cargan de una vez el objeto de la tabla maestra y sus colecciones de objetos de la tabla detalle. (QUINTERO, 2008)
  - Con el uso de la técnica ORM el programador de acceso a datos no tiene que convertir explícitamente los datos resultantes de una consulta a la BD en objetos de transferencias. El acceso a datos mediante el uso de esta técnica se realiza completamente orientado a objetos, en cierta medida se ignora la existencia del SGBDR.

1.6.1 Herramientas que utilizan el mapeo objeto-relacional

Anteriormente se describió una técnica que permite eliminar la diferencia de impedancia entre el modelo relacional, que es el modelo propio de los SGBDR, y el modelo orientado a objeto. En la práctica esta técnica permite crear una BD orientada a objetos virtual sobre la BD relacional. Esto posibilita el uso de las

características propias de la programación orientada a objetos, básicamente herencia y polimorfismo. Este método de solución es preferido por los arquitectos de software, ya que permite diseñar el modelo de objetos independientemente del sistema que se utilice para guardar estos objetos. Estas herramientas facilitan al programador:

- Reducción de hasta un 25% del tiempo de desarrollo dedicado al mapeo.
- Evita el SQL HARD-CODED, especificando en archivos de configuración las tablas, propiedades, etc. Sin preocuparse por el renombre de tablas, columnas etc.
- Correspondencia lógica y natural del modelo de objetos. (PIZARRO, 2005)

En la actualidad hay herramientas disponibles de uso libre que implementan ORM tanto para java como para .Net. Entre las herramientas que existen para Net se encuentran:

- Opf3
- Castle Project
- TierDeveloper
- NHibernate
- IBatis
- Entity Framework

1.6.2 NHibernate

NHibernate es uno de los framework más usados. Se debe principalmente al poderoso lenguaje de consulta que trae consigo: HQL. Hibernate Query Language. Está basado en el proyecto Hibernate de Java, al igual que OJB.NET es un proyecto open Source. (PIZARRO, 2005)

NHibernate también funciona en Mono que no es más que un proyecto de código abierto iniciado por la empresa proveedora de software libre Ximian. (Dario, 2007)

Al usar NHibernate para el acceso a datos el desarrollador se asegura de que su aplicación es agnóstica en cuanto al motor de BD a utilizar en producción, pues NHibernate soporta los más habituales en el mercado: MySQL, PostgreSQL, Oracle, MS SQL Server entre otros. Sólo se necesita cambiar una línea en el fichero de configuración para que se pueda utilizar una BD distinta. NHibernate es software libre, distribuido bajo los términos de la LGPL (licencia Pública General Menor de GNU). Es el framework de persistencia más completo que existe en el mercado. Soporta todos los tipos de relaciones (Uno a uno,

uno a muchos, muchos a muchos) y puede mapear varias tablas a una clase o al revés, también permite usar herencia, transacciones, agregaciones y muchas funcionalidades más, prácticamente se puede hacer cualquier cosa que se le ocurra al programador (WIKI, 2010) Algunas de las características más importantes:

- Permite el mapeo de relaciones a tipos .NET específicos.
- Permite una buena interfaz para estructurar la consulta a partir de criterios.
- Las transacciones son administradas por objetos Session.
- La configuración del mapeo y demás se administra tanto por código como por fuera de él.
- Accede a propiedades privadas, públicas o protegidas.
- No soporta aún la carga retardada en asociaciones 1 a 1.
- Las asociaciones son mapeadas a objetos IList, Collection o Dictionary.
- Asocia las clases con sus correspondientes proxys.
- Soporta bases en SQLServer, Oracle y parcialmente OleDB.
- Soporta actualizaciones en cascada, con posibilidad de especificar que tipo de actualización está permitido: insert, update y delete.
- La configuración de NHibernate es fuera de banda, a través de archivos en xml. (PIZARRO, 2005)

1.6.3 TierDeveloper

Es el estándar de la industria para el mapeo de Objetos a Relaciones y una herramienta de generación de código que le ayuda a diseñar con rapidez, generar e implementar objetos de nivel medio para aplicaciones. Permite el desarrollo de aplicaciones orientadas a objeto complejo, ahorra el 50% de tiempo de desarrollo. Genera código sin comprometer la calidad. TierDeveloper (TD) es un objeto a la cartografía relacional (O / R Mapping) herramienta de generación de código que te permite generar mapas bien diseñados. Neta de empresas y datos de objetos. TD también genera personalizado de ASP.NET y aplicaciones de Windows Forms.

Esta considerada la herramienta principal para .NET de generación de código y como técnica ORM, ha sido recientemente declarado totalmente software libre, TD puede generar código en VB.NET y C # para ASP.NET, windows forms, remoting y servicios Web. Es el único generador que te permite trabajar con sus plantillas de generación de código de gran alcance en la manera más sencilla posible. Permite

generar código para SQL Server, MS Access, DB2, Oracle y BD, OLEDB compatible. Permite construir MARCOS DE NEGOCIO con un aumento instantáneo de la productividad y al mismo tiempo simplificando enormemente la gestión del cambio. (Alachisoft, 2010)

1.6.4 Entity Framework

Entity Framework (EF) es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Los arquitectos y programadores de aplicaciones orientadas a datos se han enfrentado a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar los datos. Los datos pueden abarcar varios sistemas de almacenamiento, cada uno con sus propios protocolos; incluso las aplicaciones que funcionan con un único sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con respecto a los requisitos de escribir un código de aplicación eficaz y fácil de mantener.

EF permite a los programadores trabajar con datos en forma de objetos y propiedades específicos del dominio, por ejemplo, con clientes y direcciones, sin tener que pensar en las tablas de las BD subyacentes y en las columnas en las que se almacenan estos datos. Para ello, se eleva el nivel de abstracción en la que los programadores pueden trabajar al tratar con datos y se reduce el código requerido para crear y mantener las aplicaciones orientadas a datos. Dado que EF es un componente de .NET Framework, las aplicaciones de EF se pueden ejecutar en cualquier equipo en que esté instalado .NET Framework 3.5 Service Pack 1 (SP1). (DOTNET_CLUB, 2010)

Una aplicación de EF requiere crear un modelo conceptual que defina las entidades y las relaciones, un modelo lógico que represente el modelo relacional subyacente y las asignaciones entre los dos. Las características y componentes siguientes de EF trabajan conjuntamente para proporcionar un entorno de programación de un extremo a otro.

- El Entity Data Model (EDM) es la pieza central de EF. Específica el esquema de diseño, que se usa para generar las clases programables que usa el código de la aplicación. Las estructuras de

almacenamiento de los datos conservados se representan en un esquema de almacenamiento y una especificación de asignación conecta el esquema de diseño con el esquema de almacenamiento. Las entidades conceptuales se pueden materializar como objetos o se pueden leer en un formato serializado mediante un lector de datos. Los desarrolladores pueden extender estos objetos cuando sea necesario para la compatibilidad con diferentes necesidades de la aplicación.

- El componente Object Services permite a los programadores trabajar con las clases de Common Language Runtime (CLR) generadas a partir del modelo conceptual. También proporcionan compatibilidad de infraestructura con EF, con servicios como administración de estados, seguimiento de cambios, resolución de identidad, relaciones de carga y navegación, propagación de cambios de objeto a modificaciones de BD y compatibilidad con consultas para Entity SQL.
- LINQ to Entities proporciona compatibilidad con Language-Integrated Query (LINQ) para consultar las entidades. LINQ to Entities permite a los programadores escribir consultas con la BD utilizando uno de los lenguajes de programación de .NET Framework admitidos, como Visual Basic o Visual C#. Entity SQL es un lenguaje independiente del almacenamiento que es similar a SQL y que se ha diseñado para la consulta y manipulación de gráficos enriquecidos de objetos basados en el modelo Entity Data Model (EDM).
- El proveedor EntityClient extiende el modelo de proveedor de ADO.NET teniendo acceso a los datos en lo que respecta a las entidades conceptuales y relaciones. Ejecuta consultas que usan Entity SQL. Entity SQL proporciona el lenguaje de consulta subyacente que permite a EntityClient comunicarse con la BD.
- El componente de metadatos de ADO.NET administra los metadatos en cuanto a las necesidades de tiempo de ejecución y tiempo de diseño de EF. Todos los metadatos asociados a los modelos y asignaciones se exponen a través de las interfaces de metadatos que son independientes de los mecanismos usados para el almacenamiento de los metadatos. El mecanismo de almacenamiento actual utiliza el archivo que se basa en tres dialectos XML: el lenguaje de definición de esquemas conceptuales (CSDL), el lenguaje de definición de esquemas de almacenamiento (SSDL) y el lenguaje de especificación de asignaciones (MSL).
- EF incluye un conjunto de herramientas en evolución que generan asignaciones y clases parciales que representan las entidades en el modelo conceptual.

- EF incluye un proveedor de datos SqlClient actualizado que admite los árboles de comandos canónicos. (lfranco, 2010)

1.6.5 Ventajas del Entity Framework

EF está diseñado para permitir a los programadores crear aplicaciones de acceso a datos programando con un modelo de la aplicación conceptual en lugar de programar directamente con un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientadas a datos. Las aplicaciones de EF ofrecen las siguientes ventajas:

- Las aplicaciones pueden funcionar en términos de un modelo conceptual más centrado en la aplicación, que incluye tipos con herencia, miembros complejos y relaciones.
- Las aplicaciones están libres de dependencias de codificación rígida de un motor de datos o de un esquema de almacenamiento.
- Las asignaciones entre el modelo conceptual y el esquema específico de almacenamiento pueden cambiar sin tener que cambiar el código de la aplicación.
- Los programadores pueden trabajar con un modelo de objeto de aplicación coherente que se puede asignar a diversos esquemas de almacenamiento, posiblemente implementados en sistemas de administración de BD diferentes.
- Se pueden asignar varios modelos conceptuales a un único esquema de almacenamiento.
- La compatibilidad con Language-Integrated Query proporciona validación de la sintaxis en el momento de la compilación para consultas en un modelo conceptual. (MSDN, 2010)

Compatibilidad:

- Oracle: 9.2.0.4 o superior.
- MySQL: 5.0 o superior.
- PostgreSQL: 8.0 o superior.
- SQLite: versión (5).

1.6.6 Entity Data Model (EDM) y Procedimientos Almacenados

EF admite un modelo EDM para definir datos tanto en el nivel de almacenamiento como en el nivel conceptual y una asignación entre los dos. También permite programar directamente con los tipos de datos definidos en el nivel conceptual como objetos de Common Language Runtime (CLR). EF proporciona herramientas para generar un modelo EDM y los objetos de CLR relacionados basándose en una BD existente. Esto reduce en gran medida el código de acceso a datos que se solía necesitar para crear servicios y aplicaciones de datos basadas en objetos, y agiliza la creación de servicios y aplicaciones de datos orientadas a objetos a partir de una BD existente.

El EDM admite procedimientos almacenados para la recuperación y modificación de datos. Los procedimientos almacenados se pueden usar para recuperar, actualizar y eliminar los datos.

Muchas aplicaciones de BD se basan en los procedimientos almacenados para proporcionar las ventajas siguientes:

- Seguridad. A los usuarios de la BD se les puede denegar el acceso directo a las tablas u otros objetos de la BD. Los administradores de BD sólo necesitan conceder permisos de ejecución en los procedimientos almacenados para crear puntos de entrada únicos de acceso a los datos. Esto reduce significativamente la superficie de ataque para los ataques de inyección SQL. Los parámetros de entrada y salida configurados para utilizar los valores predeterminados habilitan la validación estricta de parámetros. El código de la validación en los procedimientos almacenados puede limitar las acciones disponibles.

