UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

IMPLANTACIÓN DE LA PORTABILIDAD EN UN SISTEMA DE INFORMACIÓN

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:

EMMANUEL ALCANTARA GARCES

ASESOR:
Mtro. LUIS RAMIREZ FLORES

MÉXICO **2009**

Implantación de la portabilidad en un sistema de información.

Para mi esposo:

Espero que estas líneas te den sabiduría para que tu tesis tenga el éxito que deseas.

Desde el día que te conocí supe que eras una persona, emprendedora con deseos de superarte; te deseo que en cada línea, y cada hoja de este libro, tengas un impulso para alcanzar tus sueños, y que estos sueños se hagan realidad; para que tu futuro sea sólido que es lo que tanto deseas, para ti y tu familia.

Amor, cada vez que tengas un problema, piensa que yo estoy a tu lado, en cada momento apoyándote y en cada decisión que tomes. Recuerda tu y yo en las buenas y en las malas.

Te ama:

Tú esposa.

Emmanuel:

Cuando me enteré que vendrías a este mundo la alegría y bienestar que sentí, es algo indescriptible; cuando naciste, tuve la seguridad que serías diferente a todos y no me equivoqué pues lo has demostrado a lo largo de tu formación, tanto moral como académicamente, una parte de ti y otra de la convivencia cotidiana, te han forjado como eres. Dios te ha protegido y te ha dotado de muchas virtudes, las cuales debes bien utilizar, tienes los medios para que seas una gran persona. Esta etapa de tu formación que vas a concluir también debe ser exitosa, pues marcará el inicio de otra más intensa.

Lo que elijas es lo mejor y más acertado pues lo desarrollaras con pasión y le dará más sentido a tu vida. Nunca te olvides de Dios que desde tu concepción siempre ha estado contigo y te ha acompañado durante toda tu vida, siempre debes darle gracias por su benevolencia hacia ti.

Hijo ¡Mucho éxito y felicidad!

Rosa.

Amigo:

Lo importante de desarrollar un proyecto es concluirlo y llegar a la meta depende del esfuerzo que demos día a día y recuerda que es muy fácil hablar y opinar, pero es más importante actuar.

Y el paso que estar por dar te va a ayudar a ser el profesionista por el que un día jugaste a ser.

No te desanimes, los grandes proyectos así inician, lo importante es terminarlos.

Gladis Fuentes.

Para todos:

Para todas las personas que me apoyaron o que tuvieron algo que ver en la realización de este trabajo simplemente me queda decirles muchas gracias de todo corazón, y que su vida esté llena de satisfacciones les deseo:

Emmanuel Alcántara Garcés.

ÍNDICE.

INTRODUCCIÓN. 1

CAPÍTULO 1: ESTUDIO DE LA PORTABILIDAD DEL SISTEMA. 3

INTRODUCCIÓN. 5
ESTUDIO DE LA PORTABILIDAD DEL SISTEMA. 7
Conceptos 7
Planteamiento del problema. 9
Antecedentes. 10
Aplicación. 10
La descarga. 11
La carga. 11
Diagrama de Flujo de Datos (DFD). 12
Proceso. 12
Flujo de Datos. 12
Almacén. 13
Entidad Externa. 13
Niveles de un DFD. 14
Implantación del método en el problema. 14
Diccionario de Datos (DD). 18
Proceso. 18
Flujo de Datos. 19
Almacén. 19
Estructura de Datos. 19
Elemento Dato. 20
Implantación del método en el problema. 20
Árboles de Decisión. 21
Dibujo de los árboles de decisión. 22
Implantación del método en el problema. 23
Tablas de Decisión. 24
Desarrollo de tablas de decisión. 24
Tipos de entradas de tablas. 25
Tablas múltiples. 25
Implantación del método para la solución del problema. 26
Lenguaje Estructurado. 28
Redacción del lenguaje estructurado. 28
Implantación del método en el problema. 29

CAPÍTULO 2: MARCO TEÓRICO. 33

INTRODUCCIÓN. 35
MARCO TEÓRICO. 37
Arquitecturas Computacionales. 37
Arquitectura basada en servidor. 37
Arquitectura basada en cliente. 38
Arquitectura Cliente-Servidor. 39
Arquitectura Cliente-Servidor de 3 capas 39

Bases de datos (BD). 41
Historia. 41
Década de 1960. 41
Década de 1970. 41
Década de 1980. 41
Principios década de los 90. 42
Finales de la década de los 90. 42
Siglo XXI. 42
Definición. 42
Sistema Gestor de Base de Datos. 42
Tipos de base de datos. 43
Base de datos jerárquica. 43
Base de datos en red. 44
Base de datos relacional. 44
Reglas de Codd. 45
Base de datos orientada a objetos. 47
Elementos de una BD. 48
Tipos de datos de una BD. 49
Lenguajes de BD. 49
Lenguaje de Definición de Datos. 50
Lenguaje de Manipulación de Datos. 51
Lenguajes de programación. 52
Niveles de lenguajes de programación. 52
Bajo nivel. 52
Lenguaje Máquina (1er Nivel). 53
Lenguaje Ensamblador (2° Nivel) 53
Alto nivel. 53
3er Nivel. 53
Generadores de aplicaciones (4º Nivel) 54
Clasificación de los lenguajes de programación. 54
En base a su forma de programar. 54
Lenguajes imperativos. 54
Lenguajes declarativos. 54
Lenguajes orientados a objetos. 55
Lenguajes orientados al problema. 55
Lenguajes funcionales. 55
Lenguajes lógicos. 55
Con base a su campo o área de aplicación. 55
Científica. 55
De negocios. 56
Inteligencia Artificial. 56
Programación de sistemas. 56
Lenguajes de Script. 56
En base a su método de implementación. 56
Ensambladores. 56
Compiladores. 57
Intérpretes. 59

Híbridos. 60
Información breve de los lenguajes utilizados. 61
Java. 61
Ventajas. 62
JavaScript. 62
Ventajas. 63
HTML. 63
Ventajas. 63
Desventajas. 63
Características especiales. 63
SQL. 64
Tecnologías Utilizadas. 64
Servlet 64
Ventajas 64
JSP 64
Ventajas 64
JavaBeans 64
CAPÍTULO 3: DISEÑO Y DESARROLLO DE SISTEMAS PORTABLES. 67
INTRODUCCIÓN. 69
DISEÑO Y DESARROLLO DE SISTEMAS PORTABLES. 71
Diseño de Salida. 71
Impresión. 71
Despliegue. 72
Audio. 72
¿Cómo elegir la tecnología de salida? 73
Diseño del formato de salida. 74
Diseño de la salida por impresión. 74
Diseño de salida por despliegue. 75
Implantación del diseño de salida para la solución del problema. 77
Diseño de Entrada. 78
Datos de Entrada. 79
Diseño de las formas. 80
Diseño de pantallas. 80
Validación de la entrada. 80
Verificación de la transacción. 81
Validación de los datos de la entrada. 81
Implantación del diseño de entrada en la solución del problema. 82
Diseño de la Interfaz de usuario. 83
Tipos de interfaces. 84
Interfaz de lenguaje 84
Interfaz de preguntas y respuestas. 84
Menús. 84
Formas de entrada/salida. 84
Interfaz en lenguaje de comandos. 84
Interfaz de manejo directo. 85
El ratón. 85

Otras interfaces del usuario. 85
Retroalimentación para el usuario. 85
Aceptación de la entrada. 85
Entrada correcta. 86
Entrada incorrecta. 86
Retrazo en el proceso. 86
Petición terminada. 86
Petición incompleta. 86
Mayor retroalimentación o ayuda. 86
Implantación del diseño de la Interfaz de usuario en la solución del problema. 87
Diseño de Archivos. 94
Tipos de archivos. 94
Archivo maestro. 94
Archivo de tabla. 95
Archivo de transacciones. 95
Archivo de trabajo. 95
Archivos de impresión. 95
Métodos de organización de Archivos. 95
Organización secuencial. 95
Organización de acceso directo. 96
Lista de enlace. 96
Organización indexada. 96
Organización de índices secuénciales. 97
Implantación del diseño de archivos en la solución del problema. 97
Dispositivos de Almacenamiento. 99
Tecnología magnética. 99
Discos magnéticos. 99
Cintas magnéticas. 99
Tecnología óptica. 99
Disco compacto. 100
Dispositivos de almacenamiento en la solución del problema. 100
Ambientes en línea y distribuidos. 100
Sistemas en línea. 101
Sistemas distribuidos. 101
Implantación en la solución del problema. 101
Desarrollo. 102
Diseño ascendente (bottom-up). 102
Diseño descendente (top-down). 102
Desarrollo modular. 103
Diagrama estructural. 104
Herramientas de documentación. 106
Diagramas HIPO. 106
Diagramas de flujo. 109
Diagramas Nassi-Schneiderman. 109
Diagramas Warnier/Orr. 111
Pseudo código. 111
Método Folklore. 112

Elección de una técnica de documentación. 112
Implantación del desarrollo en la solución del problema. 112
Pruebas. 119
Prueba del programa con datos de prueba. 120
Prueba de enlace con datos de prueba. 120
Prueba del sistema completo con datos de prueba. 120
Prueba del sistema completo con datos de reales. 121
Pruebas especiales de sistemas. 121
Realización de las pruebas en la aplicación desarrollada. 122

CAPÍTULO 4: IMPLANTACIÓN Y SOLUCIÓN. 123

INTRODUCCIÓN. 125
IMPLANTACIÓN Y SOLUCIÓN. 127
Centro de información. 127
Objetivos del centro de información. 127
Preparación del sitio. 127
Adiestramiento de los usuarios. 127
¿A quién adiestrar? 128
Adiestramiento de operadores de sistemas. 128
Adiestramiento de usuarios. 128
Personal que adiestra a los usuarios. 129
Lineamientos para el adiestramiento. 129
Objetivos del adiestramiento. 129
Métodos de adiestramiento. 129
Instalaciones para el adiestramiento. 129
Materiales para el adiestramiento. 130
Conversión. 130
Métodos de conversión. 131
Reemplazo total. 131
Conversión en paralelo. 131
Conversión gradual. 132
Prototipo modular. 132
Conversión distribuida. 132
Plan de conversión. 133
Seguridad. 133
Seguridad física. 133
Seguridad lógica. 134
Conducta del usuario. 134
Otras consideraciones. 134
Evaluación. 134
Enfoque de utilerías. 135
Utilería de posesión. 135
Utilería de forma. 135
Utilería de lugar. 135
Utilería de tiempo. 136
Utilería de actualización. 136
Utilería de objetivo. 136

Evaluación del sistema. 136
Implantación de la Aplicación desarrollada. 136
CONCLUSIONES. 139
BIBLIOGRAFÍA. 143

Introducción.

Este trabajo se hizo con el objetivo de mostrar las fases y etapas de cada una de ellas para lograr la implantación de la portabilidad en un sistema de información.

La portabilidad es requerida debido a que desde hace algunos años lo más importante para una empresa, no importando cual se el ramo al que se dedique, ha sido su información. Cuando una empresa requiere cambiar la forma en que administra su información o el sistema que la administra, la parte de mayor importancia en ese cambio es lograr llevar su información con toda la integridad y sin ninguna perdida de datos hacia la nueva forma de administración o hacia el nuevo sistema que la administrara.

El sistema al cual se le implantó la portabilidad, era un sistema que en sus versiones iniciales no tenia esta característica, debido a la forma en que se utilizaba; es decir, no requería tener la portabilidad ya que la información de todos sus cliente la tenia concentrada en una sola base de datos.

Con el paso del tiempo y el uso del sistema, los clientes del mismo se vieron en la necesidad de administrar ellos mismos su información, con lo cual solicitaron que se les entregara. La información por si sola, no les servia de mucho debido a que no se interpretaba tan fácilmente. Por la anterior, el dueño de la empresa que desarrollo el sistema de información al cual se le implantó la portabilidad, les ofreció a los clientes la venta del sistema, con lo cual ellos podían implementarlo en las condiciones y con el hardware que más convinieran a sus intereses. Cuando algunos de los clientes decidieron comprar el sistema de información surgió la necesidad de implantarle la portabilidad, con lo cual se justificó el tiempo invertido y el esfuerzo realizado para lograrlo.

Este trabajo se divide en cuatro capítulos: Estudio de la Portabilidad del Sistema, Marco Teórico, Diseño y Desarrollo de Sistemas Portables e Implantación y Solución.

En el capítulo Estudio de la Portabilidad del Sistema se explican los conceptos que se emplearon para realizar el estudio de la portabilidad. También se describe más detalladamente el problema que se resolvió, y da una reseña de los antecedentes del sistema, para mejor entendimiento de la solución, al que se le implantó la portabilidad

El capítulo Marco Teórico, se enfoca en mostrar la base sustentable de conocimientos teóricos para lograr el desarrollo de la portabilidad

En el capítulo Diseño y Desarrollo de Sistemas Portables se hablan de las múltiples opciones que se tienen sobre los métodos de diseño y desarrollo de una aplicación y de cómo se realizó la selección de cada una de ellas para implantarla en la solución del problema.

El capítulo Implantación y Solución menciona las diferentes formas que existen para implantar una solución y de las actividades que se deben hacer antes y después de la implantación. También se menciona cual fue la forma que se utilizó para implantar la portabilidad y las actividades correspondientes a dicha implantación.

Capítulo 1: Estudio de la portabilidad del sistema.

Introducción.

En este capitulo se explican los conceptos que se emplean para soportar la decisión que se tomó para definir la estructura del contenido del mismo capitulo.

Se describe el planteamiento del problema, que es objeto de este trabajo, los antecedentes (del sistema que requirió la solución) necesarios para tener una mejor comprensión del problema, y las características que requirió la solución solicitada.

También se describe como se deben emplear las herramientas Diagrama de Flujo de Datos (DFD), Diccionario de Datos (DD), Árboles de Decisión, Tablas de Decisión y Lenguaje Estructurado. Posterior a la descripción de cada herramienta se muestra como fue aplicada en el problema que se resolvió y el resultado que se obtuvo de cada una de ellas.

En la parte que corresponde al Diagrama de Flujo de Datos se explican los elementos que lo integran, los cuales son: proceso, flujos de datos, almacén y entidad externa, así como las múltiples simbologías con que pueden ser representados cada uno de ellos, y los niveles que puede alcanzar un Diagrama.

Con el Diccionario de Datos (DD) también se explican los elementos que lo integran. Estos elementos son: proceso, flujo de datos, almacén, estructura de datos y elemento dato.

En los Árboles de Decisión se explica como se deben dibujar, como se deben de interpretar y los elementos que tiene (condiciones, opciones y acciones).

En lo que comprende a las Tablas de Decisión, la explicación hace referencia a la forma en como se desarrollan, indicando los pasos a seguir (determinar el número de condiciones, determinar el número de acciones, contemplar las combinaciones de las condiciones, llenar las opciones de las condiciones, llenar las entradas, eliminar redundancia y eliminar contradicciones), los tipos de entradas que existen para una tabla (limitada, extendida y mixta) y el uso de las tablas múltiples.

Por último en el Lenguaje Estructurado se explica como se debe hacer la redacción, usando los verbos adecuados y las frases correctas, así como las tres estructuras (secuencia, decisión e iteración) necesarias para describir un proceso.

Estudio de la portabilidad del sistema.

"*El punto clave del análisis de sistemas se consigue al adquirir un conocimiento detallado de todas las facetas dentro del área de negocios que se investiga*" [1]

"*El objetivo de la fase de análisis es el establecimiento de los requerimientos para el sistema que es desarrollado*" [2]

Conceptos

Para llevar a cabo la actividad de "Determinación de Requerimientos" (del ciclo de vida de un sistema de información), cada autor propone y utiliza estrategias y métodos diferentes, para obtener los requerimientos de un sistema a desarrollarse, de acuerdo a sus convicciones: por ejemplo:

- El libro "Análisis y Diseño de Sistemas de Información", del autor James A. Senn, propone utilizar dos estrategias, las cuales se llaman: "Estrategia de Flujo de Datos" y "Estrategia de Análisis de Decisiones". La primera estrategia maneja dos métodos: Diagramas de Flujo de Datos, y Diccionario de Datos. La segunda estrategia maneja tres métodos: Árboles de Decisión, Tablas de Decisión, e Inglés Estructurado.
- El libro "The Basic of System Analysis and Design", de Jennifer Rowley propone utilizar dos estrategias, las cuales son: "Técnicas de Descripción de Sistemas" y "Modelación Lógica". La primera estrategia maneja seis métodos: Graficas Organizacionales, Graficas de Actividades, Graficas de Flujo, Tablas de Decisión, Árboles de Decisión, e Inglés Estructurado. La segunda estrategia maneja tres métodos: Diagramas de Flujo de Datos, Diccionario de Datos, e Historia de Vida de Entradas.
- En la bibliografía de Kenneth E. Kendall y Julie E. Kendall, "Análisis y Diseño de Sistemas", se proponen dos estrategias: "Análisis de Flujo de Datos", la cual utiliza los métodos de Diagramas de Flujo de Datos, y el Diccionario de Datos; y "Análisis de la Lógica de Decisiones", que emplea los métodos de Lenguaje Estructurado, Tablas de Decisiones, y Árboles de Decisiones.
- El autor George M. Marakas en su libro "System Analysis and Design" menciona los métodos de Diagrama de Flujo de Datos, Inglés Estructurado, Tablas de Decisión, Árboles de Decisión, y Diagramas de Transición de Estado para obtener lo requerimientos de un sistema.

[1] JAMES A. Senn. "Análisis y Diseño de Sistemas de Información."
[2] ROWLLEY Jennifer. "The Basic of Systems Analysis and Design."

Como se puede leer cada autor utiliza diferentes estrategias para realizar la tarea correspondiente a la Determinación de Requerimientos. Sin embargo existen varios métodos en común que son empleados por todos, o si no por la mayoría, de los autores arriba mencionados; estos métodos son:

- Diagramas de Flujo de Datos (DFD).
- Diccionario de Datos (DD).
- Árboles de Decisión.
- Tablas de Decisión.
- Lenguaje Estructurado.

Los métodos que se enlistaron, serán los métodos que se mencionaran en este capítulo, y los cuales fueron utilizados para cubrir la actividad de "Determinación de Requerimientos", del desarrollo de la aplicación. Dicha aplicación es el objeto sobre el cual se desarrolla este trabajo de tesis, y que resolvió el problema que se plantea mas adelante.

Cuando se mencione cada uno de los métodos enlistados, este será acompañado de una explicación que indica la forma como se debe utilizar. Al terminar la explicación se mostrará la implantación que se realizó del método con respecto a nuestro problema.

Planteamiento del problema.