- Encapsulación. La lógica de datos complejos, las transacciones explícitas y otras operaciones de BD se pueden escribir una vez en el código de los procedimientos almacenados y ejecutar desde varias aplicaciones cliente. Los errores, excepciones e infracciones de la simultaneidad se pueden controlar desde el código del servidor, reduciendo el número de viajes de ida y vuelta desde las aplicaciones cliente. El código de los procedimientos almacenados se puede modificar sin afectar a las aplicaciones cliente siempre que la firma del procedimiento almacenado permanezca inalterada.
- Previsibilidad. Los administradores de BD deben asegurarse con frecuencia de que las aplicaciones cliente no emiten consultas ineficaces que pueden afectar adversamente al rendimiento del servidor. Las consultas en los procedimientos almacenados se pueden ajustar

individualmente para generar planes de ejecución aceptables. Además del ajuste, los procedimientos almacenados bien escritos simplifican la solución de problemas del servidor como los bloqueos e interbloqueos.

- Rendimiento. En algunas situaciones, el uso de procedimientos almacenados puede producir mejoras en el rendimiento. Los motores de BD modernos generalmente tratan las instrucciones SQL con la misma eficacia que las instrucciones de los procedimientos almacenados; los administradores de BD mantienen más control sobre el rendimiento exigiendo el uso de procedimientos almacenados.

Los procedimientos almacenados que devuelven datos se llaman desde las funciones con nombre en el modelo de objetos EDM. Los procedimientos almacenados que actualizan datos se asignan a las entidades y asociaciones, y se llaman implícitamente cuando las operaciones que definen usan los métodos generados por el sistema, en el caso de que no se hayan implementado y asignado procedimientos almacenados. (MSDN, 2010)

1.7 Paradigma, Lenguaje, Plataforma e IDE de desarrollo.

1.7.1 Paradigmas de programación

Los paradigmas son un conjunto de conocimientos y creencias que forman una visión del mundo (cosmovisión), en torno a una teoría hegemónica en determinado periodo histórico. Cada paradigma se instaura tras una revolución científica, que aporta respuestas a los enigmas que no podían resolverse en el paradigma anterior. Una de las características fundamentales, su inconmensurabilidad: ya que ninguno puede considerarse mejor o peor que el otro. Además, cuentan con el consenso total de la comunidad científica que los representa. Los paradigmas cumplen una doble función, por un lado, la positiva que consiste en determinar las direcciones en las que ha de desarrollarse la ciencia normal, por medio de la propuesta de enigmas a resolver dentro del contexto de las teorías aceptadas. Por otro lado la función negativa del paradigma, es la de establecer los límites de lo que ha de considerarse ciencia durante el tiempo de su hegemonía. Un paradigma de programación es una colección de modelos conceptuales que juntos modelan el proceso de diseño y determinan, al final, la estructura de un programa.

1.7.2 La Programación Orientada a Objetos

(POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos). A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan (ni deben separarse) información (datos) y procesamiento (métodos). Dada esta propiedad de conjunto de una clase de objetos, que al contar con una serie de atributos definitorios, requiere de unos métodos para poder tratarlos (lo que hace que ambos conceptos están íntimamente entrelazados), el programador debe pensar indistintamente en ambos términos, ya que no debe nunca separar o dar mayor importancia a los atributos en favor de los métodos, ni viceversa. Hacerlo puede llevar al programador a seguir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen esa información por otro (llegando a una programación estructurada camuflada en un lenguaje de programación orientado a objetos).

1.7.3 Lenguaje de Programación CSharp

C# es el nuevo lenguaje de programación incluido por la Microsoft en la plataforma.NET. Es el mejor adaptado y creado exclusivamente para el trabajo sobre la plataforma porque las Framework Classes están programadas en él. Según Microsoft, C# es “un lenguaje de programación con la potencia de C, la productividad de Visual Basic y la elegancia de Java”. Si se analiza un código escrito en C# se puede observar su similitud con el lenguaje Java, pero a su vez posee la sintaxis utilizada en C++ e incorpora determinadas características que lo hacen muy potente (como la sobrecarga de operadores) combinada con la sencillez y facilidad de Visual Basic que lo hace muy productivo. (SOTOMAYOR, 2002)

1.7.4 Plataforma de Desarrollo Microsoft.NET

La plataforma.NET es una capa de software que se coloca entre el sistema operativo (SO) y el programador, se abstrae de los detalles internos del SO. Las características fundamentales de esta plataforma son las siguientes:

- Multilenguaje: Cualquier lenguaje de programación puede adaptarse a la plataforma.NET y ejecutarse en ella.
- Interoperabilidad: La interoperabilidad entre los distintos trozos de códigos escritos es total.
- Portabilidad: Debido a la abstracción del programador respecto a los SO una aplicación.NET puede ser ejecutada en cualquier SO que disponga de una versión de la plataforma.

La plataforma.NET está compuesta por tres partes fundamentales:

- El Common Language Runtime (CLR): Es el entorno de ejecución y constituye su núcleo. El CLR es donde se ejecutan las aplicaciones. Las mismas pueden estar escritas en diferentes lenguajes que ofrece.NET (C#.NET, Visual Basic.NET, C++.NET).
- Las Framework Classes: Esta capa provee al programador servicios estructuras y modelos de objetos para ADO.NET, entrada/salida, seguridad y manejo de documentos XML entre otros.
- ASP.NET: Es la capa más importante de la capa superior de la plataforma.NET. ASP.NET provee una plataforma robusta para el desarrollo de las aplicaciones Web y permite separar la lógica de la aplicación de la interfaz a diferencia de su antecesor ASP. De esta manera el programador puede centrarse en la lógica de la aplicación sin preocuparse de los detalles de la interfaz. (Morrison, 2008)

1.7.5 IDE de desarrollo Microsoft Visual Studio 2008

Visual Studio(VS) .NET es la herramienta que Microsoft distribuye junto a la plataforma y permite desarrollar y construir aplicaciones .NET. Este entorno está creado para poder manejar proyectos que usen más de un lenguaje a la vez, teniendo en cuenta la característica multilenguaje de la plataforma.
VS es la herramienta para el desarrollo de aplicaciones Web ASP.NET y aplicaciones de escritorio. Visual Studio Team System 2008 ayuda a los equipos de desarrollo de software mejorando las comunicaciones y colaboración durante todo el proceso de desarrollo. Esta herramienta añade .NET Framework 3.5 donde

se han introducido mejoras en servicios como Windows Presentation Fundation (WPF), Windows Communication Fundation (WCF), Windows Workflow Fundation (WWF) y CardSpace creados con la versión 3.0 del Framework y se han mejorado otros que no habían experimentado cambios desde la versión 2.0 como ASP.NET, ADO.NET y servicios criptográficos entre otros.
En conjunto se obtiene un sistema completo formado por la plataforma de servicios.NET, los compiladores de varios lenguajes, editores y diseñadores específicos para cada tipo de tarea, opciones de depuración y análisis de código, de modelado de sistemas y aplicaciones, de desarrollo y ejecución de pruebas, administración de bases de datos, entre otras. (Rodriguez, 2007)

1.7.6 Gestor de Base de Datos Oracle

Oracle es un Sistema Administrador de Base de Datos Relacionales (RDBMS, por sus siglas en inglés), desarrollado por la Corporación Oracle. Considerado como uno de los gestores de BD más completos del mundo. Como características básicas tiene:

- Soporte: Se encuentra disponible sobre varias plataformas (Windows, Linux, Unix).
- Fiabilidad: Continua disponibilidad de las aplicaciones y datos.
- Escalabilidad: Condición que tienen los sistemas de adaptarse al crecimiento de trabajo de manera fluida y hacerse más grande sin perder calidad en los servicios ofrecidos.
- Seguridad y protección de datos: características de seguridad que permite compartir la red de recursos de una empresa con la confianza de que la privacidad se mantiene.

Auto-gestión: Oracle automatiza muchas de las funciones de infraestructura de modo que un solo Administrador puede administrar cientos de servidores. (Gonzalez_Perez, 2008)

1.7.7 Características de tecnología para Oracle: Oracle Text

Estrechamente integrado con las estructuras de BD Oracle, que abarca varias columnas de texto. Permite los índices de contenido de la BD, sistema de archivos, Internet a través de URL o la fuente de programación. Los índices de texto de más de 150 formatos de archivo, como HTML, XML, PDF, Microsoft Word, Excel, PowerPoint. Es multilingüe, incluyendo inglés, neerlandés, francés, español, alemán, japonés, coreano, chino tradicional, chino simplificado y juegos de caracteres ASCII, UTF-8, JA165JIS,

GBK, BIG5. Integrado de seguridad y control de acceso con las normas de BD. Índices de catálogo diseñados para texto breve y la recuperación de datos numéricos, actualizado de forma automática. Consultas SQL incluyen a operadores como "cerca". Admite búsquedas booleanas, comandos de consulta XPath y XML anidados y atributos de búsqueda de valor. Tiene opciones para consultar con comodines y subcadenas (se debe establecer en el índice). Consulta con ajuste de distancia, en una misma frase o párrafo, fonética (que suenan igual) se pongan en venta, se pongan en venta borrosa, derivadas. Opciones para los sinónimos diccionario de sinónimos y más funciones complejas. Editable lista de palabras de parada. Búsquedas en puntuación específica. Relevancia de clasificación de resultados, basados en TF-IDF y la ponderación de tema. Muestra la versión HTML de archivos de código fuente con los términos de búsqueda resaltados. Admin Search utiliza Oracle Enterprise Manager. Facilita la clasificación automática, el filtrado, la agrupación, y opciones de personalización., la minería de textos para extraer los temas y los resúmenes de los documentos indexados, escalas con la replicación, la sincronización, la separación y el paralelismo.

1.8 Arquitectura, Servidores y Herramientas.

1.8.1 Patrón arquitectónico Modelo Vista Controlador (MVC)

El MVC es un patrón arquitectónico que sugiere la separación del software en tres componentes. Modelo, vista y controlador los cuales serán explicados brevemente.

- **Modelo:** Es la representación de la información que maneja la aplicación. El modelo en sí lo constituyen los datos puros y la lógica de los propios datos que puestos en el contexto del sistema proveen de información al usuario y en algunos casos a la propia aplicación.

- **Vista:** Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación Web, la “vista” sería una página HTML con contenido dinámico sobre la cual el usuario puede realizar sus operaciones.

- **Controlador:** Es la parte encargada de manejar y responder las solicitudes del usuario, procesando toda la información necesaria y modificando el Modelo en caso de ser necesario.