Una empresa requirió una aplicación que obtuviera los datos de una base de datos, estos datos fueron ingresados, actualizados y almacenados en la base de datos por un software empaquetado (desarrollado por la propia empresa). De igual manera, la empresa también requirió una aplicación que ingresara los datos que fueron obtenidos, por la aplicación anterior, a una nueva base de datos para que pudieran ser utilizados y actualizados por el software desarrollado por ella. El software que requirió las aplicaciones de carga y descarga de datos, es un software que se encarga de controlar flujos de procesos (como función principal), también puede administrar los clientes, generar reportes de la información que maneja. Los flujos de procesos se configuran e implantan de acuerdo al ámbito en que se desenvuelve la empresa que los emplea, pudiendo ser este: manejo de información financiera, ventas, manufactura, etc. Debido a la amplia variedad de ámbitos en que se puede utilizar este software, y a que no existe una homogeneidad en la información que puede manejar, se necesitó desarrollar una aplicación que resolvió el problema de portabilidad de datos, ya que algunas de las empresas que lo utilizan realizan cambios de equipos de computo (hardware) y/o actualizaciones de los programas que tienen (software).

La Figura 1-1 muestra el problema que se plantea de una forma gráfica.

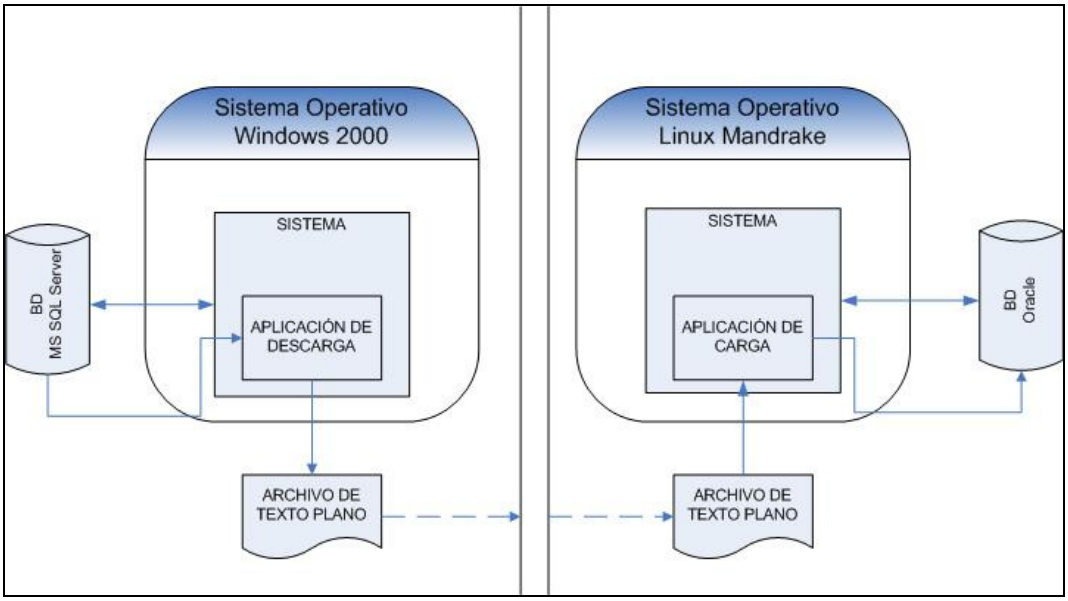


Figura 1- 1 Problema planteado.

Antecedentes.

El software de la empresa, para la cual se requirió el desarrollo de la aplicación que realice la carga y descarga de datos de la base de datos, es un software de Web en el modelo de tres capas (Base de datos-Servidor-Cliente). La primera capa (Base de Datos), es la que se encarga de realizar el manejo de datos respecto a la Base de Datos, la información que entrega esta capa es empleada por la segunda capa. La segunda capa (Servidor), es la que se encarga de realizar toda la lógica de negocios o lógica del programa, esto lo hace a petición de la tercera capa y sirve de enlace entre la primera y la tercera capa. La tercera capa (Cliente), es la interfaz de usuario y son las pantallas que pueden ser visualizadas en cualquier navegador Web; dentro de las pantallas se captura, modifica o borra la información que maneja este software, y es el único medio de comunicación con el usuario.

Otras características que tiene el software, para el que se desarrolló la aplicación, son: la funcionalidad de trabajar bajo cualquier sistema operativo que contenga una Máquina Virtual de Java (JVM, por sus siglas en inglés, Java Virtual Machine), y la posibilidad de funcionar con cualquier Base de Datos que soporte una Conexión Java a Base de Datos (JDBC, por sus siglas en inglés, Java Data Base Connection). En el siguiente capítulo se desarrollarán estos conceptos. También existieron otros requerimientos funcionales:

- **Multiempresa.** El sistema, cuando fue desarrollado se hizo bajo un esquema tipo Web, por lo que el sistema es multiempresa; esto es, el sistema puede funcionar para varias empresas al mismo tiempo.
- **Niveles de usuario.** El sistema cuenta con tres niveles de usuario, “Administrador General”, “Administrador de Organización”, y “Usuario”. Cada nivel tiene un alcance diferente, y puede realizar actividades diferentes a los otros niveles.
- **Seguridad.** El sistema implanta su seguridad con base a una clave y una contraseña. Dicha clave y contraseña dan acceso al sistema y tienen predeterminado el nivel permitido del usuario para trabajar.

Aplicación.

Con base en lo que se menciona en la sección anterior “Antecedentes”, fue necesario cubrir ciertos requerimientos cuando se desarrolló la Aplicación, para poder interactuar con el sistema que la utiliza. Esos otros requerimientos que se cubrieron fueron obtenidos verbalmente del dueño del software, y se mencionan a continuación:

Existen dos tipos: los generales, que son requerimientos que se cumplen en toda la Aplicación, y los particulares, que son los requerimientos que se cumplen de acuerdo a la acción ejecutada. La lista de los requerimientos generales de la Aplicación es la siguiente:

- **Desarrollo tipo Web de tres capas.** Esto se solicitó para poder lograr la integración con el sistema que la utiliza, de una forma natural
- **Multi-Sistemas Operativos.** Facilidad de funcionar bajo diferentes sistemas operativos que tengan una JVM.
- **Multi-Base de Datos.** Posibilidad de trabajar con diferentes Bases de Datos. Éste, al igual que el requerimiento anterior, son características del software que la utiliza.
- **Ejecución desde un cliente del sistema.** La aplicación solo puede ejecutarse por un usuario que se encuentre firmado en un cliente del sistema.
- **Ejecución dentro de la seguridad del sistema.** La aplicación solo se ejecuta por el usuario de nivel "Administrador General".

Los requerimientos particulares de la Aplicación se mencionan en la acción correspondiente.

La descarga.

La parte de la Aplicación que realiza la acción de descarga tiene varios requerimientos particulares. La lista se presenta a continuación:

- **Posibilidad de selección sobre la información requerida.** Por la característica multiempresa del software, en la aplicación se puede seleccionar la información que se desea obtener del sistema.
- **Generación de Archivos con la información.** La información que se obtiene se guarda en archivos.
- **Compresión de archivos.** Debido a la característica Web de la aplicación, la información se comprime para una mejor manipulación de la misma.
- **Entrega de la información al usuario.** Después de obtener y procesar la información del sistema, se entrega al usuario que ejecuta la aplicación.
- **Interpretación de los archivos por otras aplicaciones.** Los archivos obtenidos pueden ser interpretados por otras aplicaciones que no pertenezcan al software.

La carga.

Al igual que en la descarga, la parte de la Aplicación que realiza la carga tiene varios requerimientos particulares, éstos son los siguientes:

- **Protección a la Base de Datos.** Se tiene un control para poder introducir la información a la Base de Datos.
- **Congruencia en la información a cargar.** Se verifica que la información a introducir en la Base de Datos este completa.

Como se puede notar, algunos de los requerimientos mencionados están predefinidos por el sistema con el cual interactúa la Aplicación; con base en esto, ciertos requerimientos son cubiertos por el mismo sistema. Esto se explicará en el tercer capítulo de este trabajo de tesis "Diseño y Desarrollo de la Aplicación".

Diagrama de Flujo de Datos (DFD).

El Diagrama de Flujo de Datos (DFD) es una herramienta gráfica que permite representar la secuencia de procesos y funciones de un sistema, así como los movimientos de flujo de datos a través de ese sistema. Con el DFD se puede observar la lógica que tendrá el sistema. El DFD presenta la información de lo general a lo particular.

Un DFD se dibuja utilizando únicamente cuatro conceptos básicos que son: Proceso, Flujo de Datos, Almacén, y Entidad Externa.

Proceso.

Un proceso es la parte del sistema donde hay un cambio o transformación de los datos. Con base en lo anterior, el nombre de los datos que entran a un proceso siempre es diferente al nombre de los datos que salen.

Un proceso no identifica el componente físico que lo realiza, que puede ser una maquina o una persona.

Un proceso se identifica por medio de un nombre y un número, donde el nombre indica lo que realiza el proceso y se compone de un verbo y un objeto; por ejemplo, "Tomar Manzana", "Calcular Impuestos", etc. El número asignado a un proceso no representa la secuencia del mismo, es solamente para su identificación. Un proceso puede dividirse en varios subprocesos, cuando esto sucede, a el número del proceso se le agrega otro número separado por un punto, donde el nuevo número indica un subproceso; por ejemplo, si un proceso se identifica con el número 3 y se subdivide en dos subprocesos, los nuevos números que identificarán a los subprocesos serán 3.1 y 3.2.

Flujo de Datos.

Un Flujo de Datos representa el movimiento de datos dentro del sistema e indica la dirección que siguen desde su origen hasta su destino, no importando la forma que estos tengan; por ejemplo, una llamada telefónica, una orden de compra de un cliente, etc.

Un Flujo de Datos se identifica con un nombre que defina los datos que mueve; por ejemplo, "Tarifas Validas", "Datos de Compra", etc.

Almacén.

Un Almacén representa un repositorio para guardar datos, ya sea temporal o permanente.

Un Almacén no específica el componente físico que representa, por lo que puede ser electrónico (disco duro, disco óptico, etc.) o no (archivero metálico, estante, etc.)

Un Almacén se identifica con un nombre que indique que es lo que almacena.

Entidad Externa.

Una Entidad Externa es cualquier cosa que interactúa con el sistema, ya sea recibiendo o enviando información, pero que esta fuera de los limites y del control del mismo.

Una Entidad Externa puede ser un cliente, o una organización, etc.

Una Entidad Externa se identifica con un nombre representativo de si misma.

Las simbologías que se utilizan para representar los cuatro conceptos básicos de un DFD varían de acuerdo al enfoque que se utilice. Los dos enfoques más comunes son el de Yourdon-DeMarco y el de Gane-Sarson. Las simbologías de ambos enfoques se muestran en la Tabla 1-1.

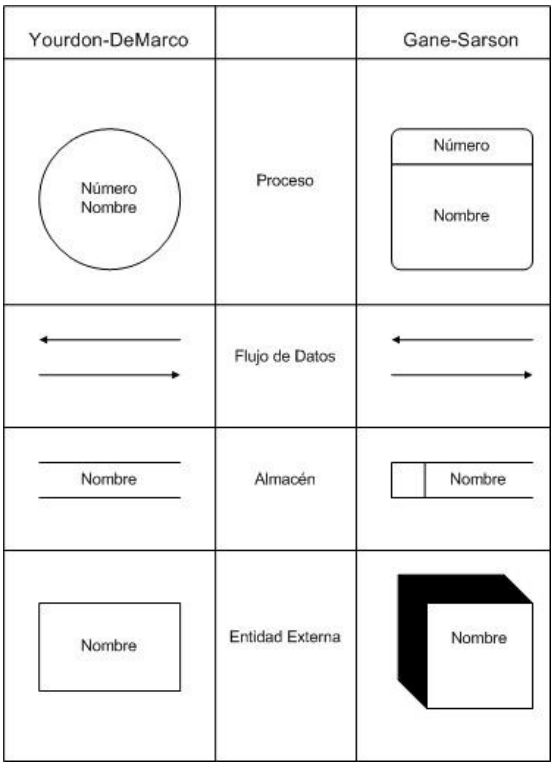


Tabla 1- 1 Comparación de Simbologías para DFD.

Niveles de un DFD.

Un DFD debe usar una estructura jerárquica para representar diversos niveles de detalle.

Los niveles de un DFD señalan el grado de detalle de la representación de un sistema. Estos niveles pueden aumentar y por consiguiente los niveles de detalle de la representación del sistema en el DFD también lo harán.

Existen tantos niveles de un DFD como se necesiten para obtener el detalle necesario del sistema que representa, siendo obligatorios los dos primeros. Una lista de niveles de un DFD puede ser:

El diagrama de contexto. (Obligatorio).
El diagrama de nivel 0. (Obligatorio).
El diagrama de nivel 1.
El diagrama de nivel 2.
.....
El diagrama de nivel n.

En el diagrama de contexto se muestran los límites del sistema, así como las fuentes o destinos (Entidades Externas) de la información. En este diagrama existe un solo proceso etiquetado con el nombre del sistema, y sus respectivos flujos de datos también etiquetados.

En el diagrama de nivel 0 se muestran los procesos más importantes del sistema, los Almacenes, y las Entidades Externas con las que interactúan. Se resalta la transformación de los datos.

En el diagrama de nivel 1 y hasta el diagrama de nivel n se realiza una descomposición de los procesos del diagrama de nivel 0 para obtener una descripción más detallada. Los flujos de entradas y salidas de datos de los procesos, deben coincidir con los flujos del proceso que se esta descomponiendo.

Implantación del método en el problema.

A continuación se muestra como se utilizó lo descrito en el método “Diagramas de Flujo de Datos” para analizar el problema que es el motivo este trabajo.

Se utilizó la simbología del enfoque Gane-Sarson para desarrollar los DFD que se necesitaron. Empezamos por el diagrama de contexto, seguido del diagrama de nivel 0, y después los diagramas de nivel “n” que se consideraron necesarios.

El diagrama de contexto muestra que las única Entidad Externa con las que la Aplicación interactúa es el Usuario del sistema. Ver Figura 1-2.

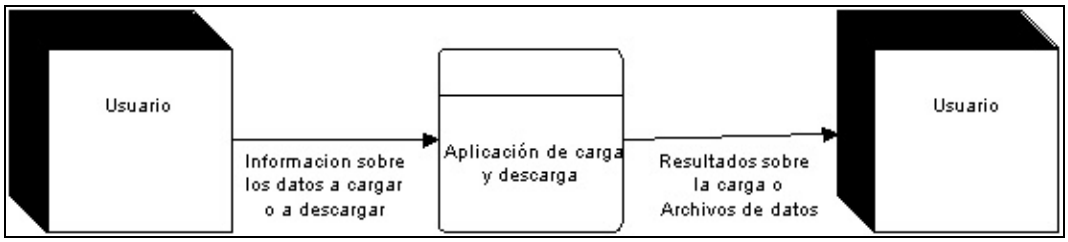


Figura 1- 2 Diagrama de Contexto.

El diagrama de nivel 0, además de las Entidades Externas, presenta todos los Almacenes que interactúan con la Aplicación. El Almacén P1-"Archivos de Datos", interactúa únicamente con el Proceso 1 "Carga". El Almacén P2-"Archivos Temporales", interactúa exclusivamente con el Proceso 2 "Descarga". Los Almacenes C1-"Archivo de Información de la B. D." y C2-"Base de Datos", interactúan con los Procesos 1 y 2. Ver Figura 1-3.

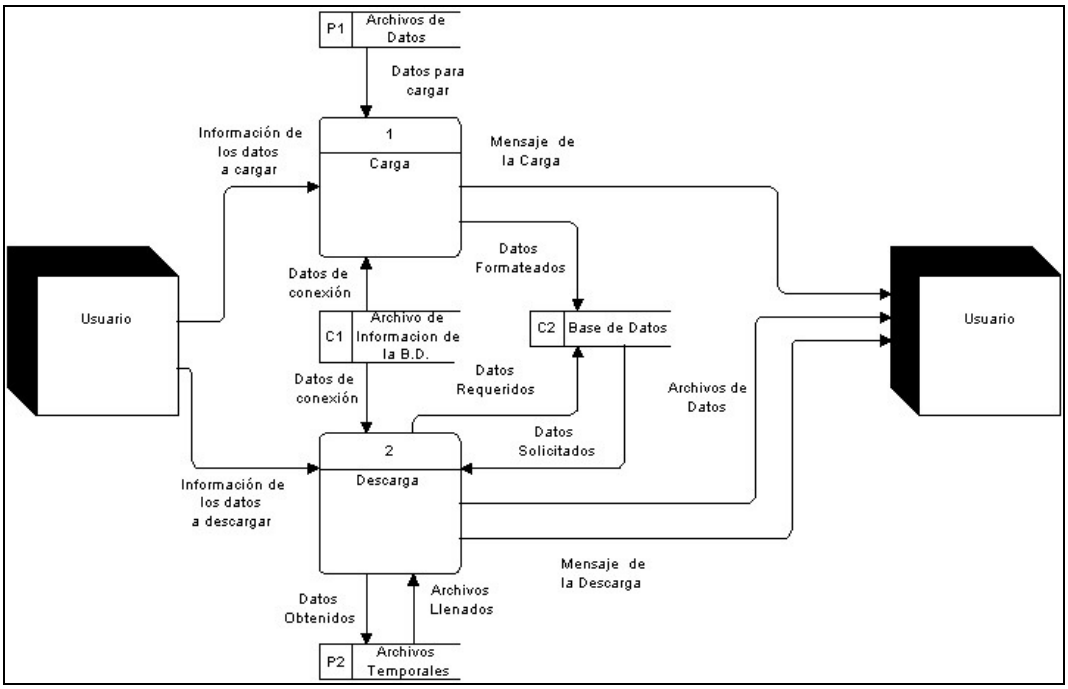


Figura 1- 3 Diagrama de nivel 0.

El diagrama de nivel 1 muestra la descomposición de los 2 procesos principales en subprocesos parciales. Para el Proceso 1 se generaron tres subprocesos: “Leer Archivos de Datos”, “Escribir Base de Datos”, y “Generar Mensaje de Carga”. Para el Proceso 2 se generaron cuatro subprocesos: “Leer Base de Datos”, “Escribir Archivos Temporales”, “Enviar Información”, y “Generar Mensaje de Descarga”. Ver Figura 1-4.