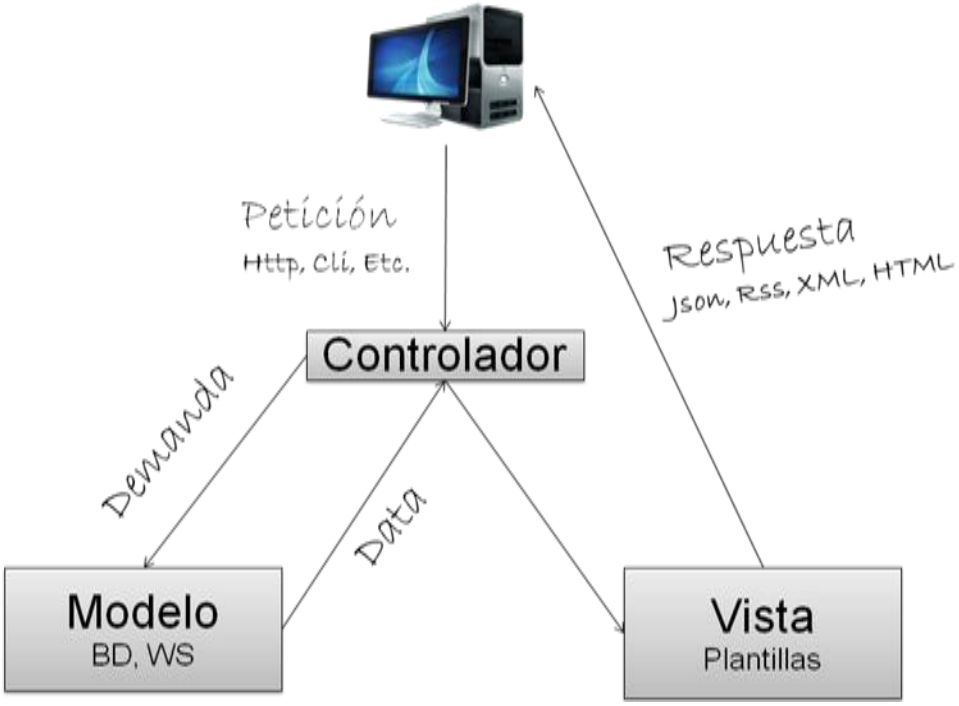


Figura 2: Patrón Modelo Vista Controlador

1.8.2 Deficiencias y Ventajas

Este modo de diseñar la aplicación ocasiona que se genere un mayor número de ficheros ya que se separa la capa de acceso a datos de las interfaces de usuario y de la gestión de funcionalidades. Además en algunos casos se pudiera complejizar la aplicación pero aporta también un gran número de ventajas como por ejemplo la separación del modelo de la vista, es decir, separar los datos de la representación visual de los mismos. Este patrón posibilita crear más de una vista para el mismo modelo, además de una conexión entre el modelo y sus vistas de manera dinámica, es decir, se produce en tiempo de ejecución y no en tiempo de compilación. Al utilizar este patrón si se agrega o se quita una funcionalidad no afecta a la aplicación y se facilita el mantenimiento en caso de errores.

1.8.3 Servidor Web

Un servidor Web es un programa que se ejecuta sobre una máquina que escucha peticiones Protocolo de Transferencia de Hipertexto (HTTP) y siempre envía una respuesta al cliente (navegador) que realizó la

petición, devolviendo un hipertexto, los cuales pueden ser páginas Web o HTML: formularios, imágenes, texto, animaciones entre otras.

1.8.4 Internet Information Server (IIS)

Es una serie de servicios que funcionan con ordenadores que tienen Windows y se encuentra destinado a ofrecer servicios. Instalar IIS nos permite servicios como Protocolo Transferencias Archivos (FTP), Protocolo Simple de Transferencia de Correo (SMTP), Protocolo para la Transferencia de Noticias en Red (NNTP), HTTP/HTTPS(Protocolo de Transferencia de Hipertexto seguro) . Por lo tanto una computadora que contenga IIS se convierte en un servidor de Internet o Intranet y en él se pueden publicar páginas Web tanto locales como remotas.

1.9 Herramientas CASE

Las herramientas CASE (por su nombre en inglés, Computer Aided Software Engineering) permiten incrementar la productividad y el control de la calidad en cualquier proceso de elaboración de software, ya que transforman la actividad de desarrollar software en un proceso automatizado. A medida que los sistemas que hoy se construyen se tornan más complejos, las herramientas de modelado con UML ofrecen muchos beneficios para todos los involucrados en un proyecto.

Los objetivos fundamentales de estas herramientas son:

- Permitir la aplicación práctica de metodologías, lo que resulta muy difícil sin emplear herramientas.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento del software.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes de software.
- Permitir un desarrollo y un refinamiento -visual- de las aplicaciones, mediante la utilización de controles gráficos (piezas de código reutilizables).

Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de

errores entre otras. Son sistemas de software que proporcionan ayuda automatizada a las actividades del proceso de desarrollo de software.

1.9.1 Erwin 7.1

ERwin es una herramienta de diseño de BD. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la BD diseñada, ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la BD. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y triggers para los principales tipos de BD.

ERwin hace fácil el diseño de una BD. Los diseñadores de BD sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves.

Más que una herramienta de dibujo, ERwin automatiza el proceso de diseño de una manera inteligente. Por ejemplo, ERwin habilita la creación de un diccionario de atributos reusables, asegurando la consistencia de nombres y definiciones para su BD.

Se mantienen las vistas de la BD como componentes integrados al modelo, permitiendo que los cambios en las tablas sean reflejados automáticamente en las vistas definidas. La migración automática garantiza la integridad referencial de la BD.

ERwin establece una conexión entre una BD diseñada y una BD, permitiendo transferencia entre ambas y la aplicación de ingeniería reversa. Usando esta conexión, ERwin genera automáticamente tablas, vistas, índices, reglas de integridad referencial (llaves primarias, llaves foráneas), valores por defecto y restricciones de campos y dominios.

ERwin soporta principalmente BD relacionales SQL y BD que incluyen Oracle, Microsoft SQL Server, Sybase, DB2, e Informix. El mismo modelo puede ser usado para generar múltiples BD, o convertir una aplicación de una plataforma de BD a otra.

1.9.2 Toad for Oracle 9.6

Toad for Oracle es una poderosa herramienta de bajo costo que facilita y acelera el desarrollo de BD y aplicaciones mientras simplifica las tareas cotidianas de administración. No importa si es un desarrollador de PL/SQL, un desarrollador de aplicaciones, un DBA o un analista de negocios, Toad for Oracle le ofrece

funciones específicas para hacerlo más productivo que nunca. Toad mejora la productividad con toda la funcionalidad que usted necesita para generar y ejecutar consultas, crear y modificar objetos de la BD y desarrollar y depurar código SQL y PL/SQL. Incluso las tareas cotidianas, tales como: importación/exportación de datos, comparación de esquemas y actualización de estadísticas las cuales se realizan con mayor rapidez y facilidad con Toad. Plus. Además Toad ofrece integración con Knowledge Xpert™ para incorporar la experiencia en BD Oracle de reconocidos expertos en este tema. También ayuda a incrementar la calidad de sus aplicaciones de BD. Las funciones de afinación SQL integrada y la revisión automática de código PL/SQL brindan a los usuarios de todos los niveles la experiencia necesaria para generar código de calidad. Además, Quest ofrece soporte de una comunidad interactiva tanto de colegas como de expertos de la industria mediante grupos de discusión en línea, la página Web y boletines Quest Pipelines y eventos diseñados para la comunidad de usuarios Toad.

1.10 Conclusiones

En este capítulo se realizó un estudio sobre las tendencias actuales para almacenar información, los mecanismos para acceder a las BD y las soluciones existentes que implementan la técnica ORM. Se pudo evidenciar que las BD relacionales son las más utilizadas actualmente y que para acceder a estas desde los lenguajes de programación tradicionales existen diferentes mecanismos, que a pesar de sus diferencias todos tienen como objetivo principal implementar y abstraer los protocolos de comunicación entre las aplicaciones informáticas y las BD, simplificando de cierta forma la tarea de consultarlas. También se describió la técnica ORM como una solución a la diferencia de impedancia entre el modelo relacional y el orientado a objeto y se presentaron tres herramientas que implementan esta técnica, una vez analizadas y teniendo en cuenta las exigencias de la aplicación a la cual se le perfeccionará la capa de acceso a datos se llega a la conclusión de que la herramienta usada será el EF**.** Éste representa una solución ORM muy completa, de gran madurez y mucha utilidad. Además ofrece numerosas ventajas para el desempeño de la capa de acceso a datos como las mencionadas en el capítulo.

En este capítulo se presentaron además las características, tecnologías, lenguajes, técnicas y herramientas del sistema HelpDesk al cual se le perfeccionará la capa de acceso a datos.

Como lenguaje de programación se utiliza el C# sobre el IDE Visual Studio 2008, como SGBD Oracle ya que con estas herramientas se realizó la aplicación. Como herramienta Case, ERwin para revisar y perfeccionar el diseño de la BD, toad for oracle para administrar la BD. Finalmente de acuerdo con el estudio realizado se empleó como proveedor para EF **dotConnect for Oracle 5.35.79.** En cuanto al tipo de carga aclarar que no se específica el tipo de carga, sino que se emplea la que trae por defecto en este caso la carga perezosa.

CAPÍTULO 2: CARACTERÍSTICAS DEL MÓDULO DE ACCESO A DATOS HELPDESK

2.1 Introducción

En este capítulo se estudian detalladamente las características de la aplicación que obligan a utilizar determinados programas sobre los que esta se inició. Se describen las características fundamentales del sistema, los requisitos funcionales y no funcionales y se mencionan algunas de las estrategias que se establecen para implementar el acceso a datos en una aplicación informática, se específica el objetivo principal, que es utilizar la técnica ORM.

2.2 Características del sistema HelpDesk.

La metodología que se utilizó para desarrollar esta aplicación fue RUP por ser muy detallado, lo que posibilita documentar todo el proceso minuciosamente, además se ajusta a grandes proyectos ya que su realización es a largo plazo, siendo así muy conveniente su elección y utilización.

HelpDesk está desarrollado utilizando la Plataforma.NET porque se ajusta a las necesidades ofreciendo solidez en la programación del lado del servidor y en la misma se han desarrollado gran parte de las aplicaciones que el MININT tiene en explotación actualmente. Como lenguaje de programación se utilizó el C#.NET ya que es el más adaptado a la plataforma y en él se encuentran programadas la mayoría de las librerías nativas del .NET y como IDE Visual Estudio 2008. El Gestor de BD seleccionado fue el Oracle por ser considerado como uno de los más potentes. Ofrece múltiples funcionalidades como líneas de comando PL/SQL y la creación de Procedimientos Almacenados, Funciones Disparadoras, Funciones y Vistas que facilitan la programación de sistemas.

2.2.3 Información que se Maneja.

La información que se maneja en la aplicación es la referente a la solicitud de servicio, que puede ser una interrupción, una petición de servicio u otra cualquiera que la organización defina. Otra información que se maneja son los datos de los usuarios que hacen la solicitud y de los que la gestionan y controlan.

2.3 Requerimientos

Condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta (IEEE). Existen tres tipos de requerimientos:

- Un requerimiento funcional puede ser una descripción de lo que un sistema debe hacer. Este tipo de requerimiento específica algo que el sistema entregado debe ser capaz de realizar.
- Un requerimiento No Funcional: de rendimiento, de calidad u otro; específica algo sobre el propio sistema, y cómo debe realizar sus funciones. Algunos ejemplos de aspectos solicitables son la disponibilidad, el testeo, el mantenimiento y la facilidad de uso.
- Otros tipos de limitaciones externas, que afectan en una forma indirecta al producto. Estas pueden ir desde la compatibilidad con cierto sistema operativo hasta la adecuación a leyes o regulaciones aplicables al producto.

Una colección de requerimientos describe las características o atributos del sistema deseado.

2.3.1 Requerimientos Funcionales

Teniendo en cuenta las capacidades o condiciones que el sistema debe cumplir se mencionan los requerimientos funcionales, utilizando el patrón CRUD.

Gestionar Categorías.

R1.1 Insertar Categoría.

R1.2 Modificar Categoría.

R1.3 Eliminar Categoría.

Gestionar Estado de las solicitudes.