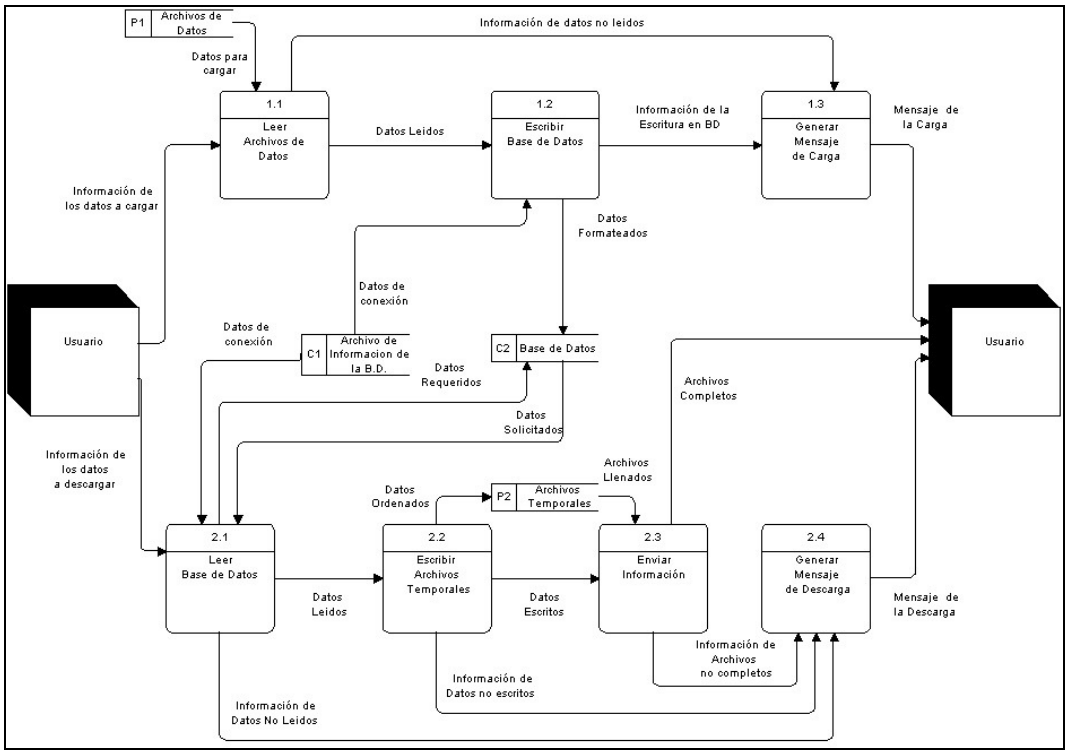


Figura 1- 4 Diagrama de nivel 1.

El diagrama de nivel 2 muestra la descomposición de los 7 procesos parciales de la Figura 1-4 en subprocesos específicos. Para el subproceso 1.1 se generaron tres subprocesos específicos: “Verificar Existencia de Archivos de Datos”, “Verificar Permiso de Lectura de Archivos de Datos”, “Leer Archivos de Datos”. Para el subproceso 1.2 se generaron seis subprocesos específicos: “Obtener Información de la Base de Datos”, “Realizar Conexión a Base de Datos”, “Verificar Existencia de Dato”, “Determinar Tipo de Dato”, “Prepara Formato de Dato”, “Escribir en Base de Datos”. Para el subproceso 1.3 se generaron dos subprocesos específicos: “Preparar Mensaje de Carga”, “Enviar Mensaje de Carga”. Para el Proceso 2.1 se generaron tres subprocesos: “Obtener Información de la Base de Datos”, “Realizar Conexión a Base de Datos”, “Leer Base de Datos”. Para el Proceso 2.2 se generaron tres subprocesos: “Crear Archivos Temporales”, “Verificar Permiso de Escritura de Archivos Temporales”, “Escribir Datos en Archivos Temporales”. Para el Proceso 2.3 se generaron cuatro subprocesos: “Verificar Permiso de Lectura de Archivos Temporales”, “Leer Archivos Temporales”, “Realizar Compresión de Archivos Temporales”, “Enviar Archivos de Datos”. Para el Proceso 2.4 se generaron dos subprocesos: “Preparar Mensaje de Descarga”, “Enviar Mensaje de Descarga”. Ver Figura 1-5.

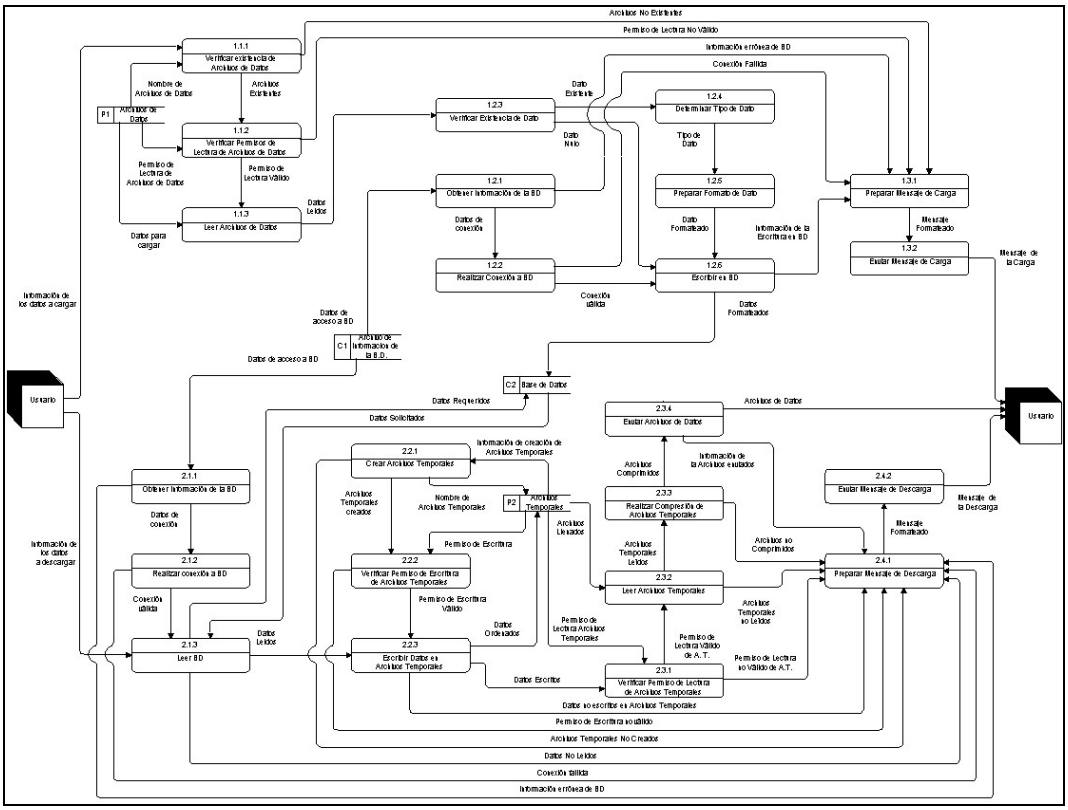


Figura 1- 5 Diagrama de nivel 2.

Diccionario de Datos (DD).

Después de leer varias definiciones de lo que es un Diccionario de Datos (DD) se puede decir que "Es un listado que menciona todos y cada uno de los elementos que aparecen en un DFD que describe un sistema". El DD almacena detalles y descripciones de los elementos que menciona. Cada elemento del DD debe contener información referente a las siguientes categorías:

- **Nombres y sinónimos.** Nombre significativo que identifica a este elemento a través de todo el desarrollo del sistema. También se mencionan los posibles sinónimos con los que se puede encontrar este elemento.
- **Descripción.** Descripción breve, concisa, e informativa del elemento. Debe poder ser interpretada por cualquier persona que la consulte.
- **Rango.** Valores permitidos que pueden ser asignados a este elemento.
- **Longitud.** Número de espacios requeridos para este elemento. Los espacios pueden ser ocupados por letras, números o símbolos.
- **Notación.** Representación del elemento y sus componentes a través de símbolos. Permite su fácil entendimiento. Los símbolos que se pueden utilizar para la notación se muestran a continuación (Tabla 1-2):

Símbolo	Significado
=	Es equivalente a
+	Y (conjunción)
[]	Alternativas
()	Optativo
{}	Iteración
*	Comentario
|	Separador de alternativas

Tabla 1- 2 Símbolos utilizados en un DD.

Los elementos que deben aparecer definidos en un DD son:

Proceso.

Describe el proceso que se encuentran en un DFD. La definición de un proceso debe contener Nombre, Descripción, Entradas, Salidas, y un Resumen Lógico. Ver Tabla 1-3.

PROCESO:
DESCRIPCION:
ENTRADAS:
SALIDAS:
RESUMEN LOGICO:

Tabla 1- 3 Forma de entrada para un DD de un proceso.

Flujo de Datos.

Describe el flujo de datos que aparecen en un DFD. La definición de un Flujo de Datos debe contener Nombre, Descripción, Fuente, Destino, y Estructuras de Datos que viajan en él. Ver Tabla 1-4.

FLUJO DE DATOS:
DESCRIPCION:
FUENTE:
DESTINO:
ESTRUCTURAS DE DATOS:

Tabla 1- 4 Forma de entrada para un DD de un flujo de datos.

Almacén.

Describe el almacén que se muestran en un DFD. La definición de un almacén debe contener Nombre, Descripción, Flujo de Datos Entrantes, Flujo de Datos Salientes, y Contenido. Ver Tabla 1-5.

ALMACEN:
DESCRIPCION:
FLUJOS DE DATOS ENTRANTES:
FLUJOS DE DATOS SALIENTES:
CONTENIDO:

Tabla 1- 5 Forma de entrada para un DD de un almacén.

Estructura de Datos.

Describe el grupo de elementos dato que se caracterizan por tener interrelación entre sí. La definición de una Estructura de Datos debe contener Nombre, Descripción, Contenido, Flujos de Datos Relacionados, y Volumen. Ver Tabla 1-6.

ESTRUCTURA DE DATOS:
DESCRIPCION:
CONTENIDO:
FLUJO DE DATOS:
VOLUMEN:

Tabla 1- 6 Forma de entrada para un DD de una estructura de datos.

Elemento Dato.

Describe el elemento mínimo de un sistema. La definición de un Elemento Dato debe contener Nombre, Descripción, Sinónimo, Tipo, Longitud, Rango, y Valores Específicos. Ver Tabla 1-7.

ELEMENTO DATO:
DESCRIPCION:
SINONIMO:
TIPO:
LONGITUD:
RANGO:
VALORES ESPECIFICOS:

Tabla 1- 7 Forma de entrada para un DD de un elemento dato.

Implantación del método en el problema.

A continuación se presenta un ejemplo de cada entrada para un DD con información correspondiente al problema que se analizó; esto debido a la gran cantidad de entradas que surgieron del DFD de la Figura 1-5. Si se requiere ver el total de entradas, consultar el **Apéndice A**.

La entrada para el proceso 1.1.1 "Verificar existencia de Archivos de Datos" se representó de la siguiente manera (Tabla 1-8):

PROCESO: *Verificar existencia de Archivos de Datos*
DESCRIPCION: *Verifica que los Archivos de Datos necesarios existan*
ENTRADAS: *Información de los datos a cargar, Nombre de Archivos de Datos*
SALIDAS: *Archivos Existentes, Archivos No Existentes*
RESUMEN LOGICO: *Con la Información de los datos a cargar crea una ruta y lee los archivos contenidos en la carpeta, regresando los nombres de estos. Busca dentro de Nombre de Archivos de Datos que existan todos los archivos necesarios para realizar la carga. Si todos los archivos existen genera Archivos Existentes, en caso contrario genera Archivos No Existentes*

Tabla 1- 8 Forma de entrada para un DD del proceso 1.1.1

"Verificar existencia de Archivos de Datos" de la Figura 1-5.

La entrada para el Flujo de Datos "Información de datos a cargar" quedó de la manera que a continuación se muestra (Tabla 1-9):

FLUJO DE DATOS: *Información de datos a cargar*
DESCRIPCION: *Ruta donde se encuentran los Archivos de Datos a cargar para verificar su Existencia*
FUENTE: *Usuario*
DESTINO: *Verificar existencia de Archivos de Datos*
ESTRUCTURAS DE DATOS: *Ruta de los Archivos de Datos*

Tabla 1- 9 Forma de entrada para un DD del Flujo de Datos "Información de datos a cargar" de la Figura 1-5.

La entrada para el Almacén "Archivos de Datos" quedó como sigue (Tabla 1-10):

ALMACEN: *Archivos de Datos*
DESCRIPCION: *Archivos de los datos que serán Cargados*
FLUJOS DE DATOS ENTRANTES:
FLUJOS DE DATOS SALIENTES: *Nombre de Archivos de Datos, Permiso de Lectura de Archivos de Datos, Datos para cargar*
CONTENIDO: *Archivos que contienen los datos que se cargarán*

Tabla 1- 10 Forma de entrada para un DD del Almacén P1

"Archivos de Datos" de la Figura 1-5.

La entrada para la Estructura de Datos "Ruta de los Archivos de Datos" se presenta a continuación (Tabla 1-11):

ESTRUCTURA DE DATOS: *Ruta de los Archivos de Datos*
DESCRIPCION: *Ubicación de la carpeta que contiene los archivos de datos que serán cargados*
CONTENIDO: *Ruta*
FLUJO DE DATOS: *Información de los datos a cargar*
VOLUMEN: *Una vez por evento*

Tabla 1- 11 Forma de entrada para un DD de la Estructura

de Datos "Ruta de los Archivos de Datos".

La entrada para el Elemento Dato "Ruta" se muestra a continuación (Tabla 1-12):

ELEMENTO DATO: *Ruta*
DESCRIPCION: *Ubicación de la carpeta que contiene los archivos de datos*
SINONIMO: *Path, strRuta*
TIPO: *Alfanumérico*
LONGITUD: *Variable*
RANGO:
VALORES ESPECIFICOS:

Tabla 1- 12 Forma de entrada para un DD del elemento dato "Ruta".

Árboles de Decisión.

Los árboles de decisión son una manera de mostrar la acción que puede resultar de una combinación de diferentes condiciones en forma secuencial. Como su nombre lo indica, un árbol de decisión (al terminar de dibujarlo) es muy similar a un árbol de la naturaleza ya que se compone de ramas (condiciones) y hojas (acciones).

Ya que un árbol de decisión es secuencial, muestra que condiciones se deben considerar inicialmente, y cuales después para poder llegar a una acción. Otra de las características de un árbol de decisión es que muestra la relación de condición y sus acciones permitidas.

Un árbol de decisión se debe leer de izquierda a derecha, iniciando por la raíz (elemento más a la izquierda) y tomando la decisión de una rama en particular. Se sigue esta rama hasta que se encuentra un nodo (múltiples decisiones), en este punto se debe tomar la decisión determinada y continuar por la rama indicada. Este proceso se repite hasta llegar a la parte final de la rama, una hoja, la cual nos indicará la acción a realizar.

Dibujo de los árboles de decisión.

Cuando se dibuja un árbol de decisión es necesario distinguir entre las condiciones y las acciones, para ello es recomendado utilizar un círculo para indicar una condición, y un cuadrado para indicar una acción. Las condiciones no se expresan, sino las opciones de la condición sobre las ramas correspondientes. El uso de números en las condiciones y acciones facilita su identificación. Ver Figura 1-6.

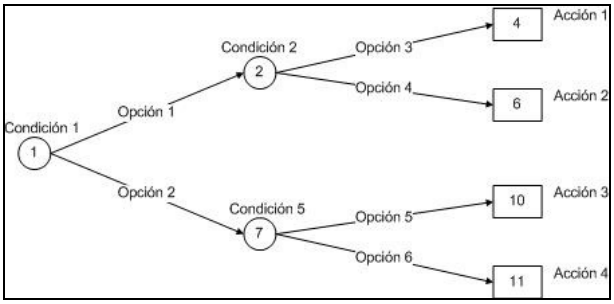


Figura 1- 6 Forma de un Árbol de Decisión.

Implantación del método en el problema.

La implantación del árbol de decisiones al problema de estudio dio como resultado un árbol con 17 nodos (condiciones) y 18 hojas (acciones). Ver Figura 1-7.

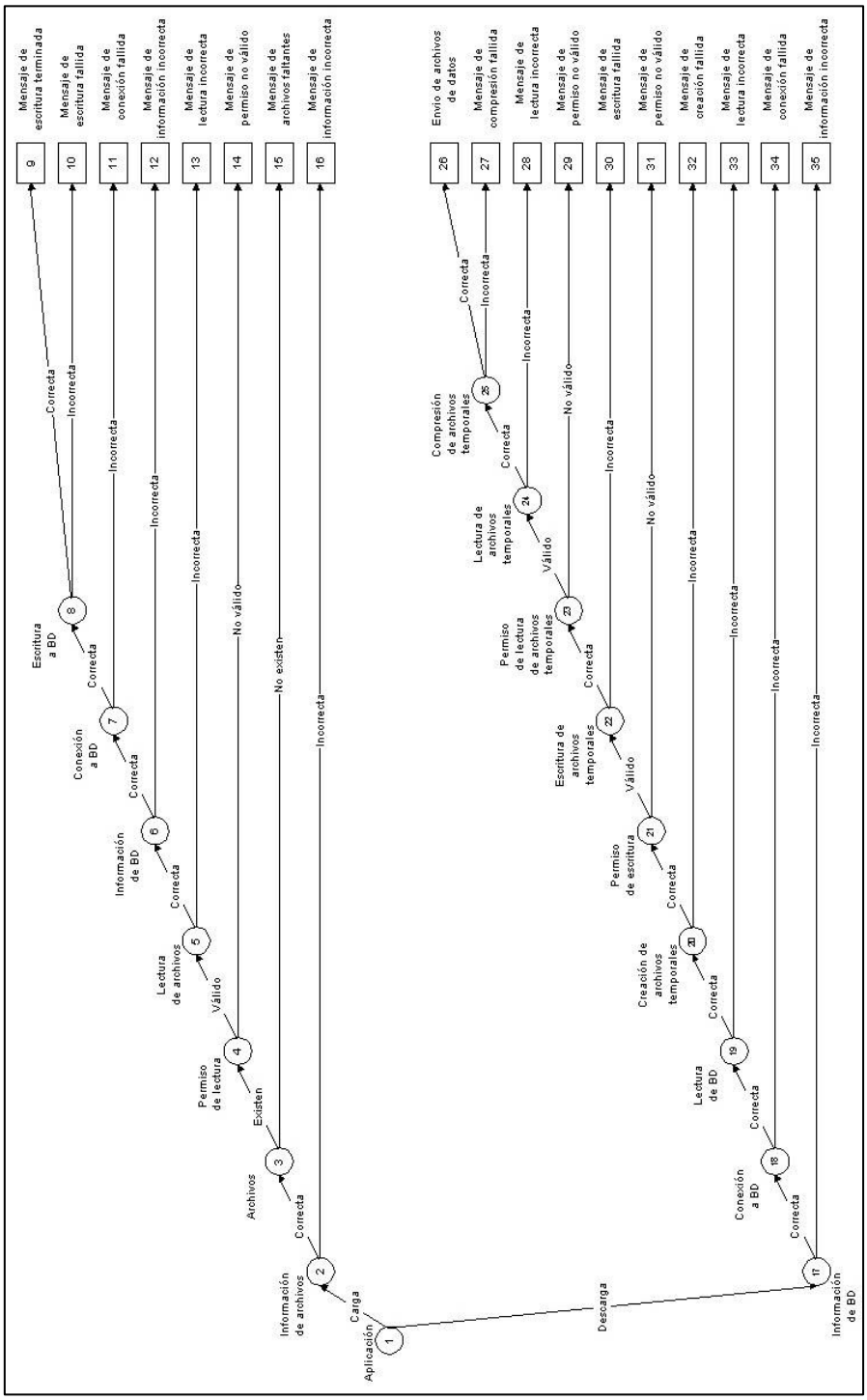


Figura 1- 7 Árbol de Decisión del problema estudiado.