R2.1 Insertar Estado de la Solicitud.

R2.2 Modificar Estado de la Solicitud.

R2.3 Eliminar Estado de la Solicitud.

Gestionar Modo

R3.1 Insertar Modo.

R3.2 Modificar Modo.

R3.3 Eliminar Modo.

Gestionar Departamentos.

R4. 1 Insertar Departamento.

R4. 2 Modificar Departamento.

R4. 3 Eliminar Departamento.

Gestionar Subcategoría

R5. 1 Insertar Subcategoría.

R5. 2 Modificar Subcategoría.

R5. 3 Eliminar Subcategoría.

Gestionar Elemento.

R6. 1 Insertar Elemento.

R6. 2 Modificar Elemento.

R6. 3 Eliminar Elemento.

Gestionar Rol.

R7. 1 Insertar Rol.

R7. 2 Modificar Rol.

R7. 3 Eliminar Rol.

Gestionar Usuario.

R8. 1 Insertar Usuario.

R8. 2 Modificar Usuario.

R8. 3 Eliminar Usuario.

Gestionar Tipo de Solicitud

R9. 1 Insertar Tipo de Solicitud.

R9. 2 Modificar Tipo de Solicitud.

R9. 3 Eliminar Tipo de Solicitud.

Gestionar Organización.

R11. 1 Insertar Organización.

R11. 2 Modificar Organización.

R11. 3 Eliminar Organización.

Gestionar Grupo de Soporte

R12. 1 Insertar Grupo de Soporte.

R12. 2 Modificar Grupo de Soporte.

R12. 3 Eliminar Grupo de Soporte.

Gestionar Prioridad.

R13. 1 Insertar Prioridad.

R13. 2 Modificar Prioridad.

R13. 3 Eliminar Prioridad.

Administrar Solicitud.

R14. 1 Crear nueva solicitud de servicio.

R14. 2 Modificar una solicitud existente

Gestionar Soluciones

R15. 1 Insertar Solución

R15. 2 Modificar Solución

R15. 3 Eliminar Solución

Realizar Voto

R16 Realizar voto

Buscador

R17 Realizar Búsquedas

2.3.2 Requerimientos No Funcionales.

Los requerimientos 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 o confiable. Entre los requerimientos No Funcionales se encuentran:

Seguridad. La información manejada por el sistema contará de protección ante intrusos y accesos no autorizados, será vista únicamente por aquellos usuarios que tengan derecho a verla. El sistema controlará los diferentes niveles de acceso y funcionalidad de usuarios al sitio, priorizando la identificación del usuario antes de que sea capaz de realizar cualquier acción sobre el sistema.

Restricciones de diseño: El IDE de desarrollo Visual Studio.Net 2008 y C# como lenguaje de programación. ASP.NET. La arquitectura que se usará será Modelo Vista Controlador (MVC).

Portabilidad. El sistema deberá ser multiplataforma, es decir debe poder ejecutarse sobre los Sistemas Operativos Windows y Linux.

Usabilidad. La aplicación Web podrá ser navegada por cualquier usuario con conocimientos básicos de computación y sobre el ambiente Web. Será flexible y de fácil aprendizaje, logrando que los usuarios tengan una plena satisfacción con su uso.

Rendimiento. La aplicación para cada solicitud del usuario debe tener una respuesta en pocos segundos para lograr que la gestión de la información sea efectiva. Las páginas solicitadas no contendrán grandes volúmenes de imágenes para evitar retrasos innecesarios. Además debe permitir conexiones simultáneas.

Apariencia o interfaz externa. El diseño de la interfaz debe ser sencillo, amigable, de fácil navegación para el usuario y con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones. El producto debe ser legible con colores agradables a la vista del usuario y mantener un formato estándar en todas las páginas.

Confiabilidad. Dentro de este requerimiento se encuentran la integridad y la disponibilidad.

- Integridad: La información será protegida contra corrupción y estados inconsistentes.
- Disponibilidad: Estará disponible las 24 horas del día durante todo el año. Los usuarios autorizados tendrán acceso a la información siempre que lo deseen.

Software. Se debe tener en cuenta los programas que deben tener instalado la PC cliente y la PC servidora.

Cliente:

- Navegador Mozilla Firefox 2.0, Google Chrome, Iceweal, Zafari.
- Sistema operativo Linux ó Windows 98 ó superior.

Servidor:

- Sistema operativo Windows Server 2003, Windows Vista o Windows Server 2008.
- Servidor Web IIS 6.0 o superior.
- Servidor de Base de Datos Oracle versión 11g.

Hardware. Se debe tener en cuenta los requerimientos mínimos para el cliente y para el servidor.

Cliente:

- Procesador Pentium III a 1GHz de velocidad de procesamiento y 128MB de memoria RAM.
- Tarjeta de red.

Servidor:

- Procesador Intel Pentium Dual Core a 3.0HGz de velocidad de procesamiento y 2Gb de memoria RAM.
- 50Gb de espacio libre en disco.
- Tarjeta de red.

2.4 Estrategias de acceso a datos

Establecer una estrategia para crear o mejorar capas de acceso a datos en aplicaciones informáticas es tal vez una de las decisiones más difíciles ya que en una aplicación estándar un gran por ciento del código generado está relacionado con el código para guardar y recuperar la información inherente a su dominio. Una estrategia inmediata a emplear será, utilizar directamente algún API de acceso a datos, ADO.Net para el desarrollo en la plataforma .Net.

Además se puede añadir que estas APIs de acceso a datos representan una capa liviana sobre los sistemas de BD relacionales, y como tal se preocupan solamente de abstraer e implementar los mecanismos y protocolos de comunicación; y todo el resto de los aspectos relacionados con la interacción con las bases de datos debe ser manejado por los desarrolladores.

Una estrategia recomendada y compatible con las buenas práctica de diseño es crear un modelo de objeto del dominio que represente el ciento por ciento de la información que maneja la aplicación y utilizar la técnica ORM para guardar y recuperar estos objetos contra la BD.

2.4.1 Ventajas que ofrece la estrategia

- Ya no se maneja información tabular sino puros objetos.
- La implementación del acceso a datos se realiza con un enfoque orientado a objeto.
- Se puede establecer la relación de herencia
- Se puede mapear una jerarquía de clases a una sola tabla o crear una tabla por cada clase concreta.
- Soporta el mapeo de todo tipo de relaciones que pueden existir entre los objetos de dominio, como asociaciones de uno a mucho, uno a uno, muchos a muchos.
- Se disminuye las veces que se accede a la BD, ya que se guardan en memoria los objetos que son accedidos varias veces.
- Soporta múltiples dialectos SQL, se puede independizar por completo la aplicación del SGBDR que se utilice.
- La aplicación puede guardar sus datos en SQL Server, MySQL, PostgreSQL, y demás gestores con solo cambiar la configuración correspondiente para cada uno de ellos.

- Se aprovechan todas las capacidades que brinda la técnica de ORM. En cuanto a simplificación del acceso a BD relacionales.

2.4.2 Estrategia a seguir

Teniendo en cuenta estas dos posibles estrategias, se utilizará como mecanismo de acceso a datos en la construcción de los módulos, la técnica de los ORM. Para ello se realizó un estudio de las mismas, seleccionando la herramienta EF como la más idónea, por sus prestaciones, estabilidad y su flexibilidad para realizar el mapeo entre clases y tablas de la BD.

2.4.3 Procedimiento para trabajar con Entity Framework.

En el capítulo anterior se hace referencia a las características generales del EF y a continuación se muestra la forma de trabajar con la misma.

Existen 3 pasos fundamentales a la hora de trabajar con estas herramientas de mapeo:

- Diseñar el modelo de objetos.
- Modelar el relacional.
- Aplicar los mapeos correspondientes.

Para trabajar con EF es fundamental saber que es una herramienta que permite manejar un modelo de objetos (con relaciones entre ellos), expresados en .NET que persisten, gracias a Providers escritos, a distintas BD. Específicamente se utiliza el dotConnect for Oracle 5.35.79**.**

Permite modelar en el nivel adecuado de abstracción ejemplo de esto los EDM que nos permite hacer los modelos de objetos que representan las entidades de negocio y sus relaciones. Luego de tener el modelo EDM es necesario cargarlo con los datos, este paso clave es lo que se conoce como mapeo. Los EDM presentan los servicios de objetos que permite tratar los datos como objetos y olvidar las filas y columnas a las que los desarrolladores acostumbran. Luego de tener un mapeo correcto de la BD, la implementación es completamente a nivel de objetos.

2.5 Distribución, estructura y componentes de la capa de persistencia

2.5.1 Distribución.

HelpDesk está dividido en cinco módulos Figura #3: administración, negocio, gestionar soluciones, gestionar solicitudes y servicios de aplicación de los cuales 4 en su desarrollo contendrán el uso de la herramienta seleccionada; en este caso EF.

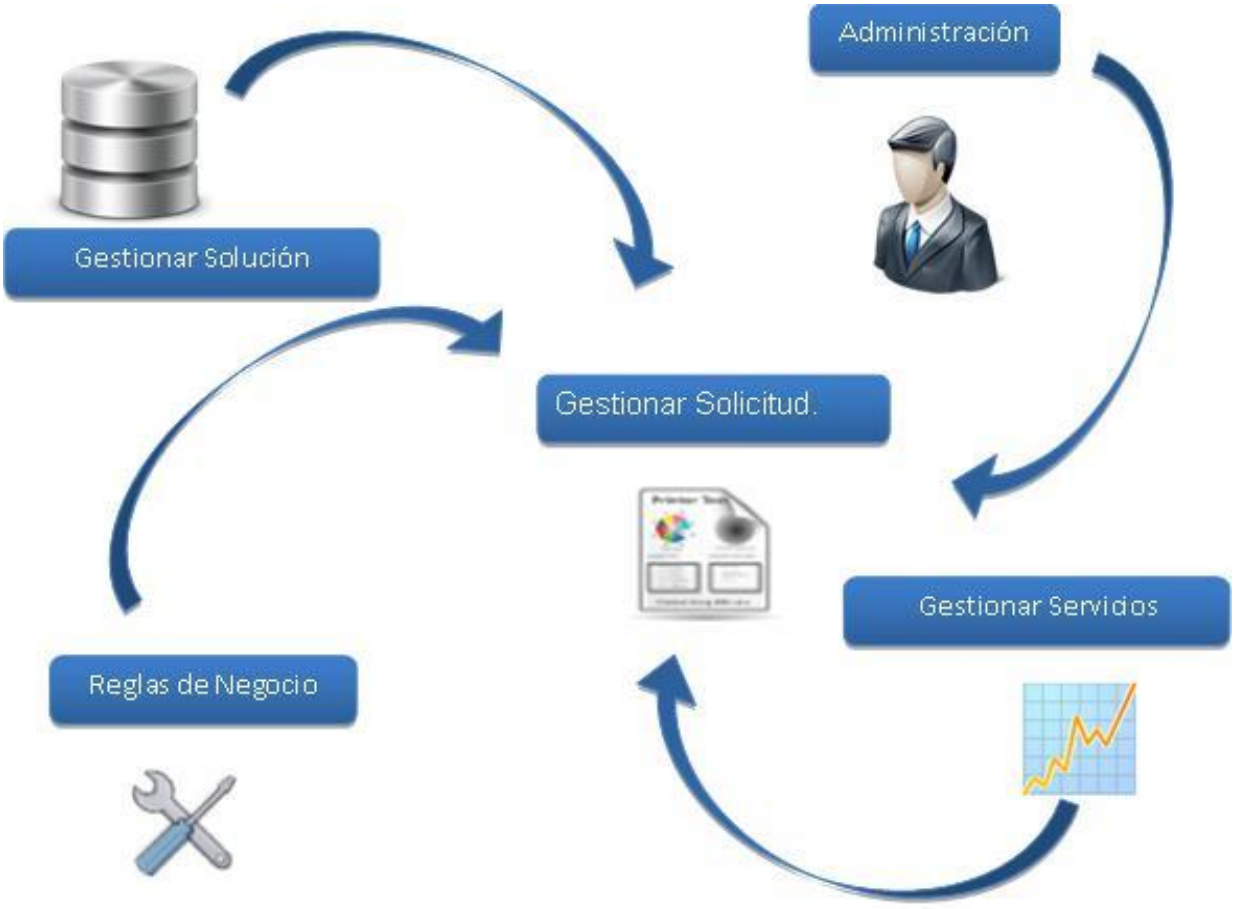


Figura 3: Distribución de módulos para el desarrollo de la capa persistente

2.5.2 Estructura

La capa de acceso a datos o capa de persistencia como también se suele llamar es el grupo de interfaces, clases y componentes de configuración que le permiten manipular los objetos de persistencia de la aplicación; entiéndase por manipularlos principalmente guardarlos, eliminarlos, modificarlos y obtenerlos según sea al caso que se esté manejando. Además es esta capa la que vincula directamente a la aplicación con la BD y es la base de todas las demás capas superiores. En este caso se centrará la explicación sobre los módulos de administración, negocio, solicitud, soluciones y servicio: Los componentes y clases de dichos módulos se encuentran dentro de la aplicación en los siguientes paquetes y carpetas:

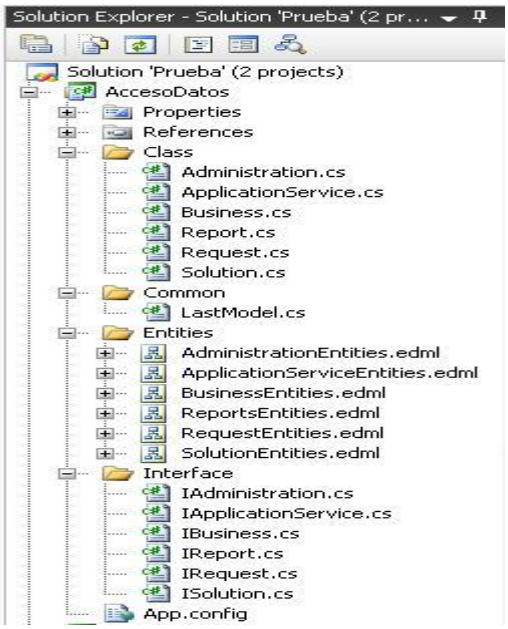


Figura 4: Distribución del Proyecto

2.5.3 Componentes

Para satisfacer el objetivo principal de perfeccionar la capa de acceso a datos del sistema HelpDesk aplicando un ORM se creó el proyecto como una librería de clases para hacer más fácil la integración. Los EDM se desarrollaron uno por cada módulo.

2.6 Construcción de EDM

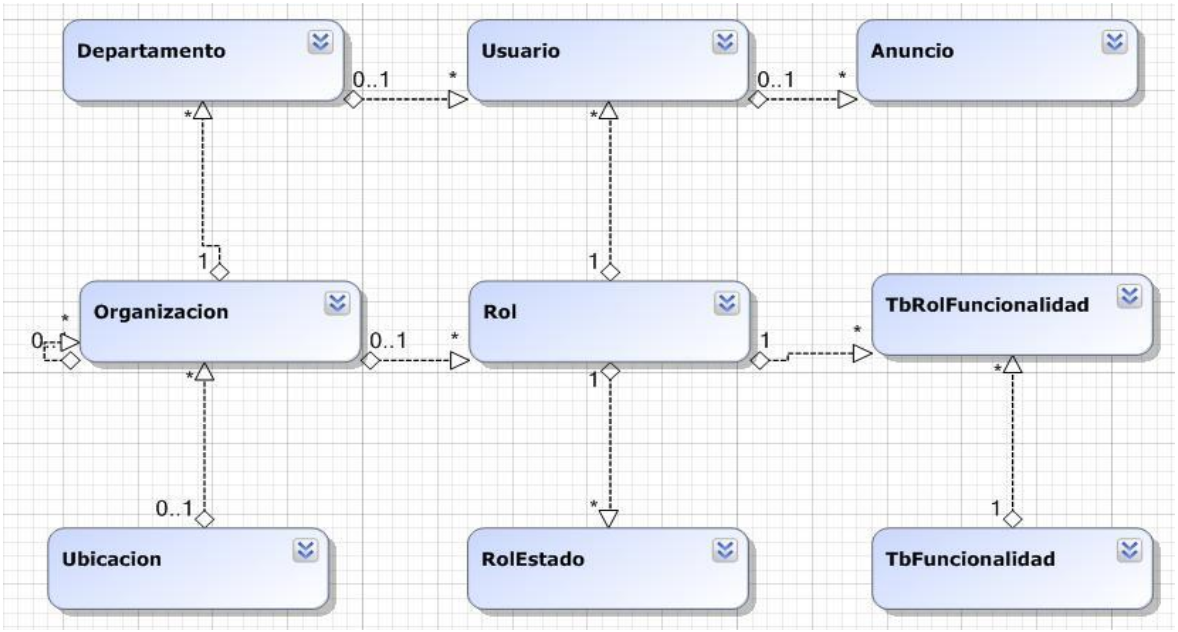


Figura 5: EDM para el Módulo Administración

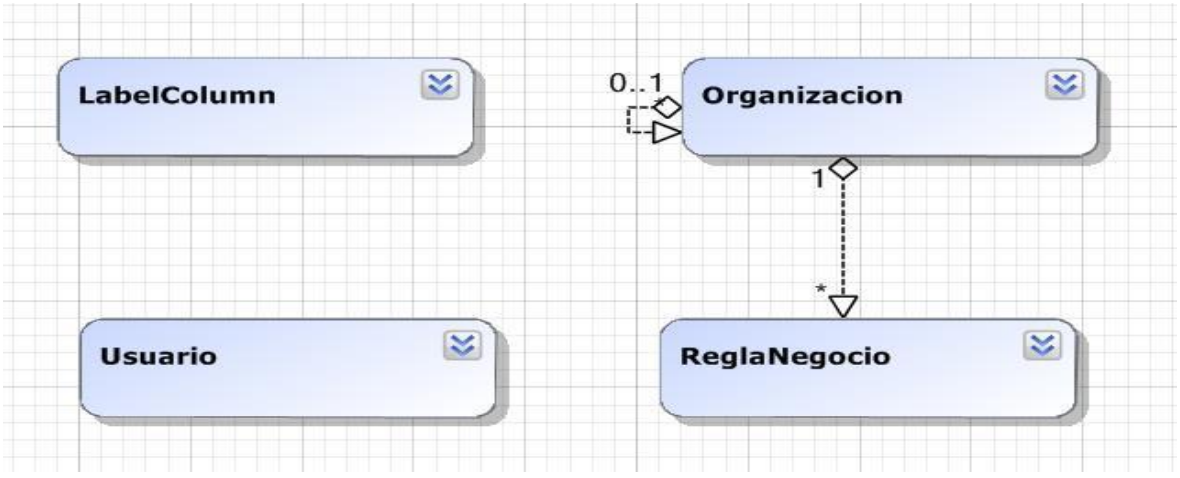


Figura 6: EDM para el Módulo Negocio

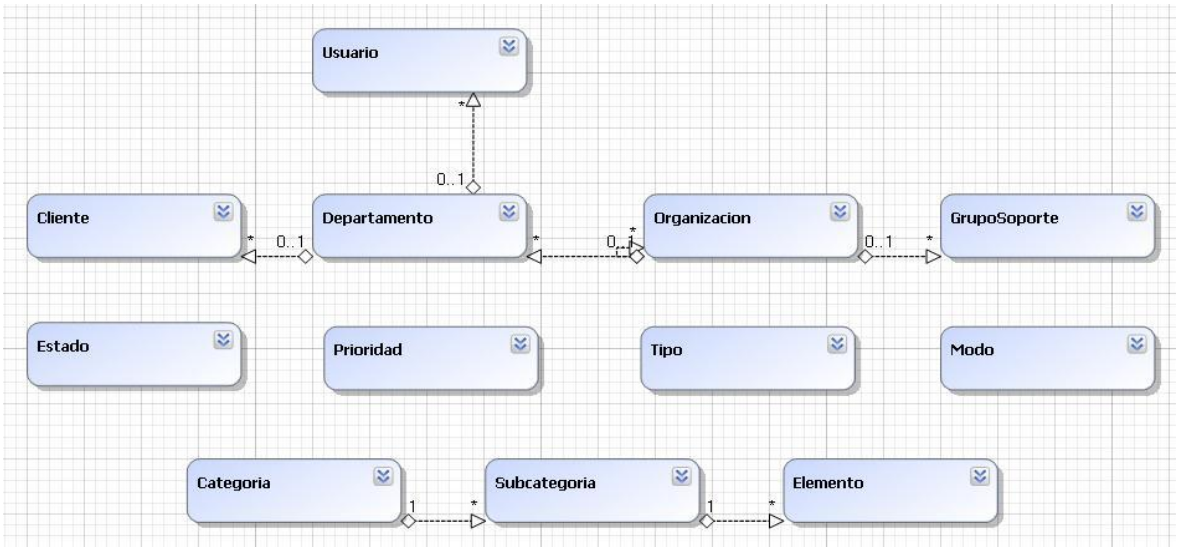


Figura 7: EDM para el Módulo Solicitud de Servicio

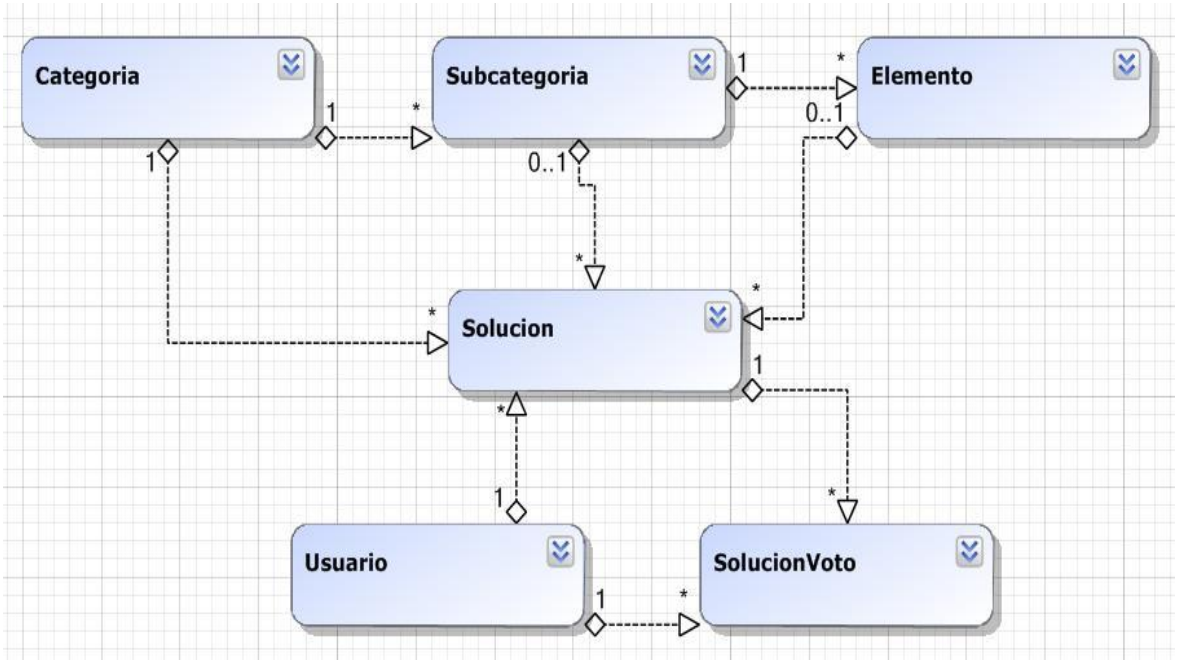


Figura 8: EDM para el Módulo Soluciones

Módulo de Incidencias

El Módulo de incidencias no muestra sus objetos mapeados debido a que son procedimientos almacenados pertenecientes al paquete PK_Solicitud. En éste paquete se mapean los procedimientos Editar Solicitud e Insertar solicitud y Historial de solicitud.