Tablas de Decisión.

Una tabla de decisión es un diagrama de toda la lógica y posibles salidas asociadas con un proceso en particular.

La tabla de decisión puede ser dividida en tres áreas básicas: reglas de procesos, condiciones, y acciones. Las reglas de procesos, combinadas con las condiciones, representan las reglas de decisión, las cuales establecen que condiciones deben satisfacerse para llevar a cabo un conjunto de acciones en particular.

Desarrollo de tablas de decisión.

Una tabla de decisiones es una tabla de renglones y columnas que contiene cuatro secciones: establecimiento de las condiciones, entrada de las condiciones, establecimiento de las acciones, y entradas de las acciones. La sección de establecimiento de condiciones identifica las condiciones importantes. Las entradas de condición señalan que valor aplica para una condición. El establecimiento de acciones lista el conjunto de todas las acciones que pueden llevarse acabo cuando se cumple una condición. Las entradas de acción indican bajo que regla se ejecuta esa acción. Ver Tabla 1-13.

Condiciones y Acciones	Reglas de decisión
Establecimiento de condiciones	Entradas de condiciones
Establecimiento de acciones	Entradas de acciones

Tabla 1- 13 Forma general de una tabla de decisión.

Los pasos a seguir para desarrollar una tabla de decisión son:

1. *Determinar el número de condiciones que pudiera afectar la decisión.* El número es igual al número de renglones en la sección de Establecimiento de condiciones.
2. *Determinar el número de acciones posibles que puedan realizarse.* Este número será igual al número de renglones en la sección de Establecimiento de acciones.
3. *Estudiar las combinaciones de condiciones que son posibles.* Para cada número N de condiciones existen EN número de combinaciones posibles que pueden considerarse, donde E es el número de posibles valores de entrada para cada condición. Este número es igual al número de columnas en la sección Entradas de condiciones.
4. *Llenar las opciones de condición.* Empezando con la primer condición, dividir el número de columnas de la sección Entradas de condición entre el número de posibles valores de entrada, llenar el número de columnas resultante con el primer valor, después llenar el mismo número de columnas con el siguiente valor, y así sucesivamente hasta completar el total de columnas. Repetir lo anterior para la siguiente condición, utilizando un subconjunto de columnas igual al número de columnas que tienen el mismo valor en la condición anterior. Continuar con este patrón para cada condición.

5. *Llenar la sección de Entradas.* Llenar la sección Entradas de acción con una X donde las reglas sugieran cierta acción.
6. *Eliminar la redundancia.* La redundancia se presenta cuando son verdad los siguientes puntos: a) dos reglas de decisión son idénticas excepto por una condición y b) las acciones para las dos reglas son idénticas.

Regla	1	2
Condición 1	S	S
Condición 2	S	N
Acción 1	X	X

Regla	1	
Condición 1	S	
Condición 2	-	
Acción 1	X	

   El guión (-) significa que la condición 2 puede ser S o N y la acción 1 aún así se lleva a cabo.
7. *Eliminar contradicciones.* Las reglas de decisiones se contradicen cuando dos o mas reglas son idénticas y las acciones son diferentes (Esto significa que hay un error en la información).

Tipos de entradas de tablas.

Existen tres tipos de entradas para las tablas de decisión:

1. *Forma de entrada limitada.* Los valores posibles para la sección Entradas de condiciones son: “S”, “N”, o “-”, y para la sección Entradas de acciones son: “X”.
2. *Forma de entrada extendida.* Esta forma remplaza las “S” y “N” con las condiciones, las “X” con una pequeña frase que indica la acción a realizarse.
3. *Forma de entrada mixta.* Combina las características de las dos formas de entrada anteriores en una sola tabla.

Tablas múltiples.

Cuando se manejan tablas de decisión muy grandes puede llegar a ser muy difícil manejarlas, para estas circunstancias se pueden utilizar tablas de decisión múltiple ligadas entre si. Dependiendo de las acciones seleccionadas en la primera tabla, las acciones adicionales se explican por una o más tablas adicionales; cada tabla adicional proporciona más detalles sobre las acciones a realizar. Existen dos tipos de transferencias entre las tablas múltiples:

1. *Transferencia directa.* La transferencia directa utiliza una transferencia de una sola vez: la tabla a la que se hace referencia no envía el regreso a la tabla original. Para realizar la transferencia se utiliza la proposición de acción “GO TO Nombre de la tabla”

2. *Transferencia temporal.* La transferencia temporal utiliza la proposición de acción "PERFORM Nombre de la tabla", al final de la tabla a la que se hace referencia se encuentra la proposición de acción "RETURN" que manda el control de regreso a la siguiente acción después de "PERFORM" en la tabla que hizo la referencia.

Implantación del método para la solución del problema.

Para realizar la implantación de las tablas de decisiones en el problema de estudio se utilizó el tipo de entrada limitada, para el llenado de la tabla, y la transferencia directa, para lograr múltiples tablas.

De igual manera que en el método de Diccionario de Datos, el método de Tablas de Decisión nos proporciono una extensa cantidad de información, por ello solamente se presenta un ejemplo con la primera tabla (Tabla 1-14), y la tabla final que resulta después de utilizar este método. Si se requiere ver el desarrollo completo del método sobre el problema estudiado, consultar el **Apéndice A**.

Tabla 1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Carga	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N
Información de archivos correcta	S	S	S	S	N	N	N	N	S	S	S	S	N	N	N	N
Existen archivos	S	S	N	N	S	S	N	N	S	S	N	N	S	S	N	N
Permiso de lectura válido	S	N	S	N	S	N	S	N	S	N	S	N	S	N	S	N
Verificar existencia de archivos	X	X	X	X												
Verificar permiso de lectura	X	X														
Leer archivos	X															
Leer información de BD	X								X	X	X	X	X	X	X	X
GO TO Tabla 2	X															
GO TO Tabla 3									X	X	X	X	X	X	X	X
Preparar Mensaje		X	X	X	X	X	X	X								
Enviar Mensaje de error		X	X	X	X	X	X	X								

Tabla 1- 14 Tabla 1 de decisión del problema estudiado.

La Tabla 1-15 muestra el resultado de realizar una primera eliminación de redundancia de la Tabla 1.

Tabla 1	1	2	3	4	5	6	7	8	9
Carga	S	S	S	S	S	N	N	N	N
Información de archivos correcta	S	S	S	N	N	S	S	N	N
Existen archivos	S	S	N	S	N	S	N	S	N
Permiso de lectura válido	S	N	-	-	-	-	-	-	-
Verificar existencia de archivos	X	X	X						
Verificar permiso de lectura	X	X							
Leer archivos	X								
Leer información de BD	X					X	X	X	X
GO TO Tabla 2	X								
GO TO Tabla 3						X	X	X	X
Preparar Mensaje		X	X	X	X				
Enviar Mensaje de error		X	X	X	X				

Tabla 1- 15 Tabla 1 después de una primera eliminación de redundancia.

La Tabla 1-16 muestra el resultado de realizar una segunda eliminación de redundancia de la Tabla 1.

Tabla 1	1	2	3	4	5	6
Carga	S	S	S	S	N	N
Información de archivos correcta	S	S	S	N	S	N
Existen archivos	S	S	N	-	-	-
Permiso de lectura válido	S	N	-	-	-	-
Verificar existencia de archivos	X	X	X			
Verificar permiso de lectura	X	X				
Leer archivos	X					
Leer información de BD	X				X	X
GO TO Tabla 2	X					
GO TO Tabla 3					X	X
Preparar Mensaje		X	X	X		
Enviar Mensaje de error		X	X	X		

Tabla 1- 16 Tabla 1 después de una segunda eliminación de redundancia.

La Tabla 1-17 muestra el resultado de realizar una tercera eliminación de redundancia de la Tabla 1.

Tabla 1	1	2	3	4	5
Carga	S	S	S	S	N
Información de archivos correcta	S	S	S	N	-
Existen archivos	S	S	N	-	-
Permiso de lectura válido	S	N	-	-	-
Verificar existencia de archivos	X	X	X		
Verificar permiso de lectura	X	X			
Leer archivos	X				
Leer información de BD	X				X
GO TO Tabla 2	X				
GO TO Tabla 3					X
Preparar Mensaje		X	X	X	
Enviar Mensaje de error		X	X	X	

Tabla 1- 17 Tabla 1 después de una tercera eliminación de redundancia.

Lenguaje Estructurado.

El lenguaje estructurado consiste en representar un proceso usando el lenguaje cotidiano con una estructura restringida (ya que el lenguaje cuenta con estructura).

El lenguaje estructurado se basa en: 1) la lógica estructurada, o en instrucciones que se organizan en procesos agrupados o cíclicos; 2) planteamientos sencillos del idioma tales como sumar, multiplicar, mover y otros similares, y 3) el control de la ambigüedad.

Las especificaciones del lenguaje estructurado requieren que se identifiquen las condiciones que ocurren dentro de un proceso, las decisiones que deben realizarse cuando se dan y las acciones alternas que hay que realizar.

Redacción del lenguaje estructurado.

El lenguaje estructurado utiliza tres tipos básicos de estructuras para describir un proceso. Estas instrucciones son:

1. *Estructura de Secuencia.* Es un solo paso o acción dentro de un proceso. No depende de la existencia de ninguna condición y, cuando se encuentra, siempre se lleva a cabo.
2. *Estructura de Decisión.* Ocurren cuando dos o más acciones se pueden llevar a cabo, dependiendo del valor de una condición específica. Se debe considerar la condición y entonces tomar la decisión para cumplir con las acciones establecidas o conjunto de acciones para esa condición.
3. *Estructura de Iteración.* Sirve para realizar ciertas acciones mientras se cumpla una cierta condición o hasta que ocurra cierta condición.

El lenguaje estructurado emplea verbos de acción, en infinitivo, para describir una acción; ejemplo, leer, escribir, enviar. Utiliza frases con nombres para describir estructuras de datos específicos o elementos específicos tales como nombre_cliente, id_cliente.

Las siguientes son opciones a considerar en la redacción del lenguaje estructurado:

a) Expresar toda la lógica en términos de estructuras secuenciales, de decisión o de iteración.
b) Utilizar y aprovechar términos como: SI, ENTONCES, DE LO CONTRARIO, EJECUTE, EJECUTE MIENTRAS, EJECUTE HASTA, PARA.
c) Utilizar sangrías en los bloques de proposiciones para mostrar con claridad la jerarquía.
d) Si las palabras o frases utilizadas han sido definidas en un Diccionario de Datos, destacar dichas palabras o frases.

Implantación del método en el problema.

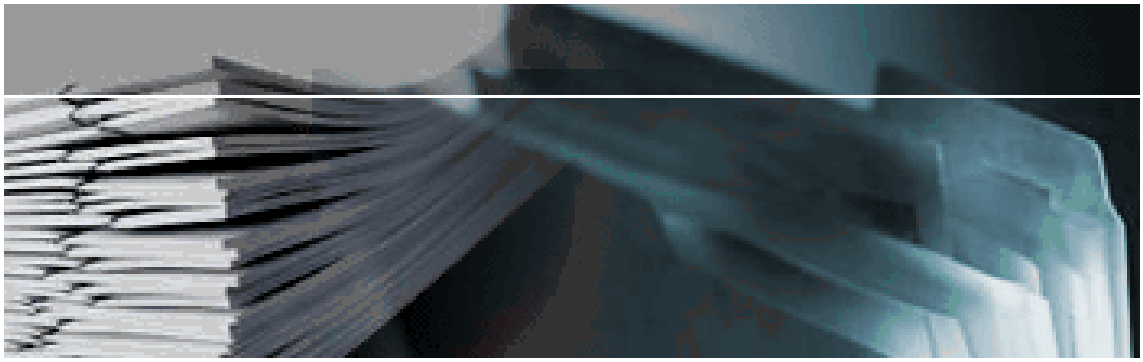
La implantación de este método en el problema de estudio, arrojó el siguiente resultado.

```
SI carga ENTONCES
    Pedir Información de los datos a cargar
    SI ruta no nula ENTONCES
        PARA todos los archivos requeridos
            Verificar existencia de Archivo de Datos
        FIN PARA
        SI Archivos Existentes ENTONCES
            PARA todos los archivos
                Verificar permiso de lectura de Archivos de Datos
            FIN PARA
            SI Permiso de Lectura Válido ENTONCES
                Obtener información de la BD
                SI Datos de conexión ENTONCES
                    Realizar conexión a BD
                    SI Conexión válida ENTONCES
                        PARA cada uno de los archivos
                            Leer primera línea
                            PARA cada línea leída
                                SI línea leída es fin de archivo ENTONCES
                                    Continuar con siguiente archivo
                                FIN SI
                                Obtener datos de línea leída
                                PARA cada dato
                                    Verificar Existencia de Dato
                                    SI Dato Existente ENTONCES
                                        Determinar Tipo de Dato
                                        Preparar Formato de Dato
                                    FIN SI
                                    DE LO CONTRARIO
                                        Colocar Dato Nulo
                                    FIN DE LO CONTRARIO
                                FIN PARA cada dato
                                Escribir en BD
```

Leer siguiente línea de archivo
FIN PARA cada línea leída
FIN PARA cada uno de los archivos
Preparar Mensaje de Carga: "Ejecución correcta, carga finalizada"
Enviar Mensaje de Carga
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Carga: "Conexión fallida"
Enviar Mensaje de Carga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Carga: "Información errónea de la base de datos"
Enviar Mensaje de Carga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Carga: "Permiso de lectura no válido"
Enviar Mensaje de Carga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Carga: "Archivos no existentes"
Enviar Mensaje de Carga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Mostrar mensaje de **ruta** nula
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Pedir **Información de los datos a descargar**
SI **datos a descargar** no nulos ENTONCES
Obtener Información de la BD
SI **Datos de conexión** ENTONCES
PARA cada tabla de la BD
Realizar conexión a BD
SI **Conexión válida** ENTONCES
Leer la BD
SI **Datos Leídos** ENTONCES
Crear Archivos Temporales
SI **Archivos Temporales creados** ENTONCES
Verificar Permiso de Escritura de Archivos Temporales
SI **Permiso de Escritura válido** ENTONCES
PARA cada registro de la tabla leída
Escribir Datos en Archivos Temporales
FIN PARA cada registro de la tabla leída
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Permiso de Escritura no válido"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Archivos Temporales no creados"
Enviar Mensaje de Descarga

FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Datos no Leídos"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Conexión fallida"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN PARA cada tabla de la BD
SI **Datos Escritos** ENTONCES
PARA cada Archivo Temporal
Verificar Permiso de Lectura de Archivos Temporales
FIN PARA cada Archivo Temporal
SI **Permiso de Lectura de Archivos Temporales** ENTONCES
Leer Archivos Temporales
SI **Archivos Temporales Leídos** ENTONCES
Realizar Compresión de Archivos Temporales
SI **Archivos Comprimidos** ENTONCES
Enviar Archivos de Datos
Preparar Mensaje de Descarga: "Ejecución correcta, descarga finalizada"
Enviar Mensaje de Descarga
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Archivos no Comprimidos"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Archivos Temporales no Leídos"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Permiso de Lectura no válido Archivos Temporales"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Datos no escritos en Archivos Temporales"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Preparar Mensaje de Descarga: "Información errónea de la base de datos"
Enviar Mensaje de Descarga
FIN DE LO CONTRARIO
FIN SI
DE LO CONTRARIO
Mostrar mensaje de **datos a descargar** nulos
FIN DE LO CONTRARIO
FIN DE LO CONTRARIO

Capítulo 2: Marco Teórico.



Introducción.

En este capitulo se dan algunas de las definiciones, teorías y antecedentes de los conceptos usados en este trabajo. Los temas que se tratan son: Arquitecturas Computacionales, Bases de Datos, Lenguajes de Programación y Tecnologías Utilizadas.

En la parte de Arquitecturas Computacionales se revisa la arquitectura basada en servidor, la arquitectura basada en cliente, la arquitectura cliente-servidor y la división en n capas para la arquitectura cliente-servidor.

En la parte de Bases de Datos se menciona una breve historia, como fue en las décadas de 1960, 1970 y 1980, a principios y finales de la década de 1990 y en el siglo XXI. Se menciona también la definición de lo que es una base de datos, lo que es un sistema gestor de base de datos, los tipos de bases de datos (jerárquica, en red, relacional y orientada a objetos) y las reglas de Codd para las bases relacionales, los tipos de datos de una base de datos y los lenguajes de una base de datos (lenguaje de definición de datos y lenguaje de manipulación de datos).

En el tema de Lenguajes de Programación se habla sobre los niveles de lenguajes de programación, la clasificación de los lenguajes de programación y se da una información breve sobre los lenguajes utilizados. En la parte de niveles de lenguajes de programación se mencionan los de bajo nivel (lenguaje máquina o 1^{er} nivel y lenguaje ensamblador o 2^{o} nivel) y de los lenguajes de alto nivel (3^{er} nivel y generadores de aplicaciones o 4^{o} nivel). En la parte de la clasificación de de los lenguajes de programación se mencionan la clasificación en base a su forma de programar (lenguajes imperativos, lenguajes declarativos, lenguajes orientados a objetos, lenguajes orientados al problema, lenguajes funcionales y lenguajes lógicos), en base a su área de aplicación (científica, de negocios, inteligencia artificial, programación de sistemas, lenguajes de script) y en base a su método de implementación (ensambladores, compiladores, interpretes e híbridos). En la parte donde se da una información breve sobre los lenguajes utilizados se habla sobre Java, JavaScript, HTML y SQL y las ventajas de cada uno.

En el tema de Tecnologías Utilizadas se habla sobre las tecnologías de Servlets, Java Server Pages (JSP) y JavaBeans y sobre las ventajas que presenta cada una.

En todos los temas se agregaron ejemplos de lo que se estaba tratando.