2.7 Estrategias de Seguridad

El acceso a las funcionalidades, se realiza mediante interfaces que controlan las diferentes funciones de los módulos independientes como medida de seguridad, de forma que los cambios que ocurran no afecten al sistema. El acceso a la aplicación será controlada por el Administrador para evitar daños o pérdida en la información contenida, siendo este el encargado de crear usuarios, estableciendo los permisos en dependencia del rol al cual son asignados.

2.8 Tratamiento de Excepciones

El tratamiento de errores brinda a la aplicación una mejor apariencia ante los usuarios y posibilita su buen funcionamiento. Para establecer una mayor organización se creará una clase lastModel localizada en la carpeta Common en la cual estarán todas las funcionalidades útiles para los cinco módulos en caso de haberlas, lo cual permitirá en este caso un mayor control y organización. Las excepciones se trabajan en la aplicación de prueba, de forma que se verifican las de la herramienta EF y las del Oracle, de no haber ninguna se va directo al código de las funcionalidades y se manejan como si fuesen las del Oracle pasando a las capas superiores que son las encargadas.

2.9 Conclusiones

En el desarrollo de este capítulo se trataron aspectos de gran importancia en el perfeccionamiento de la capa de acceso a datos del sistema HelpDesk. La aplicación está estructurada en 5 módulos de ellos el objetivo lo constituyen las secciones: administración, servicio, solución, incidencias y negocio a los cuales se le implementarán los CRUD utilizando la herramienta EF que permitirá tratar cada elemento de la BD como un objeto, facilitando así el trabajo de los desarrolladores. Se explican algunas medidas de

seguridad que se aplican como la autenticación de usuarios con acceso restringido en dependencia de su rol. Además de abordar cómo se tratan los errores para mejor comprensión de estos en caso de futuras mejoras.

CAPITULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el contenido de este capítulo se establece el estándar de codificación a seguir y se realizará un análisis con el objetivo de describir la solución dada al problema planteado. También se realizará una breve descripción de cada uno de los métodos que se implementaron y se abordarán algunos ejemplos representativos. Se describen las tablas y se presenta el modelo físico de la BD. Además se especifican las pruebas que fueron hechas al sistema para comprobar el funcionamiento de la propuesta de acceso a datos.

3.2 Estándares de Codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

Los estándares de codificación son de una gran importancia en los proyectos sobre todo para los desarrolladores.

Los mismos facilitan en gran medida la comprensión del código escrito lo que hace más sencillo la revisión del código y permiten su posterior mantenimiento con un menor costo. Es de vital importancia para la correcta integración entre los diferentes componentes de la aplicación además de que muchos de los framework a los que uno normalmente se integra presuponen la correcta utilización de los mismos. El estándar utilizado en muchos casos en el proyecto esta basado en el que ofrecen para **DOTNET** (Hommel). Además de algunos acuerdos que tomaron los desarrolladores para nombramientos y otras.

Tipos de	**Reglas para nombrar**	**Ejemplos**

identificadores		
Paquetes	En forma sustantiva, letra inicial mayúscula y en inglés. Cada paquete tiene sus propias reglas.	Class, Interface, Common
Clases	Sustantivo, en inglés y si es compuesto seguido.	Administration, AplicationService
Interfaces	Sustantivo en inglés con la primera letra I y la inicial mayúscula.	IBusiness, IReport
Métodos	Localizados en regiones en inglés y corridos en caso de ser compuestos y con cada inicial de palabra en mayúscula.	AddUsser, AddOrganization
Variables	Minúscula y diminutivas.	var org
Regiones	Definen su contenido en español y dentro de 3 asteriscos.	***Adicionar***

3.3 Diagrama de Despliegue

El Modelo de Despliegue describe la distribución física del sistema en términos de cómo las funcionalidades se distribuyen entre los nodos de cómputo sobre los que se va a instalar el sistema. Estos nodos son utilizados para identificar cualquier servidor, terminal de trabajo u otro hardware host que se utiliza para desplegar componentes en el ambiente de producción. El Modelo de Despliegue proporciona un modelo bien detallado donde los componentes se despliegan a lo largo de la infraestructura del sistema, detallando además las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otras informaciones relacionadas al despliegue del sistema propuesto.

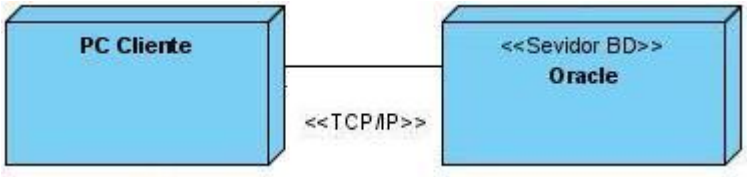


Figura 9: Diagrama de Despliegue

3.4 Normalización de la BD

Uno de los aspectos que debe contener la solución propuesta es la correcta normalización de la BD, para ello se realizó una ingeniería inversa a toda la BD con la herramienta case ERwin, donde se llevaron a cabo operaciones que facilitarán obtener un diseño de BD organizado, bien relacionado y limpio. Para lograr esto se normalizó la BD, se eliminaron varias tablas, campos y relaciones innecesarios y se agregaron relaciones entre tablas que estaban aisladas.

3.5 Modelo Físico

Partiendo de la normalización de la BD se realiza Modelo Físico que está conformado por las tablas que contiene la BD del sistema con sus atributos y relaciones entre ellas.

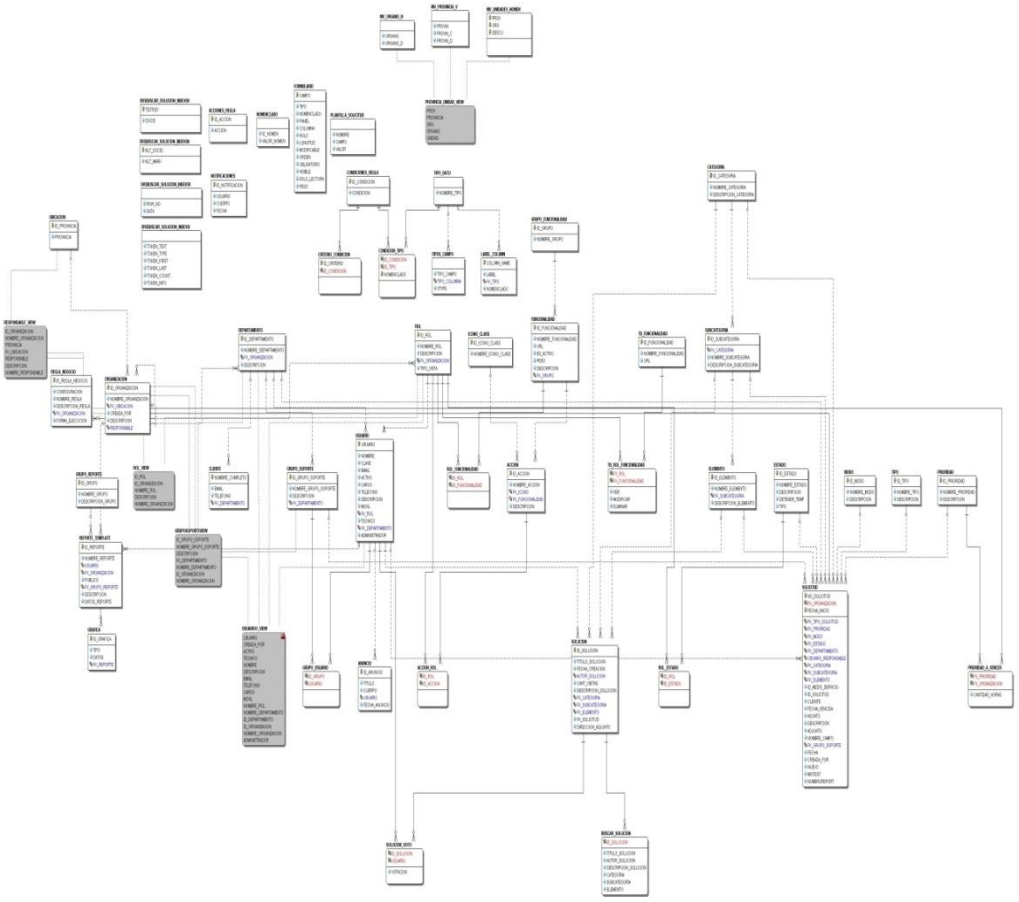


Figura 10: Diagrama de Entidad-Relación

3.6 Propuesta de solución

La selección de una propuesta que de solución a determinado problema suele ser en ocasiones tedioso, debido a que puede que exista más de una solución y la idea es seleccionar aquella que se adecue a nuestra situación. En el capítulo anterior se expone la estrategia de datos a emplear, la cual forma parte indisoluble de nuestra solución.

Entre los aspectos fundamentales de la propuesta se encuentran el uso de la técnica ORM- EF y procedimientos almacenados.

Para el perfeccionamiento de la capa de acceso a datos se toma en cuenta las peculiaridades de la herramienta se adoptan algunas estrategias en el desarrollo de algunas funcionalidades como:

Métodos Adicionar

Tomando en cuenta que la herramienta EF no tiene soporte actualmente para las llaves autoincrementales, se crean secuencias que se incrementan mediante procedimientos almacenados y posteriormente se mapean para lograr con una sola consulta a la BD realizar el proceso de adicionar, por lo que es una solución más factible. Para esto se utiliza la herramienta Toad for Oracle 9.6 en la cual se crea un paquete llamado PK_AdicionarEF donde estarán guardados todos los procedimientos para adicionar.

3.7 Funcionalidades desarrolladas

La implementación está dada siguiendo la solución propuesta anteriormente. Las funcionalidades que se desarrollaron fueron:

3.7.1 Módulo Administración

- Gestionar Usuario.
- Gestionar Departamentos.
- Gestionar Rol.
- Gestionar Organización.

3.7.2 Módulo Solución

- Gestionar Solución
- Realizar Voto

3.7.3 Módulo Negocio

- Gestionar Regla

3.7.4 Servicio

- Gestionar Estado
- Gestionar Prioridad.
- Gestionar Categorías.
- Gestionar Subcategoría

- Gestionar Elemento.
- Gestionar Grupo de Soporte
- Gestionar Modo
- Gestionar Cliente
- Gestionar Tipo de Solicitud

3.7.5 Solicitud

- Administrar Solicitud
- Insertar Historial de solicitud

3.8 Implementación del Buscador de la aplicación

En el desarrollo del buscador de la aplicación se utilizó la tecnología Oracle text que trae implementada el Oracle, la cual mejora la eficiencia de búsquedas. Se realizó partiendo de la creación de un datastore de tipo multi_column_datastore que permite almacenar el texto de varias columnas. Posteriormente se creó un índice que tiene como parámetro este datastore y a través de los operadores Contains y Fuzzy se realizan las operaciones de búsqueda.

3.10 Modelo de prueba

Las pruebas son el proceso de ejercitar un programa con la intención de especificar y encontrar errores previos a la entrega del sistema al usuario final. Estas van encaminadas a garantizar la calidad del software en todo momento del desarrollo. Durante este flujo de trabajo se verifica el resultado de la implementación, planificando, diseñando e implementando los casos de prueba.

3.10.1 Descripción de las pruebas

Siempre que se desarrolla algún tipo de software, hay que tener en cuenta las pruebas a realizar, con esto se asegura que nuestro programa o librería funcionen correctamente. Unas de las pruebas más comunes son las directas. Para probar el correcto funcionamiento de este módulo de acceso a datos se desarrolló una sencilla aplicación desktop, con una interfaz para cada módulo, en cada una de ellas se tendrán los distintos campos para confirmar el correcto funcionamiento del proyecto.

La prueba de unidades se plantea a pequeña escala, y consiste en ir probando uno a uno los diferentes módulos que constituyen una aplicación. Esta prueba la llevan a cabo los implementadores sobre las unidades mínimas desarrolladas por ellos, estas unidades pueden ser clases, métodos, propiedades o componentes. Estas unidades se prueban separadas unas de otras y básicamente se realizan estas pruebas durante la implementación del software. Para la realización de las pruebas se debe tener en cuenta que:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.
- Las pruebas de unidades son orientadas en la mayoría de los casos a nivel de clases, ya que las clases representan la unidad básica por excelencia en el desarrollo orientado a objetos. Por lo que se realizarán pruebas de unidad a las clases, como también se le realizarán pruebas de unidad a cada módulo como un todo una vez que se termine su desarrollo.

3.10.2 Criterio de Prueba aplicado

Para el desarrollo de las pruebas de la capa de datos se escogen las de caja negra. Éstas se denominan:

- Pruebas de caja opaca
- Pruebas funcionales
- Pruebas de entrada/salida
- Pruebas inducidas por los datos

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, entre otros) Este comentario no consta para que sean útiles en cualquier módulo del sistema.

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea de cobertura para esto se tratan todas las entradas no válidas de modo que siempre se mostrará un mensaje que le indique el estado de su operación. El problema con las pruebas de caja negra no suele estar en el número de funciones proporcionadas por el módulo (que siempre es un número muy limitado en diseños razonables); sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio (por ejemplo, un entero).

Ejemplo de prueba para la tabla Gestionar Categoría.

Figura 11 Interfaz Principal

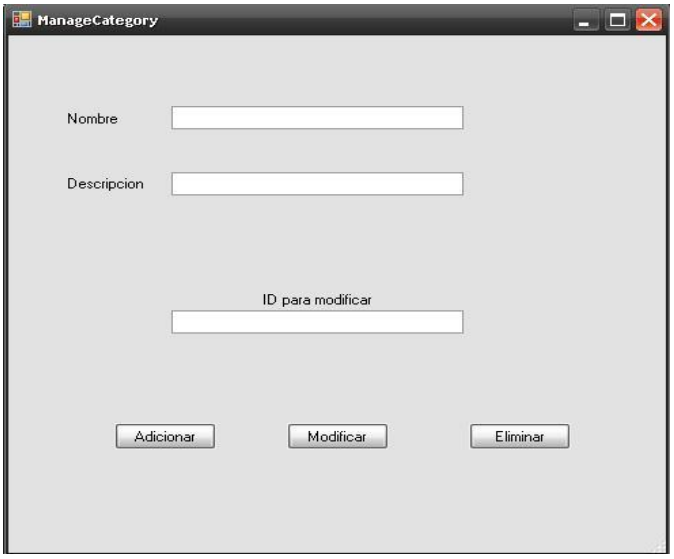


Figura 12: Interfaz Gestionar Categoría

Se introducen los datos correctamente y la interfaz mostrará un mensaje verificando su correcta interacción. De lo contrario se le especificará el error cometido.

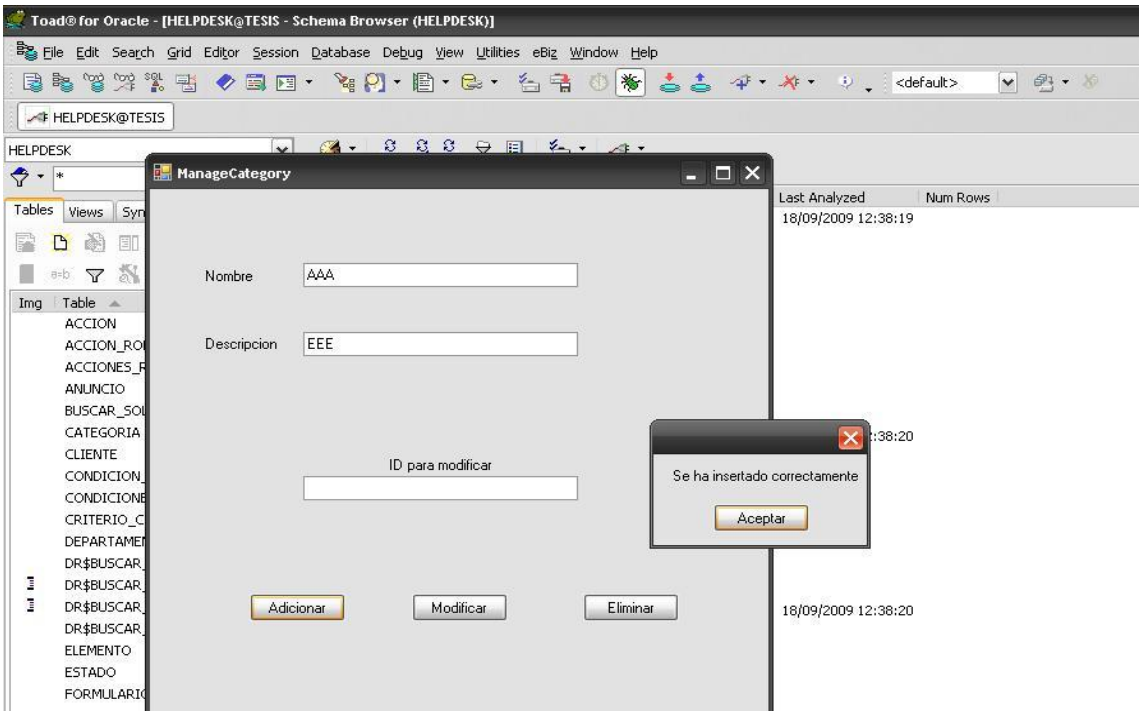


Figura 13: Insertar Categoría

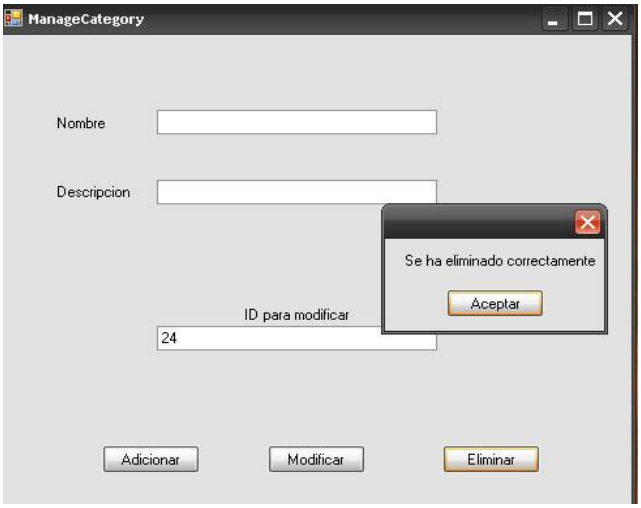


Figura 14: Eliminando Categoría

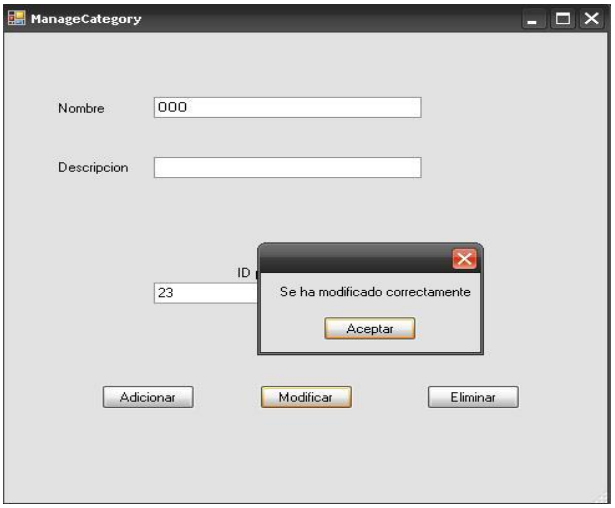


Figura 15: Modificando Categoría

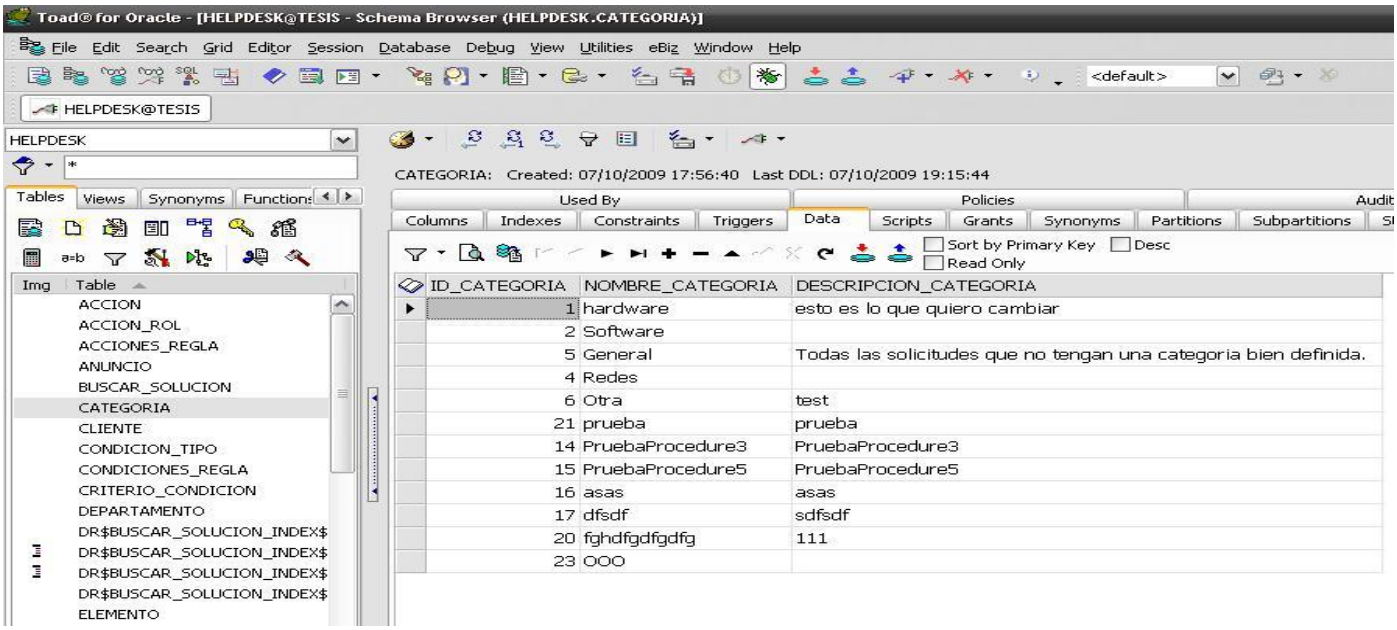


Figura 16: Cambios en la Tabla Categoría

3.11 Conclusiones

En este capítulo se abarcó lo relacionado a la solución escogida la que facilitará la erradicación de algunos problemas que presenta la BD inicial. Se detallan las normas a seguir para el desempeño del código, se explica cómo se realizan las distintas funcionalidades. Además de mostrar la BD luego de haber sido modificada. Se muestra el Modelo de datos donde se relacionan todas las tablas, además de los EDM que exponen la BD mapeada. Se especifican las descripciones de algunas tablas de la BD y se ofrece el modelo de despliegue el cual contiene una PC cliente y el servidor de BD conectados mediante el protocolo TCP/IP. Finalmente se aclara el tipo de prueba a empleado, así como su importancia para un mejor funcionamiento.