Marco Teórico.

Arquitecturas Computacionales.

Hay tres arquitecturas computacionales fundamentales. En la computación basada en servidor, también llamada computación centralizada, el servidor virtualmente realiza todo el trabajo. En la computación basada en cliente, también llamada presentación distribuida, las computadoras cliente son las responsables de la mayoría de funciones de la aplicación. En la computación cliente-servidor, también llamada de datos distribuidos o de lógica y datos distribuidos, el trabajo es compartido entre el servidor y los clientes.

El trabajo hecho por un sistema puede ser dividido en cuatro funciones generales. La primera es almacenamiento de datos. La mayoría de los programas requieren datos que son almacenados y recuperados, si no en un pequeño archivo, en una gran base de datos. La segunda función es la lógica de acceso a los datos, el procesamiento requiere acceso a los datos, lo cual significa consulta a la base de datos en SQL. La tercera función es la lógica de la aplicación, esta es la lógica que es documentada en el Diagrama de Flujo de Datos. La cuarta función es la lógica de presentación, la presentación de información hacia el usuario y la admisión de comandos del usuario (la interfaz de usuario).

Los componentes de las arquitecturas computacionales incluyen servidores, clientes, y redes. Los servidores pueden ser: mainframes, minicomputadoras, y microcomputadoras. Un mainframe es una computadora de muy grande propósito general; es decir, es capaz de ejecutar muchas funciones simultáneamente. Una minicomputadora es una computadora de gran propósito general; es decir, es utilizada para dedicarla a propósitos específicos tales como soportar un sistema administrador de base de datos o un proceso de control en manufacturación. La microcomputadora (o computadora personal) puede ser una maquina de escritorio de uso común.

El cliente, es el dispositivo de hardware de entrada-salida que es empleado por el usuario. Existen tres categorías principales de clientes: terminales, computadoras personales, y terminales de propósito especial. Las terminales son usadas solamente para el despliegue y entrada de datos, mientras que las computadoras personales son “inteligentes” y pueden manejar un gran numero de tareas de procesamiento. Las terminales de propósito especial incluyen aquellas como cajeros automáticos, kioscos, y cualquier otra interfase con la cual es usuario pueda comunicarse con una aplicación.

Para este trabajo, basta con entender que una red es el hardware y software que se usa para conectar clientes y servidores, unos con otros. Una red soporta la comunicación entre el cliente y el servidor, componentes que hacen la arquitectura computacional.

Arquitectura basada en servidor.

La primera arquitectura fue la basada en servidor, en la cual el servidor realizaba las cuatro aplicaciones. El cliente permitía al usuario enviar hacia y recibir mensajes desde la computadora servidor. Los clientes solamente capturaban pulsaciones de teclas y las

enviaban hacia el servidor para procesarlas, aceptaban instrucciones del servidor que desplegaban. La Figura 2-1 muestra una arquitectura basada en servidor.

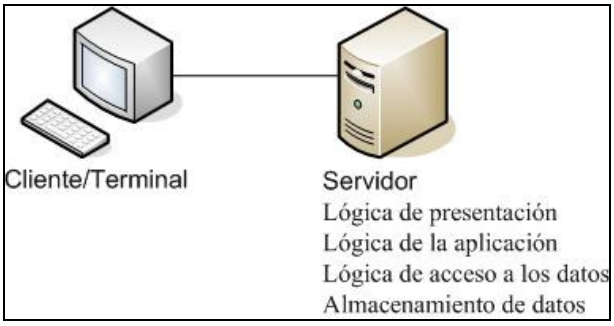


Figura 2- 1 Arquitectura basada en servidor.

El software de la aplicación es desarrollado y almacenado en una computadora, y todos los datos están en la misma computadora. Hay un punto de control porque todo el flujo de mensajes es lanzado en un servidor central.

El problema fundamental con la arquitectura basada en servidor es que el servidor debe procesar todos los mensajes y operaciones.

Algunos ejemplos de aplicaciones realizadas bajo esta arquitectura pueden ser: el sistema operativo sun 6000, el sistema operativo hp-9000, las aplicaciones que se utilizan en las tiendas departementales (kioskos donde consultas saldos), o las aplicaciones en las tiendas de autoservicio donde consultas los precios (verificadores de precio).

Arquitectura basada en cliente.

En la arquitectura basada en cliente, los clientes son computadoras personales en una Red de Área Local (LAN, por sus siglas en ingles, Local Area Network), y la computadora servidor también esta en la misma red. El software de la aplicación esta en las computadoras cliente y es responsable de la lógica de presentación, de la lógica de aplicación, y de la lógica de acceso a los datos; el servidor simplemente almacena los datos. La Figura 2-2 muestra la arquitectura basada en cliente.

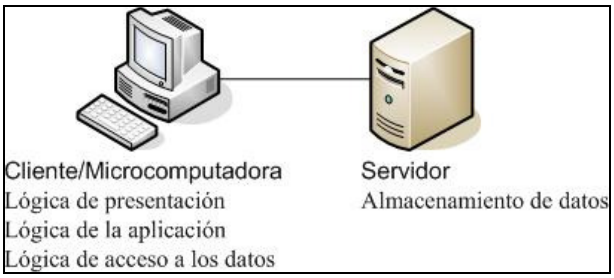


Figura 2- 2 Arquitectura basada en cliente.

El problema fundamental en la arquitectura basada en cliente es que todos los datos del servidor deben viajar hacia el cliente para ser procesados.

Algunos ejemplos de aplicaciones realizadas bajo esta arquitectura pueden ser: los sistemas de cajas de un banco, un sistema de inscripción escolar, un sistema de control de suscriptores de una empresa proveedora de cable.

Arquitectura Cliente-Servidor.

Buscando el balance de procesamiento entre el cliente y el servidor, por lo cual ambos tengan un poco de las funciones de la aplicación, se avanzo a la arquitectura Cliente-Servidor o de dos capas. En esta arquitectura, el cliente es responsable de la lógica de presentación, mientras que el servidor es responsable de la lógica de acceso a los datos y del almacenamiento de datos. La lógica de la aplicación puede estar en el cliente, en el servidor o dividida en ambos. La Figura 2-3 muestra la arquitectura cliente-servidor.

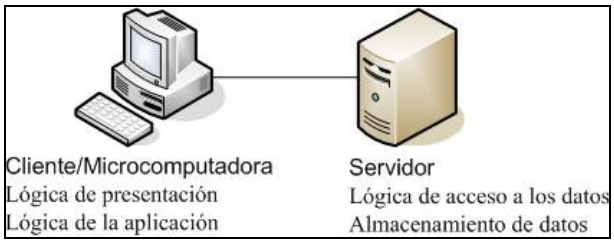


Figura 2- 3 Arquitectura cliente-servidor.

La arquitectura Cliente-Servidor tiene algunos beneficios importantes.

- Es escalable.
- Puede soportar diferentes tipos de clientes y servidores.
- No una sola computadora servidor soporta toda la aplicación.

La arquitectura Cliente-Servidor también tiene algunas limitaciones críticas:

- Es compleja.
- Actualizar una versión del software es más complicado.
- En una actualización, se deben actualizar tanto clientes como servidores.

Actualmente la mayoría de las empresas que tienen aplicaciones propias se están moviendo a este tipo de arquitectura.

Arquitectura Cliente-Servidor de 3 capas

Hay muchas formas en las que la lógica de la aplicación puede ser dividida entre el cliente y el servidor. Un caso es en el que el servidor es responsable de los datos y el cliente es responsable de la aplicación y la presentación; este es llamado arquitectura de dos capas porque usa solamente dos conjuntos de computadoras, clientes y servidores.

Una arquitectura de tres capas utiliza tres conjuntos de computadoras. En este caso, el software en la computadora cliente es responsable de la lógica de presentación, un servidor de aplicación es responsable de la lógica de aplicación, y un servidor separado de base de datos es responsable de la lógica de acceso a los datos y del almacenamiento de datos. La Figura 2-4 muestra la arquitectura cliente-servidor de tres capas.

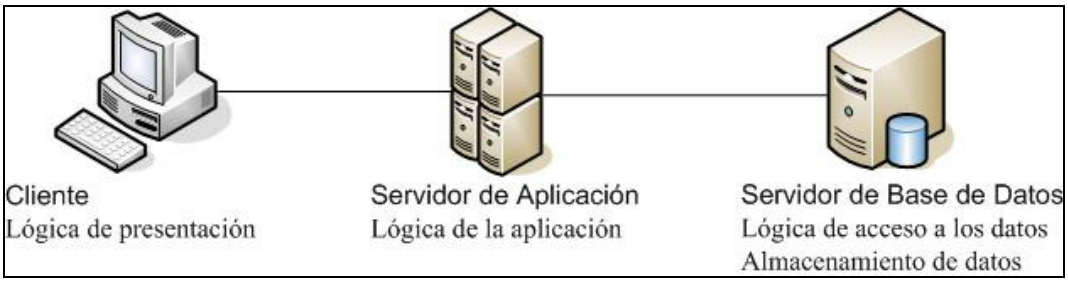


Figura 2- 4 arquitectura cliente-servidor de tres capas.

Una arquitectura de n capas utiliza más de tres conjuntos de computadoras. En este caso, el cliente es responsable de la presentación, un servidor de base de datos es responsable de la lógica de acceso a los datos y del almacenamiento de datos, y la lógica de la aplicación es expandida a través de dos o más diferentes conjuntos de computadoras. La Figura 2-5 muestra una arquitectura cliente-servidor de cuatro capas.

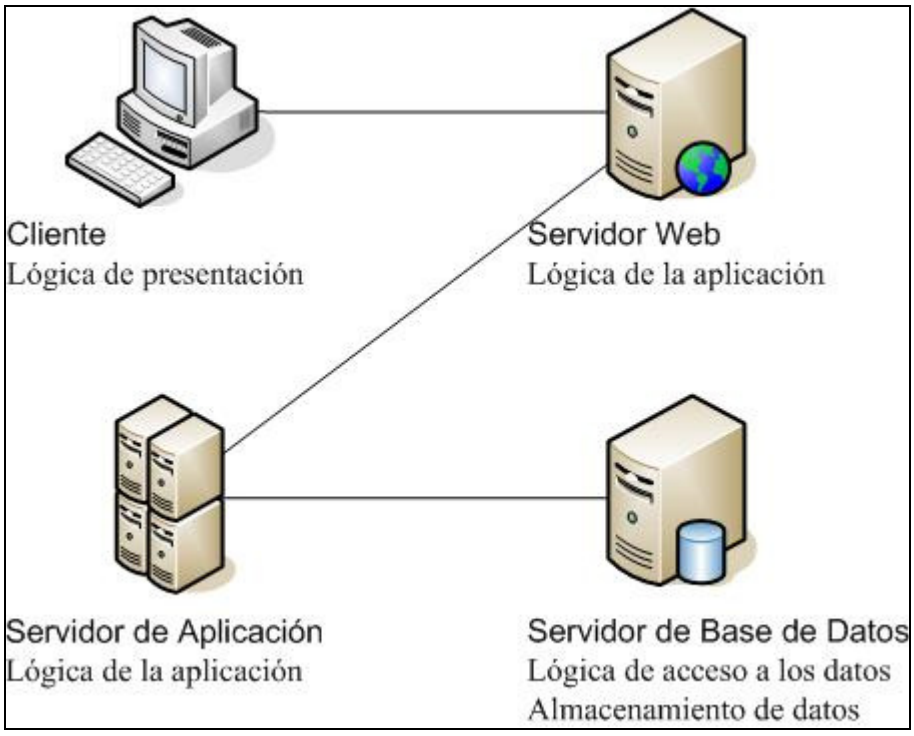


Figura 2- 5 Arquitectura cliente-servidor de cuatro capas.

Bases de datos (BD).

Historia.

Década de 1960.

Las BD se originaron entre 1960 y 1962, junto con las computadoras que trabajaban a base de tarjetas perforadas. Las DB se crearon con el objetivo de almacenar las grandes cantidades de datos que se registraban y guardaban en libros, lo cual era bastante costoso, complejo y lento; por ejemplo, cuando se modificaba un registro de algún libro, era necesario modificar ese registro en todos los libros que apareciera.

Las primeras BD se manejaban en archivos que se almacenaban en tarjetas o soportes magnéticos. Cuando las computadoras empezaron a evolucionar y se volvieron más poderosas y fáciles de manejar, ese es el momento en el que las BD empiezan a ser realmente útiles.

Los discos dieron inicio a las Bases de Datos, de red y jerárquicas, pues los programadores con su habilidad de manipulación de estructuras junto con las ventajas de los discos era posible guardar estructuras de datos como listas y árboles.

Década de 1970.

Edgar Frank Codd, en su artículo "Un modelo relacional de datos para grandes bancos de datos compartidos" ("A Relational Model of Data for Large Shared Data Banks") en 1970, definió el modelo relacional y publicó una serie de reglas para la evaluación de administradores de sistemas de datos relacionales y así nacieron las bases de datos relacionales.

Inicialmente no se uso el modelo relacional debido a que tenía inconvenientes por el rendimiento, ya que no podían ser competitivas con las bases de datos jerárquicas y de red. Ésta tendencia cambio por un proyecto de IBM el cual desarrolló técnicas para la construcción de un sistema de bases de datos relacionales eficientes, llamado System R.

Década de 1980.

Las bases de datos relacionales con su sistema de tablas, filas y columnas, pudieron competir con las bases de datos jerárquicas y de red, ya que su nivel de programación era bajo y su uso muy sencillo.

En esta década el modelo relacional ha conseguido posicionarse del mercado de las bases de datos. Y también en este tiempo se iniciaron grandes investigaciones paralelas y distribuidas, como las bases de datos orientadas a objetos.

Principios década de los 90.

Para la toma de decisiones se crea el lenguaje SQL, que es un lenguaje programado para consultas. El programa de alto nivel SQL es un lenguaje de consulta estructurado que analiza grandes cantidades de información el cual permite especificar diversos tipos de operaciones frente a la misma información, a diferencia de las bases de datos de los 80 que eran diseñadas para las aplicaciones de procesamiento de transacciones. Los grandes distribuidores de bases de datos incursionaron con la venta de bases de datos orientada a objetos.

Finales de la década de los 90.

El boom de esta década fue la aparición de la WWW "Word Wide Web ya que por éste medio se facilitaba la consulta de las bases de datos. Actualmente tienen una amplia capacidad de almacenamiento de información, también una de las ventajas es el servicio.

Siglo XXI.

En la actualidad existe gran cantidad de alternativas en línea que permiten hacer búsquedas orientadas a necesidades especificas de los usuarios, una de las tendencias más amplias son las bases de datos que cumplan con el protocolo Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH) los cuales permiten el almacenamiento de gran cantidad de artículos que permiten una mayor visibilidad y acceso en el ámbito científico y general.

Definición.

Cada autor define a una BD con diferentes palabras y frases, pero la esencia de esa definición se mantiene a través de todas ellas; dicha esencia puede mencionarse como sigue: "Conjunto de datos almacenados y organizados bajo un esquema estructurado, es decir una estructura de datos, para obtener una finalidad".

Con la frase antes mencionada se puede entender que una BD es: una colección de datos (que integran la información) almacenada en archivos organizados bajo cierta estructura lógica que nos permiten llegar a un fin deseado.

Sistema Gestor de Base de Datos.

Un Sistema Gestor de Base de Datos (SGBD), o también conocido como Sistema Administrador de Base de Datos (SABD) o como Servidor de Base de Datos (SBD), es básicamente una colección de datos interrelacionados (Base de Datos) y un conjunto de programas (Software) para acceder y o modificar dichos datos. El objetivo principal de un SGBD es proporcionar un entorno que sea tanto conveniente como eficiente para las personas que lo ocupan en el almacenamiento y recuperación de la información. Existen varios productos de software, fabricados por diferentes proveedores y con diferentes características que los diferencian unos de otros, que son SGBD, entre los que podemos

mencionar: MySQL, MS-SQL Server, Dbase, Informix, Oracle. Las características de cada SGBD, son establecidas por el fabricante del mismo, y son estas características las que hacen más o menos popular.

Tipos de base de datos.

La mayoría de los autores mencionan principalmente cuatro tipos para organizar lógicamente los archivos que contienen los datos, los cuales son: el jerárquico, de redes, el relacional, y el orientado a objetos. Con estos cuatro tipos de organización lógica de los datos se obtienen cuatro tipos de bases de datos: base de datos jerárquica, base de datos en red, base de datos relacional, y base de datos orientada a objetos.

Base de datos jerárquica.

La BD jerárquica es una base que tiene una estructura arborescente, es decir, los datos se organizan como una colección de árboles (raíz y sus ramificaciones). Un dato se interconecta con otro a través del esquema *un padre-un hijo*. El nodo que no tiene padre es llamado raíz, y los nodos que no tienen hijos son llamados hojas. La Figura 2-6 muestra la estructura de una Base de datos jerárquica.

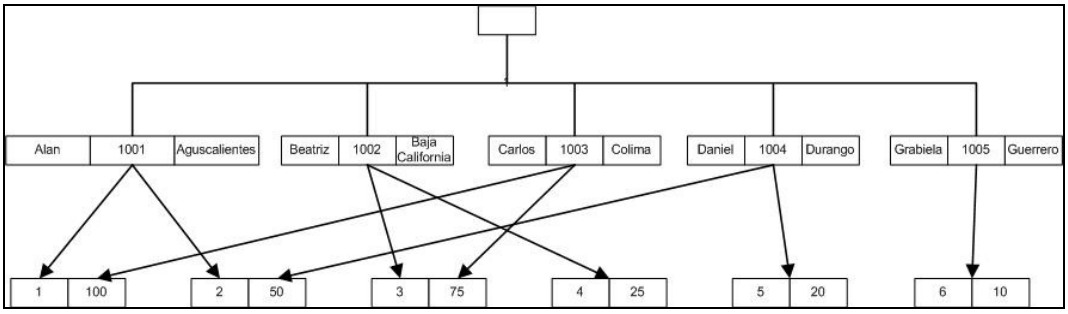


Figura 2- 6 Estructura de base de datos jerárquica.

La consulta de datos se da en el sentido hijo-padre; por ejemplo, tomando la Figura 2-6 se puede buscar directamente a quien pertenece la cuenta "3", lo cual da como resultado "Beatriz" y "Carlos". Si se requiere hacer una consulta al revés, es necesario realizar una consulta secuencial a través de todos los hijos; por ejemplo, si se busca las cuentas que pertenecen a "Beatriz" es necesario pasar por todas las cuentas para saber a quien pertenecen.

Base de datos en red.

La BD en red es una base que permite la relación de sus datos en forma multidireccional. Esta BD es una variante de la BD jerárquica, por lo que al interconectar sus datos se puede encontrar el esquema *un padre-un hijo* o *muchos padres-un hijo*. La Figura 2-7 muestra la estructura de una Base de datos en red.