CONCLUSIONES

Como conclusiones generales de la investigación se realizó un estudio del estado del arte haciendo un mayor enfoque en las técnicas ORM utilizando el lenguaje C# específicamente, junto a esto se investigaron también los diversos frameworks *de* acceso a datos para facilitar así que la aplicación pueda tener toda la información referente a las interrupciones normalizada y centralizada. A partir de este estudio se muestran alcanzados los objetivos propuestos satisfactoriamente:

- Investigar las tecnologías y herramientas ORM existentes para la selección del apropiado. Para dar cumplimiento a este objetivo se analizaron varias herramientas ORM muy utilizadas, este análisis posibilitó la elección del EF para llevar a cabo el mejoramiento de la capa de acceso a datos del sistema HelpDesk. Al seleccionar esta técnica ORM se logró que la implementación del acceso a datos, utilizando el componente, se realice totalmente orientada a objeto, aprovechando además todas las capacidades para gestionar la BD y erradicar los problemas existentes.
- Elaborar el modelo de objetos. Para elaborar el modelo de objetos se realizó un análisis profundo de las herramientas que permiten diseñarlo, en este caso el ORM al mapear la BD nos permite tener el EDML directo. Para tener un mayor control y seguridad del modelo realizado se llevo a cabo la ingeniería inversa al mismo en el ERwin permitiendo así una mayor organización.
- Implementar Módulos con la técnica ORM. Para dar cumplimiento a este objetivo se partió de la creación de un modelo general que permitió la comprensión y modificación de la BD. Partiendo de ello se logró erradicar las dificultades existentes de funcionalidades el componente.
- Realizar pruebas al Módulo de acceso a datos. Para completar este objetivo se utilizó VStudio 2008, permitiéndonos realizar las pruebas de caja negra, ejercitando los requisitos funcionales desde el exterior del módulo a nivel de funcionalidades, verificando las entradas y salidas de la BD.

Por lo anteriormente descrito, se concluye el cumplimiento del objetivo general de la investigación: Mejorar los módulos existentes con una herramienta ORM que contribuya a perfeccionar la capa de acceso a datos del sistema HelpDesk. Para ello se utilizó el EF herramienta de gran prestigio actual facilitando así que la aplicación pueda tener toda la información en la BD referente a las interrupciones y servicios de forma normalizada y centralizada.

RECOMENDACIONES

Teniendo en cuenta el papel que juega la BD en el desarrollo de una aplicación para la gestión de interrupciones y solicitudes se recomienda que:

- Agregar nuevas funcionalidades y componentes a la capa de acceso a datos de la aplicación HelDesk según surjan necesidades en el MININT.
- Integrar el módulo Reportes a la capa de acceso a datos para una mayor organización
- Integrar el módulo de acceso a datos al sistema para establecer la correspondencia entre estos.

REFERENCIAS BIBLIOGRÁFICAS

Alachisoft. 2010. Framework TierDeveloper para :NET.

Borroto Cintra.J.M. Implementación de la capa de Acceso a Datos de los módulos de Administración y Configuración del Proyecto Convenio Integral de Cooperación Cuba Venezuela.

CEYUSA. 2006. El problema de la diferencia de impedancia. Disponible en: http://www.glib.org.mx/article.php?story=20060611151541892

Codd. 1970.
Dario. 2007.

DOTNET_CLUB. 2010. Entity Framework. Disponible en: http://feedproxy.google.com/~r/DotnetclubUoc/~3/TO62_2uy15g/

EDUARDO. 2003.

Gonzalez_Perez. 2008. Articulo

Gregorio. 2001.

HYDE. J. Base de datos. 2002.
Disponible en: http://www.monografias.com/trabajos11/basda/basda.shtml

Hanson, Jeff. Programación en castellano. Persistencia de Objetos Java utilizando Hibernate. [En línea] 2006. H:\DATA\Docencia\Bibliografia\Programación en castellano_ Persistencia de Objetos Java utilizando Hibernate programación gratis.htm.

LAGO. 2007. Patrones de diseños. Disponible en: http://www.proactiva-calidad.com/java/patrones/index.html

lfranco. 2010. NHibernate. Disponible en: http://ayende.com/blog/archive/2010/01/05/nhibernate-vs.-entity-framework-4.0.aspx

Luque. 2005. Capas de acceso a datos. Articulos.

Losada León Efraín y Cuba García Katiuska "Sistema de Control y Gestión de las Interrupciones y Servicios de las Tecnologías de la Información y las Comunicaciones"

Morrison. 2008.

MSDN. 2005. *[Online] 2005. Introducción a Entity Framework.*
—. 2010. *Características y ventajas.*

QUINTERO. J, 2008. *Directrices para la construcción de artefactos de persistencia en el proceso de desarrollo de software. Colombia: 2008.*
Disponible en: http://revista.eia.edu.co/articulos9/articulo%206.pdf

Regueiro Martínez Leandro.2009. *"Componente de acceso a datos para soluciones de emisión de documentos de identificación"*

Rodriguez. 2007.

SKINNER. ***2002.*** *B. J., DONNY MACK. Professional ADO.Net Programming, 2001.*

SOTOMAYOR, Basilio, Borja. *La plataforma .NET:¿el futuro de la Web? ,Eside,2002:18-21.*

VALDÉS. *D. P. ¿Qué son las bases de datos?*
Disponible en: http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/

BIBLIOGRAFÍA

Alachisoft. 2010. Framework TierDeveloper para: NET.

Ambler1, Scott. The Design of a Robust Persistence Layer for Relational Databases, http://www.ambysoft.com/persistenceLayer.pdf.

Ambler, Scott. Mapping Objects to Relational Databases: O/R Mapping In Detail. http://www.agiledata.org/essays/mappingObjects.html.

Beauchamp, Richard. OJB.NET an object-to-relational persistence tool for the .NET platform: User QuickStart Tutorial. http://ojb-net.sourceforge.net.

Bellware, Scott. Object-Relational Persistence for .NET, http://www.15seconds.com/issue/040112.htm.

Berglas, Anthony. Object Relational Mapping Tools. http://www.uq.net.au/~zzabergl/simpleorm/ORMTools.html.

Borroto Cintra.J.M. Implementación de la capa de Acceso a Datos de los módulos de Administración y Configuración del Proyecto Convenio Integral de Cooperación Cuba Venezuela.

CEYUSA. 2006. El problema de la diferencia de impedancia. Disponible en: http://www.glib.org.mx/article.php?story=20060611151541892

Codd. 1970. Normalización de una Base de Datos. Reglas. Primera forma normal.

Crespo Martín, Cesar. Tutorial de Hibernate, http://www.adictosaltrabajo.com/tutoriales/view.php?tutorial=hibernate.

Dario. 2007.

DOTNET_CLUB. 2010. Entity Framework. Disponible en: http://feedproxy.google.com/~r/DotnetclubUoc/~3/TO62_2uy15g/

EDUARDO. 2003.

Fowler, Martin.Catalog of Patterns of Enterprise Application Architecture. http://martinfowler.com/eaaCatalog/index.html. .

Fussell, Mark. Foundations of Object Relational Mapping. http://www.chimu.com/objectRelational.pdf.

Glögls, Michael. Why we need Hibernate: What an object-relational mapper does. http://www.gloegl.de/17.html.

Gonzalez_Perez. 2008. Artículo

Gregorio. 2001.

Hanson, Jeff. Persistencia de Objetos Java utilizando EJBs, Traducción de Juan Antonio Palos.

Hanson, Jeff. Persistencia de Objetos Java utilizando Hibernate, Traducción de Juan Antonio Palos. http://programacion.com/articulo/jap_persis_hib.

Hanson, Jeff. Programación en castellano. Persistencia de Objetos Java utilizando Hibernate. [En línea] 2006. H:\DATA\Docencia\Bibliografia\Programación en castellano_ Persistencia de Objetos Java utilizando Hibernate programación gratis.htm.

HYDE. J. Base de datos. 2002.
Disponible en: http://www.monografias.com/trabajos11/basda/basda.shtml

Keller, Wolfgang. Persistence Options for Object-Oriented Programs. http://www.objectarchitects.de/PersistenceOptionsOOP2004e.pdf.

Keller, Wolfgang. Object/Relational Access Layers: A Roadmap, Missing Links and More Patterns. http://www.objectarchitects.de/or06_proceedings.pdf.

LAGO. 2007. Patrones de diseños. Disponible en: http://www.proactiva-calidad.com/java/patrones/index.html

Losada León Efraín y Cuba García Katiuska “Sistema de Control y Gestión de las Interrupciones y Servicios de las Tecnologías de la Información y las Comunicaciones”

Marguerie, Fabrice. Choosing an object-relational mapping tool. http://madgeek.com/Articles/ORMapping/EN/mapping.htm, http://weblogs.asp.net/fmarguerie.

Morrison. 2008.

MSDN. 2005. [Online] 2005. Introducción a Entity Framework.

—. 2010. Características y ventajas. Disponible en: http://msdn.microsoft.com/es-es/default.aspx.

Algunas Páginas Consultadas

http://msdn.microsoft.com/es-es/default.aspx.

http://www.asp.net/learn/mvc/.

http://www.devart.com/dotconnect/entityframework.html

http://www.microsoft.com/spanish/msdn/vs2008/sp/sp1.mspx

GLOSARIO DE TÉRMINOS

API: Application Programming Interface (Interfaz de Programación de Aplicaciones), es un conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones.

Framework: es un marco de trabajo que expone un conjunto de librerías que no forman parte de una aplicación específica y que han sido creadas para ser utilizados por cualquier aplicación. Sus componentes pueden ser usados para propósitos generales y están ampliamente probados para funcionar en gran variedad de escenarios.

Manejador de base de datos: representa un conjunto de clases que permite comunicarse con un sistema de BD. En el caso del la API JDBC solo se exponen interfaces para acceder a la BD y los manejadores de BD tiene las clases concreta que implementan estas interfaces para acceder al sistema de BD específico.

XML: Extensible Markup Language (Lenguaje de Marcas Extensible), es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

Hibernate: es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una BD relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Spring: El Spring Framework (también conocido simplemente como Spring) es un framework de integración que se utiliza para el desarrollo de aplicaciones para la plataforma Java.

Persistencia: Es uno de los conceptos más importantes en el desarrollo de aplicaciones. Se refiere a la capacidad de estas de preservar la información de sus datos incluso cuando hayan sido cerradas.