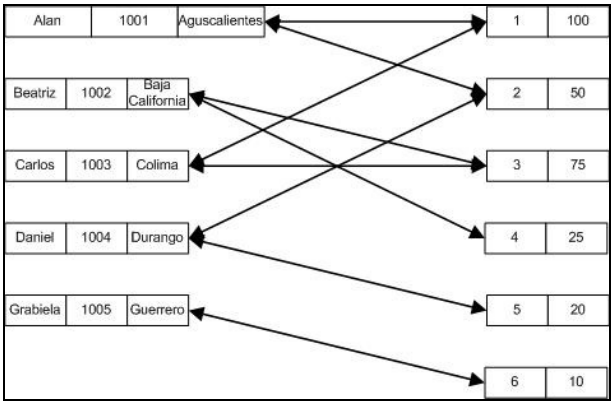
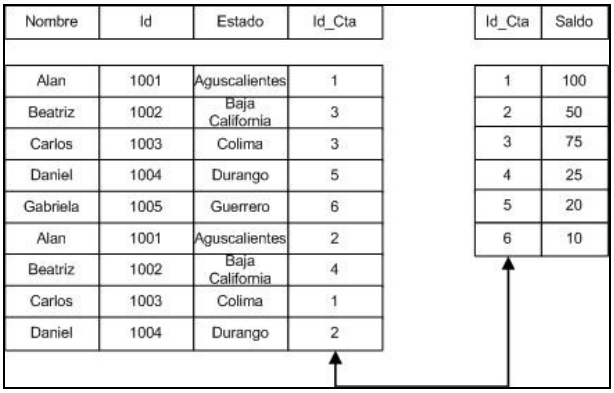


Figura 2- 7 Estructura de base de datos en red.

Este tipo de BD tiene una forma similar de consultar los datos a la del tipo jerárquico, con la diferencia de que la consulta puede ser en el sentido hijo-padre o padre-hijo indistintamente; por ejemplo, si se requiere saber las cuentas que pertenecen a "Beatriz", la búsqueda se realiza directamente sin necesidad de pasar por todas las cuentas.

Base de datos relacional.

La BD relacional es una de las más recientes, esta BD organiza todos sus datos en tablas (filas y columnas) de dos dimensiones. La interconexión entre sus datos se hace a través de una relación, es decir, una tabla tendrá interconexión con otra a través de un dato que tengan en común. En una base de datos relacional, las filas representan registros (conjuntos de datos acerca de elementos separados) y las columnas representan campos (atributos particulares de un registro). La Figura 2-8 muestra la estructura de una Base de datos relacional.



Nombre	Id	Estado	Id_Cta
Alan	1001	Aguscalientes	1
Beatriz	1002	Baja California	3
Carlos	1003	Colima	3
Daniel	1004	Durango	5
Gabriela	1005	Guerrero	6
Alan	1001	Aguscalientes	2
Beatriz	1002	Baja California	4
Carlos	1003	Colima	1
Daniel	1004	Durango	2

Id_Cta	Saldo
1	100
2	50
3	75
4	25
5	20
6	10

Figura 2- 8 Estructura de base de datos relacional.

La consulta de datos se realiza a través de buscar en dos tablas los registros que tienen en común una relación, no importando el orden de las tablas; por ejemplo, si se busca los saldos de las cuentas que pertenecen a “Beatriz”, se debe buscar todos los registros de la tabla clientes que cumplen la condición nombre = “Beatriz” y para cada registro obtenido buscar en la tabla cuentas el registro que tiene el número de cuenta que indica.

Reglas de Codd.

Codd se percató de que existían bases de datos en el mercado las cuales decían ser relacionales, pero lo único que hacían era guardar la información en las tablas, sin estar estas tablas literalmente normalizadas; entonces éste publicó 12 reglas que un verdadero sistema relacional debería tener, en la práctica algunas de ellas son difíciles de realizar. Un sistema podrá considerarse "más relacional" cuanto más siga estas reglas.

Regla No. 1 - La Regla de la información. "Toda la información en un Sistema Gestor de Bases de Datos Relacionales está explícitamente representada de una sola manera por valores en una tabla".

Cualquier cosa que no exista en una tabla no existe del todo. Toda la información, incluyendo nombres de tablas, nombres de vistas, nombres de columnas, y los datos de las columnas deben estar almacenados en tablas dentro de las bases de datos. Las tablas que contienen tal información constituyen el Diccionario de Datos.

Regla No. 2 - La regla del acceso garantizado. "Cada ítem de datos debe ser lógicamente accesible al ejecutar una búsqueda que combine el nombre de la tabla, su clave primaria, y el nombre de la columna".

Esto significa que dado un nombre de tabla, dado el valor de la clave primaria, y dado el nombre de la columna requerida, deberá encontrarse uno y solamente un valor. Por esta razón la definición de claves primarias para todas las tablas es prácticamente obligatoria.

Regla No. 3 - Tratamiento sistemático de los valores nulos. "La información inaplicable o faltante puede ser representada a través de valores nulos".

Un Sistema Gestor de Bases de Datos Relacionales (RDBMS, por sus siglas en ingles, Relational DataBase Manager System) debe ser capaz de soportar el uso de valores nulos en el lugar de columnas cuyos valores sean desconocidos o inaplicables.

Regla No. 4 - La regla de la descripción de la base de datos. "La descripción de la base de datos es almacenada de la misma manera que los datos ordinarios, esto es, en tablas y columnas, y debe ser accesible a los usuarios autorizados".

La información de tablas, vistas, permisos de acceso de usuarios autorizados, etc, debe ser almacenada exactamente de la misma manera: En tablas. Estas tablas deben ser accesibles igual que todas las tablas, a través de sentencias de SQL.

Regla No. 5 - La regla del sub-lenguaje Integral. "Debe haber al menos un lenguaje que sea integral para soportar la definición de datos, manipulación de datos, definición de vistas, restricciones de integridad, y control de autorizaciones y transacciones".

Esto significa que debe haber por lo menos un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos.

Regla No. 6 - La regla de la actualización de vistas. "Todas las vistas que son teóricamente actualizables, deben ser actualizables por el sistema mismo".

La mayoría de las RDBMS permiten actualizar vistas simples, pero deshabilitan los intentos de actualizar vistas complejas.

Regla No. 7 - La regla de insertar y actualizar. "La capacidad de manejar una base de datos con operandos simples aplica no solo para la recuperación o consulta de datos, sino también para la inserción, actualización y borrado de datos".

Esto significa que las cláusulas SELECT, UPDATE, DELETE e INSERT deben estar disponibles y operables sobre los registros, independientemente del tipo de relaciones y restricciones que haya entre las tablas.

Regla No. 8 - La regla de independencia física. "El acceso de usuarios a la base de datos a través de terminales o programas de aplicación, debe permanecer consistente lógicamente cuando quiera que haya cambios en los datos almacenados, o sean cambiados los métodos de acceso a los datos".

El comportamiento de los programas de aplicación y de la actividad de usuarios vía terminales debería ser predecible basados en la definición lógica de la base de datos, y éste comportamiento debería permanecer inalterado, independientemente de los cambios en la definición física de ésta.

Regla No. 9 - La regla de independencia lógica. "Los programas de aplicación y las actividades de acceso por terminal deben permanecer lógicamente inalteradas cuando quiera que se hagan cambios (según los permisos asignados) en las tablas de la base de datos".

La independencia lógica de los datos especifica que los programas de aplicación y las actividades de terminal deben ser independientes de la estructura lógica, por lo tanto los cambios en la estructura lógica no deben alterar o modificar estos programas de aplicación.

Regla No. 10 - La regla de la independencia de la integridad. "Todas las restricciones de integridad deben ser definibles en los datos, y almacenables en el catalogo, no en el programa de aplicación".

Las reglas de integridad son:

1. Ningún componente de una clave primaria puede tener valores en blanco o nulos. (Ésta es la norma básica de integridad).
2. Para cada valor de clave foránea deberá existir un valor de clave primaria concordante. La combinación de estas reglas asegura que haya Integridad referencial.

Regla No. 11 - La regla de la distribución. "El sistema debe poseer un lenguaje de datos que pueda soportar que la base de datos esté distribuida físicamente en distintos lugares sin que esto afecte o altere a los programas de aplicación".

El soporte para bases de datos distribuidas significa que una colección arbitraria de relaciones, bases de datos corriendo en una mezcla de distintas máquinas y distintos sistemas operativos y que esté conectada por una variedad de redes, pueda funcionar como si estuviera disponible como en una única base de datos en una sola máquina.

Regla No. 12 - Regla de la no-subversión. "Si el sistema tiene lenguajes de bajo nivel, estos lenguajes de ninguna manera pueden ser usados para violar la integridad de las reglas y restricciones expresadas en un lenguaje de alto nivel (como SQL)".

Algunos productos solamente construyen una interfaz relacional para sus bases de datos No relacionales, lo que hace posible la subversión (violación) de las restricciones de integridad. Esto no debe ser permitido.

Base de datos orientada a objetos.

Este modelo, bastante reciente, trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Las bases de datos orientadas a objetos derivan de la integración de la tecnología de base de datos con los lenguajes de programación orientados a objetos. El uso de estas bases permite que los beneficios de las características de orientado a objetos sean transportados al área de administración de datos, los cuales son:

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

La Figura 2-9 muestra un ejemplo de un esquema de base de datos orientada a objetos.

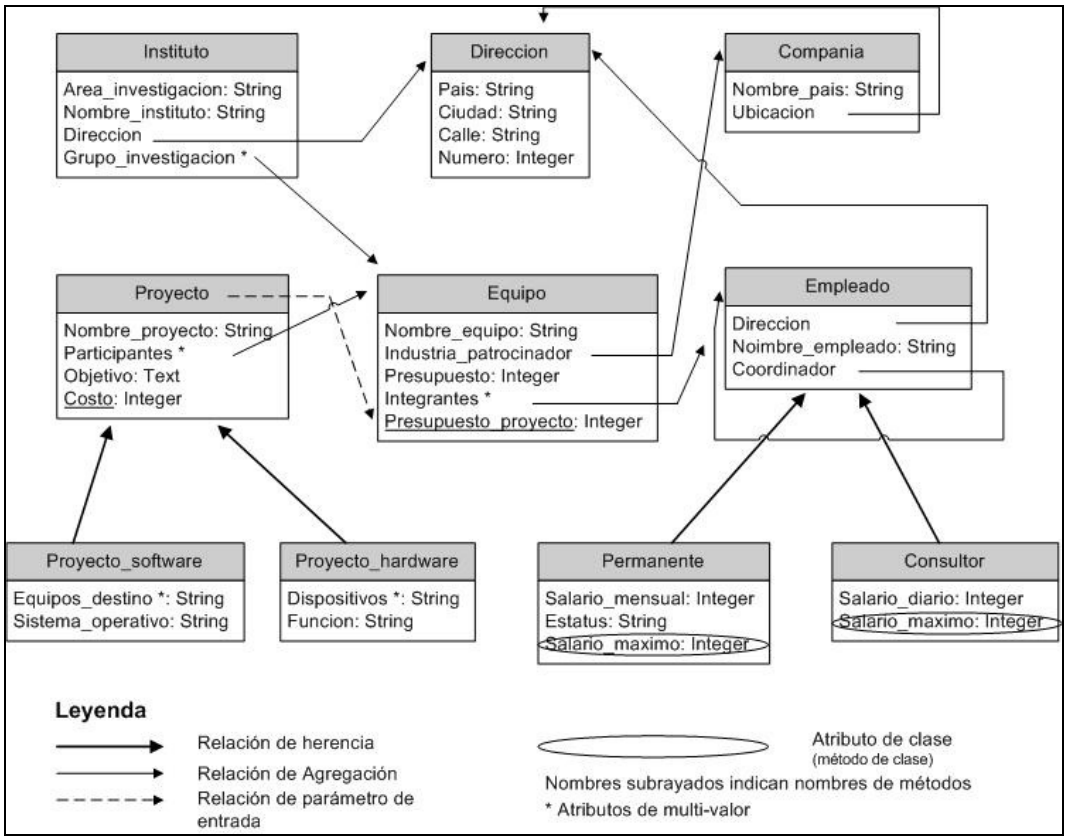


Figura 2- 9 Ejemplo de esquema de base de datos orientada a objetos.

Elementos de una BD.

La lista que sigue a continuación presenta los principales elementos que se encuentra en una BD:

- **Dato:** Es la parte esencial de la información, es decir, la información que llega y se almacena en la base de datos, y es una representación simbólica de una entidad. El dato no tiene valor semántico (sentido) en sí mismo.
- **Atributo:** Es un campo que conforma la estructura de una base de datos, y representa un determinado tipo de dato que se pretende almacenar. Además que su nombre o descripción debe estar relacionado con la información a capturar dentro de él.
- **Campo:** Es la unidad más pequeña de información en la base de datos a la que se puede acceder. La mayoría de los campos tienen atributos asociados a ellos. Por ejemplo, algunos campos son numéricos mientras otros almacenan texto, también varía el tamaño de estos.
- **Registro:** Es un conjunto de campos relacionados entre sí que contienen los datos que pertenecen a una misma repetición de entidad.
- **Archivo:** Es un conjunto de registros relacionados.

Tipos de datos de una BD.

La siguiente lista contiene los tipos de datos más comunes que se manejan en una BD:

- **Texto:** Admite contenido de caracteres alfanuméricos, el tamaño debe estar entre 1 y 255 Bytes[3].
- **Memo:** Admite contenido de tipo alfanumérico, el tamaño debe estar entre 1 y 64000 Bytes, hay manejadores que soportan tamaños mayores, lo que hacen es señalar un apuntador a un archivo externo.
- **Numéricos:** Admite contenido de valores numéricos, el tamaño debe estar entre 1, 2, 4 y 8 Bytes, dependiendo del formato de tipo numérico. El formato de tipo numérico puede ser de tipo: DOUBLE, FLOAT, INTEGER, etc.
- **Fecha y Hora:** Solo admite horas y fechas. Este tipo de dato ocupa 8 Bytes normalmente.
- **Moneda:** Admite contenido de caracteres numéricos, dándoles a estos un formato automático para una moneda (Coma de millar, símbolo del peso, etc.) dependiendo este formato de la configuración regional del panel de control. Este tipo de datos puede llevar decimales.
- **Si / No (Boleano):** Tipo de datos boléanos, los valores boléanos pueden tener dos valores posibles, 1 o 0, donde 1 es verdadero y 0 es Falso.

Lenguajes de BD.

SQL es una abreviatura de "Lenguaje de Consulta Estructurado", por sus siglas en ingles, "Structured Query Language". Con este lenguaje se puede realizar operaciones que permitan definir y manipular una base de datos relacional. Se describe a SQL como un lenguaje para realizar consultas, pero, en realidad es mucho más que eso, ya que dispone de otras funciones además de las de consultar una Base de Datos. Entre éstas se incluyen las relativas a: definición de datos y de manipulación de los datos de la Base de Datos. También cabe mencionar que SQL pertenece al grupo de los lenguajes declarativos.

Todas las sentencias SQL comienzan con un verbo, una palabra clave que describe lo que la sentencia hace. CREATE, INSERT, DELETE son verbos típicos[4]. La sentencia continua con una o más cláusulas. Una cláusula puede especificar los datos sobre los que debe actuar la sentencia, o proporcionar más detalles acerca de lo que la sentencia debe hacer. Todas las cláusulas comienzan también con una palabra clave, tal como WHERE, FROM, INTO, etc.[5]. Algunas cláusulas son opcionales, otras necesarias. La estructura y contenido específico varían de una cláusula a otra. Muchas cláusulas contienen nombres de tablas o columnas; algunas pueden contener palabras claves adicionales, constantes o expresiones

[3] Unidad de medida, de espacio para almacenamiento físico, en las computadoras.
[4] Crear, Insertar, Borrar.
[5] Donde, De, Dentro.

Lenguaje de Definición de Datos.

DDL por sus siglas en ingles, Data Definition Language. Es un conjunto de declaraciones o definiciones que permiten expresar las especificaciones del esquema de la base de datos. Permiten crear, las bases de datos, las tablas, etc. Igualmente modificar y borrar lo antes definido.

La sintaxis general de una instrucción para crear una tabla es:

CREATE TABLE nombre_de_la _tabla
{
nombre_de_columna tipo_de_dato (longitud_del_campo) [habilitación para soportar datos nulos]
...
[llave primaria]
[llave foránea]
}

Ejemplo:

```
CREATE TABLE tabla1
{
    columna1 numeric(2) NOT NULL,
    columna2 numeric(2) NOT NULL,
    columna3 varchar(150),
    ...
    columnan date,
    PRIMARY KEY (columna1),
    FOREIGN KEY (columna2)
    REFERENCES tabla2(columna1)
}
```

La sintaxis general de una instrucción para modificar una tabla es:

Agregar una columna:

ALTER TABLE nombre_de_la_tabla ADD nombre_de_columna tipo_de_dato [PRIMARY KEY]

Borrar una columna:

ALTER TABLE nombre_de_la_tabla DROP nombre_de_columna

Modificar una columna:

ALTER TABLE nombre_de_la_tabla MODIFY nombre_de_columna nueva_característica

Ejemplo:

```
ALTER TABLE tabla1 ADD columnax numeric(5)
ALTER TABLE tabla1 DROP columnan
ALTER TABLE tabla1 MODIFY columna3(100)
```

Lenguaje de Manipulación de Datos.

DML, por sus siglas en ingles, Data Manipulation Language. A diferencia del lenguaje anterior este tiene estrecha relación con las operaciones que los usuarios realizan sobre los datos almacenados. Estas operaciones pueden ser: recuperación o consulta, inserción, borrado y modificación de los datos.

La sintaxis general de una instrucción SELECT (recuperación o consulta) es:

SELECT ...
FROM ...
WHERE ...

SELECT *lista_columnas* o expresiones a ser recuperadas.
FROM *nombre_tablas* que contienen los datos requeridos.

Ejemplo:

```
SELECT columna1, columna2, ..., columnan
FROM tabla1, tabla2, ..., tablam.
```

Si se requiere la consulta de información (registros) bajo condiciones especiales, se incluye la cláusula WHERE (donde):

SELECT *lista_columnas* o expresiones a ser recuperadas
FROM *nombre_tablas* que contienen los datos requeridos.
WHERE *condiciones* que debe cumplir la sentencia.

Ejemplo:

```
SELECT columna1, columna2, ..., columnan
FROM tabla1, tabla2, ..., tablam
WHERE tabla1.column1= tabla2.columna1 AND … AND tablam.columnan < 14
```

Las condiciones pueden ser de comparación:

- **Igualdad**: =
- **Desigualdad**: != ^= <> (depende del sistema)
- **Mayor que**: >
- **Menor que**: <
- **Mayor o igual que**: >=

Se pueden usar los operadores lógicos AND, OR e IN[6].

Si se desea presentar todos los campos de la consulta basta con colocar el comodín * en la cláusula SELECT

[6] Y, O, Incluido.

Ejemplo:

```
SELECT * FROM tabla1
```

Estas son solo unas pocas instrucciones del amplio catálogo que existe para el SQL[7]. Cada SGBD maneja ciertas características particulares en el lenguaje SQL, sin embargo, todos cumplen con las características generales[8].

Lenguajes de programación.

Un lenguaje de programación es una notación, definida por una gramática o un conjunto de reglas, para escribir programas (comandos e instrucciones), a través de los cuales podemos comunicarnos con la computadora y dar así las ordenes necesarias y adecuadas para que, al ser ejecutadas, realicen un determinado proceso.

Un programa se define como el conjunto de instrucciones, comandos y símbolos reconocidos por la computadora, que le permite ejecutar una secuencia de control deseada. El programa esta formado por un conjunto de instrucciones, sentencias, bloques funcionales y grafismos que indican las operaciones a realizar.

Las instrucciones representan la tarea más elemental de un programa: leer una entrada, realizar una operación, activar una salida, etc.

Las sentencias representan el conjunto de instrucciones que definen una tarea completa: encontrar el valor de una función lógica en combinación de varias variables, consultar un conjunto de condiciones, etc.

El bloque funcional es el conjunto de instrucciones o sentencias que realizan una tarea o función compleja: contadores, registros de desplazamientos, transferencias de información, etc.

Todos estos elementos están relacionados entre sí mediante los símbolos o grafismos.

Niveles de lenguajes de programación.

Los lenguajes de programación han sido agrupados en niveles, de acuerdo al nivel de abstracción del mismo, los cuales son: bajo nivel y alto nivel.

Bajo nivel.

Los lenguajes de bajo nivel son aquellos que por sus características se encuentran más próximos a la arquitectura (hardware) de la computadora. Existen dos lenguajes dentro del bajo nivel:

[7] Si se desea conocer más acerca de las instrucciones del LDD, se recomienda consultar un manual de SQL.
[8] Las características particulares de cada SGBD, se mencionan en los manuales que vienen incluidos en la distribución correspondiente de cada proveedor de su software.

Lenguaje Máquina (1er Nivel).

Se dice que es el código binario. El fabricante de un procesador fija los bloques de bits que llevará a la Unidad Central de Proceso (CPU, por sus siglas en inglés, Central Processing Unit) a reconocer y realizar diferentes operaciones. Este lenguaje que la máquina puede interpretar y transformar en acciones es evidentemente muy difícil de usar para un ser humano.

Prácticamente nadie trabaja hoy a este nivel, excepto los diseñadores de chips (procesadores).

Las características principales de este lenguaje son:

- Ser considerado el primer lenguaje de programación.
- Ser el único lenguaje entendible directamente por una computadora.
- Basarse en la comparación de dos únicos símbolos, el cero y el uno denominados bit (**bi**nary digi**t**)
- Ser propio de un determinado procesador, es decir ningún otro procesador puede interpretarlo.

Lenguaje Ensamblador (2° Nivel)

Surge como sustituto del lenguaje máquina y esta basado en el uso de nemotécnicos (palabras abreviadas procedentes del inglés formadas por letras y números), por ejemplo "ADC" significa "sumar con reserva" (en inglés: "**AD**d with **C**arry")

Algunas de sus principales características son:

- Ejecutarse más rápido que lenguajes de alto nivel.
- Ocupar mucho menos espacio en memoria.
- No ser transportables.
- Ser considerado un lenguaje de programación difícil.

Alto nivel.

Los lenguajes de alto nivel son aquellos lenguajes que por sus características se encuentran más próximos al usuario o programador. Comúnmente los lenguajes de alto nivel se conocen como lenguajes de 3er Nivel y 4° Nivel.

3er Nivel.

Las características principales de estos lenguajes son:

- Son independientes de la arquitectura de la computadora que los utiliza.
- Tienen una mayor facilidad en el desarrollo y depuración de programas.
- Fácil aprendizaje del mismo.

Los lenguajes de alto nivel tienen necesidad de traducir los programas escritos en ellos a un lenguaje de programación tan básico como el lenguaje máquina, lo cual significa disponer de un traductor para su implementación (este concepto se cubrirá más adelante).

Generadores de aplicaciones (4º Nivel)

Posteriormente, usando estos lenguajes, se han redactado programas destinados a facilitar un número variado de operaciones en campos de aplicación específicos como simulación de fenómenos físicos, manipulación de datos estadísticos, etc. Los más avanzados y flexibles de estos programas son las planillas electrónicas u hojas de cálculo y los programas de administración de archivos o bases de datos.

Dados que tales aplicaciones no "hacen nada" sin que el usuario defina ciertas estructuras y ciertas operaciones, pueden ser consideradas como "generadores" de aplicaciones, aunque este nombre se reserva habitualmente para niveles más avanzados en que los usuarios pueden generar sistemas muy diferentes unos de otros, con "herramientas" que se parecen a lenguajes de programación. Estas herramientas conforman los lenguajes de cuarto nivel que son por esencia "programas para crear programas" con una finalidad específica, como el "CASE" destinado a facilitar el trabajo de los analistas de sistemas.

Clasificación de los lenguajes de programación.

Los lenguajes de programación pueden ser clasificados de múltiples formas, por lo que encontramos clasificación en base a su forma de programar, en base a su campo de aplicación, en base a su método de implementación.

En base a su forma de programar.

Lenguajes imperativos.

Establecen como debe ejecutarse una tarea, dividiéndola en partes que especifican como realizar cada una de las subtareas asociadas. Estos lenguajes se fundamentan en el uso de variables para almacenar valores y el uso de instrucciones que indican las operaciones a realizar sobre los datos almacenados. La mayoría de los lenguajes de alto nivel son de este tipo. Algunos ejemplos son: ADA, BASIC, C, FORTRAN, JAVA, MODULA−2, PASCAL.

Lenguajes declarativos.

En este caso, el proceso por el cual se ejecuta el programa no aparece de forma explícita en el programa, el programador no tiene que indicar el proceso detallado de cómo realizar la tarea. Algunos ejemplos son: HTML, SQL.

Lenguajes orientados a objetos.

El diseño de los programas se centra más en los datos y su estructura. Los programas consisten en descripciones de unidades denominadas objetos, que encapsulan los datos (almacenados en variables) y las operaciones que actúan sobre ellos (que indican el comportamiento del objeto). Algunos ejemplos son: Simula 67, Smalltalk.

Lenguajes orientados al problema.

Están diseñados para problemas específicos, principalmente de gestión. En estos lenguajes, los programas están formados por sentencias que ordenan que se quiere hacer. Generalmente, estos lenguajes suelen ser generadores de aplicaciones que permiten automatizar en la medida de lo posible la tarea de desarrollo de software de aplicaciones de gestión.

Lenguajes funcionales.

Los lenguajes funciones tratan al cómputo como la evaluación de funciones matemáticas y evitan el estado y datos mutables. Acentúa el uso de funciones para llegar al objetivo. De hecho en estos lenguajes los programas se construyen mediante descripciones de funciones. Los lenguajes funcionales se basan en el cálculo lambda, que es la única teoría lógica de orden superior que es demostradamente computable. Ejemplos de este tipo de programas: APL, Erlang, Haskell, Lisp.

Lenguajes lógicos.

La programación lógica consiste en la aplicación del conocimiento sobre la lógica para el diseño de lenguajes de programación. en la programación lógica, se trabaja de una forma descriptiva, estableciendo relaciones entre entidades, indicando no cómo, sino qué hacer. La ecuación de Robert Kowalski (Universidad de Edimburgo) establece la idea esencial de la programación lógica: algoritmos = lógica + control. Es decir, un algoritmo se construye especificando conocimiento en un lenguaje formal (lógica de primer orden), y el problema se resuelve mediante un mecanismo de inferencia (control) que actúa sobre aquél.; es decir, expresiones lógicas que indican las relaciones entre determinadas estructuras de datos. Ejemplos de este tipo de programas: ALF, Fril, Mercury, PROLOG.

Con base a su campo o área de aplicación.

Científica.

Tienen estructuras de datos simples, pero pueden realizar un largo número de operaciones aritméticas sobre datos con decimal. Las estructuras de datos más comunes son arreglos y matrices; las estructuras de control más comunes son contadores cíclicos y selecciones. Lenguajes de este tipo son: ALGOL 60, FORTRAN, PASCAL.

De negocios.

Se caracterizan por la facilidad para producir reportes elaborados, formas precisas de describir y almacenar números decimales y datos alfabéticos, y la habilidad para especificar operaciones aritméticas con decimales. Lenguajes de este tipo son: COBOL, SQL.

Inteligencia Artificial.

Es una amplia área de aplicaciones computacionales caracterizadas por el uso de símbolos más que números computacionales. Los símbolos computacionales significan aquellos símbolos, consistentes de nombres más que de números, que son manipulados. Lenguajes de este tipo son: LISP, PROLOG.

Programación de sistemas.

El sistema operativo y todas las herramientas de soporte de programación de un sistema computacional son conocidos colectivamente como software de sistemas. Los software de sistemas son continuamente utilizados y por si mismos deben ser eficientes. Lenguajes de este tipo son: PL/S, ALGOL, C, Java.

Lenguajes de Script.

Los lenguajes de script fueron usados para colocar una lista de comandos, llamada script (guión), en un archivo para ser ejecutada. La WWW (World Wide Web) fue la primera en usarlo ampliamente. Lenguajes de este tipo son: sh, JavaScript, PHP.

En base a su método de implementación.

Como se menciona anteriormente, los lenguajes de alto nivel necesitan ser convertidos a lenguaje máquina para que puedan ser implementados y ejecutados en una computadora. El elemento encargado de realizar esta tarea de conversión se le conoce como traductor[9]. Los traductores[10] pueden ser Ensambladores, Compiladores, Intérpretes, o Híbridos.

Ensambladores.

Son los encargados de traducir directamente los programas escritos en lenguaje Ensamblador a su equivalente en lenguaje máquina. La Figura 2-10 muestra el proceso de conversión del código ensamblador a código máquina.

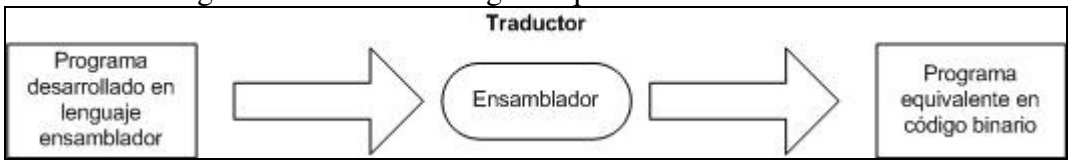


Figura 2- 10 Traductor Ensamblador.

[9] Sistema o programa de software.
[10] También se les conoce como "Traductores del lenguaje".

Compiladores.

Los compiladores son programas de software escritos en algún lenguaje de programación cuyo objetivo es traducir el correspondiente programa fuente[11] a su equivalente en lenguaje máquina, con lo cual puede ser ejecutado directamente en la computadora. Este método tiene la ventaja de que la ejecución de un programa es muy rápida, una vez que el proceso de traducción esta completado, ya que primero traduce el código fuente a lenguaje máquina y después lo ejecuta. Algunos de los lenguajes que lo usan este método son C, COBOL, etc. El método de compilación tiene varias etapas, las cuales son:

- **Edición:** Consiste en la escritura del programa. Debe realizarse mediante el uso de un editor, que puede formar parte o no del compilador utilizado. En esta fase se obtiene el denominado programa fuente.
- **Compilación:** En esta etapa el programa fuente se traduce a su equivalente en lenguaje máquina, obteniendo el llamado programa objeto.
- **Enlace:** Esta etapa consiste en unir o enlazar el programa objeto con determinadas rutinas internas del lenguaje para obtener el programa ejecutable.
- **Ejecución:** Esta etapa consiste en la llamada del programa ejecutable a través del sistema operativo.

La Figura 2-11 muestra las etapas del método de compilación.

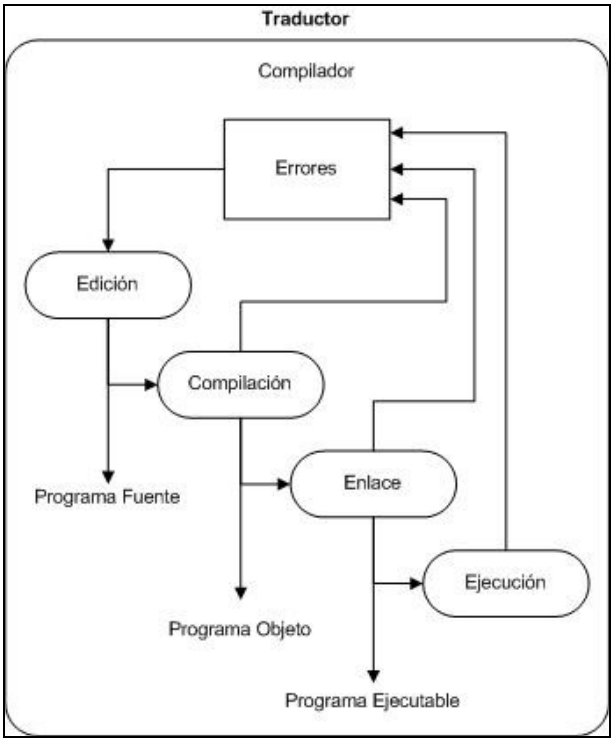


Figura 2- 11 Traductor Compilador.

[11] Archivo constituido por un conjunto de instrucciones desarrolladas en un lenguaje de alto nivel.

La etapa de compilación se divide en varias fases, las más importantes son:

- **Analizador Léxico:** El analizador léxico obtendrá, de los caracteres del código fuente, las unidades léxicas o tokens. Las unidades léxicas de un programa son identificadores, palabras especiales, operadores, y símbolos de puntuación.
- **Analizador Sintáctico:** El analizador sintáctico toma las unidades léxicas y las usa para construir estructuras jerárquicas llamadas parse trees[12]. Y verifica la correcta construcción
- **Generador de Código Intermedio (y Analizador Semántico):** El generador de código intermedio produce un programa en un lenguaje diferente, en un nivel intermedio entre el programa fuente y la salida final del compilador, el programa en lenguaje máquina. Los lenguajes intermedios algunas ocasiones son muy parecidos a los lenguajes ensamblador y en realidad algunas veces son lenguajes ensamblador actuales.
- **Generador de Código:** El generador de código traslada la versión en lenguaje intermedio del programa en un equivalente programa en lenguaje máquina.

La Figura 2-12 muestra el proceso de la etapa de compilación.

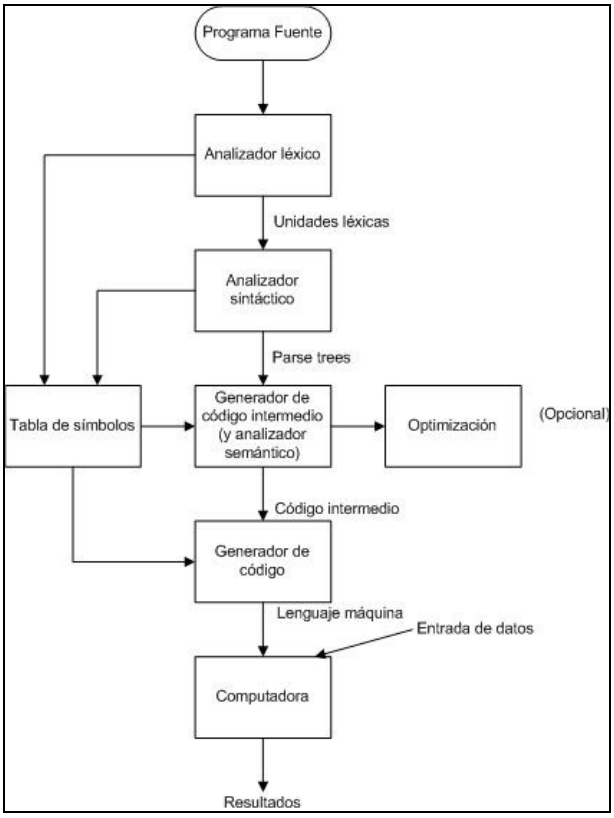


Figura 2- 12 Proceso de compilación.

[12] Árboles analizados sintácticamente.

Intérpretes.

Los intérpretes son programas que interpretan otro programa; es decir, es un software encargado de procesar y traducir cada instrucción o sentencia de un programa escrito en un lenguaje de alto nivel a lenguaje máquina y después ejecutarla. La traducción o interpretación y la ejecución no se realizan como procesos independientes, sino en una misma operación e instrucción por instrucción, respetando rigurosamente el orden establecido entre ellas. Este método tiene la ventaja de que fácilmente permite la implementación de varias operaciones de depuración a nivel código fuente, porque todos los mensajes de error en tiempo de ejecución pueden referirse a las instrucciones en el código fuente. Algunos de los lenguajes que lo usan este método son PHP, JavaScript, etc.

La Figura 2-13 muestra las etapas del método de implementación Intérpretes.

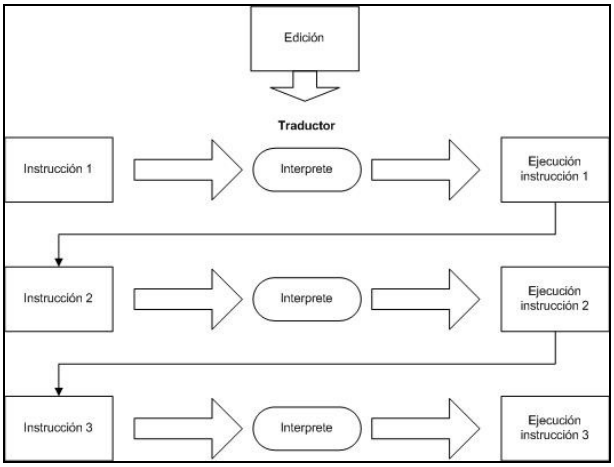


Figura 2- 13 Traductor Interprete.

La Figura 2-14 muestra el proceso del método de implementación Intérpretes.

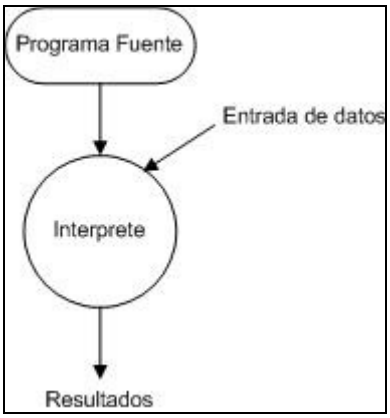


Figura 2- 14 Proceso de interpretación.

Híbridos.

Algunos sistemas de implementación de lenguaje son una composición entre compiladores e intérpretes, ellos trasladan el lenguaje de programación de alto nivel a un lenguaje intermedio designado para permitir su fácil interpretación. Un lenguaje que utiliza este método es Java.

La Figura 2-15 muestra las etapas del método de implementación Híbridos.

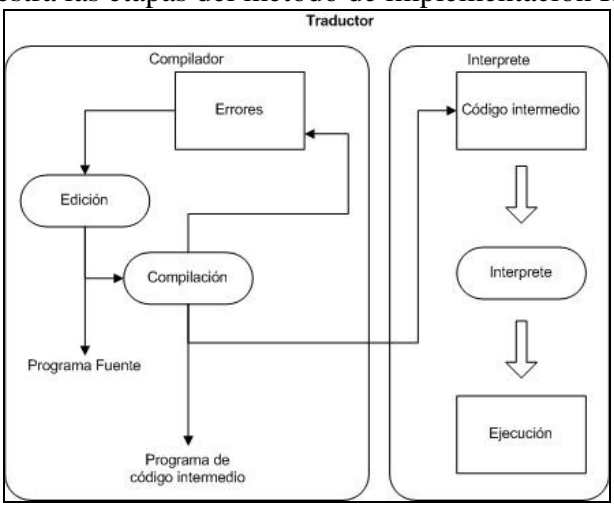


Figura 2- 15 Traductor Híbrido.

La Figura 2-16 muestra el proceso del método de implementación Híbridos.

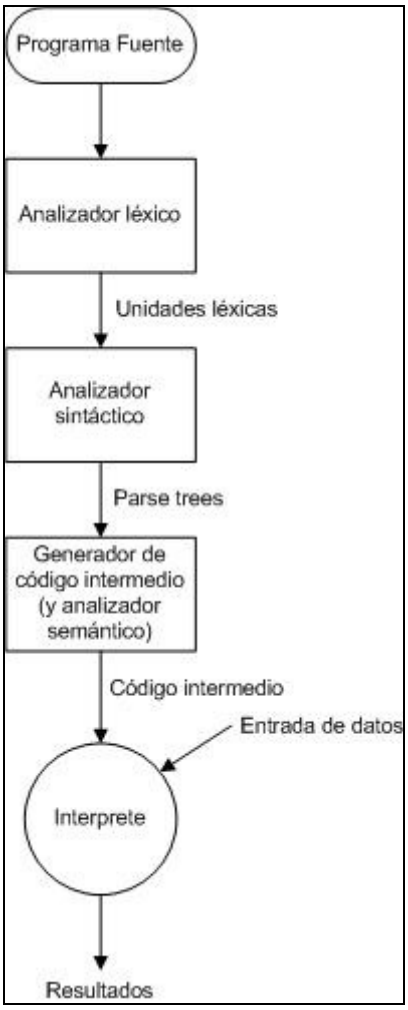


Figura 2- 16 Proceso de un híbrido.

Información breve de los lenguajes utilizados.
Java.

Al igual que muchos lenguajes, fue diseñado para una aplicación para la cual no se satisfacían las necesidades con los lenguajes existentes. Java nació como un lenguaje para dispositivos electrónicos embebidos (equipos de propósito específico controlados por microchips y software) de consumidor, tales como tostadores, hornos de microondas, y sistemas de televisión interactivos.

James Gosling, a principios de los 90 y empleado de Sun Microsystems, tenía en mente un lenguaje independiente de plataforma, que pudiera correr bajo cualquier CPU, que fuera sencillo y permitiera desarrollar software libre de errores. Trato de adecuar inicialmente el lenguaje C++ para estos propósitos, pero no cumplió con las expectativas. Se desarrolló entonces Oak (el primer nombre para Java) y se probó en un pequeño control remoto. A mediados de 1994, con el auge de la Web, el equipo de trabajo de Gosling trabajó en el desarrollo de un Browser basado en Java. WebRunner fue terminado para finales de 1994 y

se vislumbraron las ventajas de esta nueva tecnología, como una poderosa herramienta para la programación Web. Después de esto la carrera en el mundo Java ha sido vertiginosa:

- Mayo de 1995. Sun anuncia Java y HotJava al mundo.
- Verano de 1995. Los primeros desarrolladores interesados obtienen de Sun la versión Alpha del ambiente de desarrollo.
- Septiembre de 1995. Tiene lugar el primer Java contest. Sun libera el primer pre-beta.
- Diciembre 7, 1995. Microsoft pide la licencia para desarrollar productos bajo el lenguaje Java

Ventajas.

Java como lenguaje de programación presenta un conjunto de ventajas sobre las demás alternativas en lenguajes de desarrollo:

- Simple y poderoso.
- Seguro.
- Orientado por Objetos.
- Robusto.
- Interactivo.
- Independiente de arquitectura de hardware.
- Interpretado y rápido.
- Fácil de aprender

JavaScript.

Es básicamente un lenguaje de Script, que son aquellos lenguajes que se ejecutan sin que sea necesario compilarlos, como apoyo a otros lenguajes o aplicaciones mayores, y siempre dentro de una aplicación cliente.

Originalmente fue llamado LiveScript, pero luego fue renombrado con el nombre de JavaScript, con la idea de capitalizar la fama de Java. Éste es un complemento ideal del lenguaje HTML, al permitir a la página realizar algunas tareas por si misma, sin necesidad de estar sobrecargando el servidor del cual depende; JavaScript es un lenguaje diseñado especialmente para ejecutarlo en Internet. Entre estas tareas, puede estar, por ejemplo, realizar algunos cálculos simples, formatear un texto para que sea leído por distintas personas de manera distinta, proveer de un medio de configurar la visualización de una página, realizar una revisión previa de validación en formulario antes de enviarlo, etc.

Ventajas.

JavaScript es un lenguaje que tiene las siguientes ventajas:

- Se ejecuta como apoyo a otro lenguaje, el HTML.
- No necesita compilación.
- Únicamente se ejecuta dentro de un programa mayor.
- Su sintaxis es parecida a la del Java

HTML.

Para escribir documentos de hipertexto se ha desarrollado un nuevo formato de datos o lenguaje llamado Hyper Text Markup Language (HTML). Este lenguaje permite dar indicaciones precisas al programa cliente de cómo debe presentarse el documento en pantalla o al ser impreso. El lenguaje HTML, sirve por lo tanto, para realizar páginas Web. Está formado por un conjunto de identificadores, designados con el término Inglés tag, que definen el formato de una página de texto, permitiendo insertar en ella elementos multimedia, tales como imágenes, sonido y vídeo. Por lo tanto, la función del navegador de Internet es la de traducir este código un contenido gráfico. El HTML 4.0 es una aplicación Standard Generalized Markup Language (SMLG) conforme al estándar internacional ISO 8879 y está ampliamente considerado como el lenguaje de publicación estándar del World Wide Web.

Ventajas.

- HTML simplemente texto. Lo primero es saber que un documento HTML es un archivo de texto simple, por lo tanto, se puede editar con cualquier editor de textos.
- No importan los tabuladores ni los saltos de línea. Los interpretes HTML no toman en cuenta las tabulaciones, los saltos de líneas ni los espacios en blanco extra, que permite obtener resultados uniformes y de buena presentación de manera bastante fácil.

Desventajas.

- La principal desventaja es que un documento HTML, por lo menos se debe usar los comandos <P>... </P> o
 para evitar que quede todo el texto en una sola línea.

Características especiales.

- < menor que, se usa para indicar el comienzo de un comando HTML
- > mayor que, se usa para indicar el término de un comando HTML
- & Ampersand, se usa para escribir caracteres especiales (símbolos matemáticos, comerciales, así como el signo menor que y el mayor que entre otros) en un documento.

SQL.

Lenguaje desarrollado especialmente para facilitar la consulta de bases de datos, acotando progresivamente la búsqueda de ahí el nombre de "Structured Query Language".

Tecnologías Utilizadas.

Servlet

Un Servlet es un objeto Java en el API (Application Programmer Interface) de servlet y extiende la funcionalidad de un servidor HTTP. Direcciona hacia URLs y administra con una simple arquitectura. Disponible y funcionando en la mayoría de los servidores Web y los servidores de aplicaciones. Independiente de sistema operativo y servidor.

Ventajas

- No tiene las limitaciones de un CGI (Common Gateway Interface).
- Abundantes herramientas de terceros y soporte para servidores Web.
- Acceso a la familia entera de APIs de Java.
- Confiable, mejor rendimiento y escalabilidad.
- Independiente de sistema operativo y servidor.
- La mayoría de los servidores permiten recargar un servlet a través de una acción administrativa.

JSP

Un documento basado en texto con la capacidad de regresar contenido dinámico a un navegador del cliente. Contiene código HTML, XML y etiquetas JSP que facilita su programación y permite el acceso a componentes como JavaBeans.

Ventajas

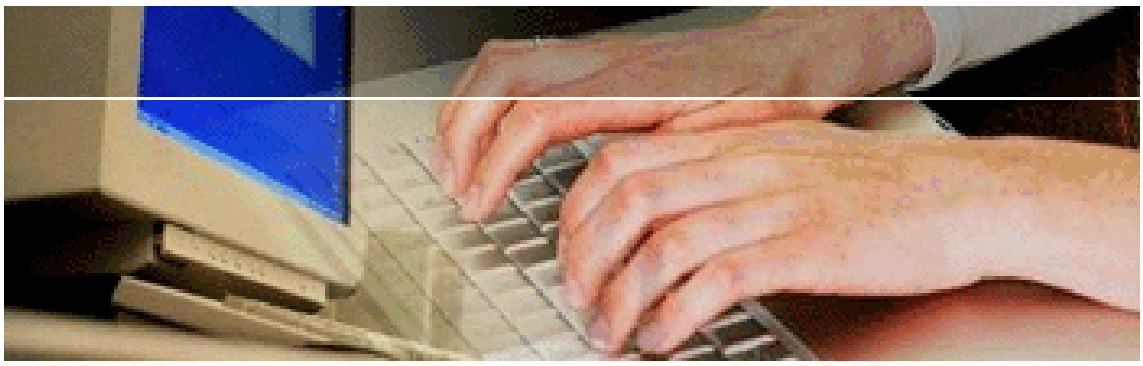
- La lógica de contenido y desplegado están separadas.
- El uso de JavaBeans y etiquetas personalizadas, simplifica el desarrollo con JSP.
- Soporta reutilización de software a través del uso de componentes.
- Recompila automáticamente cuando los cambios son hechos en el archivo fuente.
- Más fácil de escribir.
- Independiente de plataforma.

JavaBeans

Los JavaBeans son un modelo componente, portable e independiente de sistema operativo, escrito en el lenguaje de programación Java, desarrollado en colaboración con los lideres de la industria. Estos permiten a los desarrolladores escribir cada componente reutilizable y ejecutable donde sea, beneficio de la poderosa independencia de sistema

operativo de la tecnología Java. Los JavaBeans actúan como un puente entre los modelos componentes propietarios y provee un poderoso e ilimitado significado para los desarrolladores que construyen componentes que funcionan en un contenedor de aplicaciones Activex. Los JavaBeans pueden ser combinados para crear aplicaciones tradicionales o sus pequeños equivalentes en orientación Web, applets.

Capítulo 3: Diseño y Desarrollo de sistemas portables.



Introducción.

En es este capitulo se habla de los temas Diseño de Salida, Diseño de Entrada, Diseño de la Interfaz de Usuario, Diseño de Archivos, Dispositivos de Almacenamiento, Ambientes en Línea y Distribuidos, Desarrollo de una solución y Pruebas.

El tema Diseño de Salida habla de las diferentes formas que se tienen para indicar una salida de un sistema las cuales son: impresión, despliegue y audio. También se habla de ¿cómo elegir una tecnología de salida?, del diseño del formato de salida (tanto para una salida de impresión como para una salida de despliegue) y de la implantación del diseño de la salida en la solución del problema.

El tema Diseño de Entrada habla sobre los datos de entrada, el diseño de formas, el diseño de pantallas, la validación de la entrada (verificación de la transacción y validación de los datos de la entrada) y de la implantación del diseño de entrada en la solución del problema.

El tema Diseño de la Interfaz de Usuario de mencionan los diferentes tipos de interfaces que hay (interfaz de lenguaje, interfaz de preguntas y respuestas, menús, formas de entrada/salida, interfaz de lenguajes de comandos, interfaz de manejo directo, el ratón y otras interfaces), de las diferentes retroalimentaciones para el usuario que existen (aceptación de entrada, entrada correcta, entrada incorrecta, retrazo en el proceso, petición terminada, petición incompleta y mayor retroalimentación o ayuda) y de la implantación del diseño de la interfaz de usuario en la solución del problema.

El tema Diseño de Archivos habla de los tipos de archivos (archivo maestro, archivo de tabla, archivo de transacciones, archivo de trabajo y archivos de impresión), de los métodos de organización de archivos (organización secuencial, organización de acceso directo, lista de enlace, organización indexada y organización de índices secuenciales) y de la implantación del diseño de archivos en la solución del problema.

El tema Dispositivos de Almacenamiento habla sobre las tecnologías de los dispositivos, las cuales son la tecnología magnética (discos magnéticos y cintas magnéticas) y la tecnología óptica (disco compacto). También se habla del dispositivo de almacenamiento que se utilizó en la solución del problema.

El tema Ambientes en Línea y Distribuidos explica lo que son sistemas en línea y sistemas distribuidos, y la implantación de ellos en la solución del problema.

El tema Desarrollo habla de las diferentes formas de diseño y desarrollo como son: diseño ascendente, diseño descendente, desarrollo modular. También habla del diagrama estructural, de las herramientas de documentación (diagramas HIPO, diagramas de flujo, diagramas Nassi-Schneiderman, diagramas Warnier/Orr, pseudo código y método folklore) y de la implantación del desarrollo en la solución del problema.

El tema Pruebas habla de los diferentes tipos de pruebas que hay, como son: prueba de programa con datos de prueba, prueba de enlace con datos de prueba, prueba del sistema completo con datos de prueba, prueba del sistema completo con datos reales y pruebas especiales de sistemas. También se habla de la realización de las pruebas en la solución desarrollada.

Diseño y Desarrollo de sistemas portables.

"El diseño es una solución; es decir, es la traducción de los requerimientos en formas que los satisfagan."[13]

Existen dos niveles de diseño:

- **Lógico:** Escribe las especificaciones detalladas del sistema, las cuales describen sus características (las salidas, entradas, archivos y bases de datos, y procedimientos.)
- **Físico:** Este diseño produce programas de software, archivos y un sistema en marcha.

El diseño lógico indica cómo hacerlo; y el diseño físico lo lleva a cabo.

Las especificaciones de diseño indican a los programadores qué debe hacer el sistema. Los programadores a su vez escriben los programas que aceptan entradas por parte de los usuarios, procesan los datos, producen los informes y almacenan estos datos en archivos.

Diseño de Salida.

La salida es la información que reciben los usuarios del sistema de información, de hecho es la característica más importante del sistema.

Antes de convertirse en una salida adecuada, ciertos datos requieren de un proceso extensivo, otros solo se almacenan y cuando se les solicita, se consideran salida con poco o nada de proceso.

La salida debe ser entendida y creada de una manera general, de tal forma que cualquier información producida por el sistema (y que sea útil para la gente) sea considerada como tal. Es posible entender la salida como cualquier cosa que sale de la organización, a la cual se le llamaría "salida externa", o que permanezca dentro de la organización, lo cual sería una "salida interna".

Los sistemas de información utilizan tres métodos principales para la salida, los cuales se explican a continuación.

Impresión.

La salida impresa es la más común de los tipos que existen. Las opciones de tecnología que se tienen para obtener este tipo de salida son las impresoras de impacto, las impresoras de no impacto, y equipos de cámaras especiales. Las opciones de salida que obtenemos al utilizar estas tecnologías son impresiones en papel, impresiones en formas especiales, y formas impresas para enviar por correo, de las impresoras de impacto y no impacto; y de las cámaras especiales, los microfilmes y las microfichas (una hoja de microfilmes).

[13] JAMES A. Senn. "Análisis y Diseño de Sistemas de Información."

Despliegue.

La salida de la computadora puede ser desplegada en una pantalla, dentro de las cuales tenemos la opción de escoger entre monitores (Tubo de Rayos Catódicos), pantallas de cristal líquido, o pantallas de plasma. Este tipo de salida es el más utilizado en los sistemas de información en línea ya que ofrece una salida inmediata y se puede crear una convivencia entre el sistema y los usuarios.

Audio.

Este método de salida se apoya en patrones digitales almacenados, los cuales producen sonidos. Los sonidos pueden amplificarse a través de un bafle o escucharse a través de las pequeñas bocinas de la microcomputadora.

La Tabla 3-1 muestra una comparación entre los tres tipos de salidas mencionados, realizando una división en el método de impresión entre impresoras y cámaras especiales.

Método	Ventajas	Desventajas
Impresión (impresora)	Disponible para la mayoría de las organizaciones. Gran flexibilidad en el tipo de salida. Capacidad para manejar grandes volúmenes de salida a bajo costo. Muy confiable con un mínimo de tiempos muertos.	Puede ser ruidosa. Problemas de compatibilidad con cierto software. Puede requerir suministros especiales y costosos. Requiere de la participación del usuario. Puede ser lenta dependiendo del modelo.
Impresión (Cámaras Especiales)	Manejo de grandes volúmenes de información. Reduce la necesidad de espacio de almacenamiento. Conserva materiales frágiles, pero de uso frecuente. Evita el problema de paginación en reportes voluminosos.	Requiere de software particular para poder contar con un sencillo acceso. Requiere de equipo especial para la impresión de copias. En un principio puede ser una inversión costosa.
Despliegue	Interactiva. Opera en línea, en tiempo real por medio de su enlace a una red de amplio alcance. Silenciosa. Toma ventaja de las facilidades del equipo de cómputo para movilizarse dentro de archivos y bases de datos. Adecuado para el envío de frecuente de mensajes	Requiere de cableado y espacio disponible. Todavía puede requerirse de una documentación impresa. Llega a ser costosa si se requiere para numerosos usuarios.

	transitorios.	
Audio	Adecuada para usuarios individuales. Adecuada para mensajes transitorios que requieran una acción inmediata, y dicho mensaje pueda descartarse. Ideal para aquellos trabajadores que requieren tener las manos libre. Conveniente si la salida es muy poco frecuente.	Su desarrollo es costoso. Requiere de un lugar específico para evitar que la salida no interfiera con otras actividades. Cuenta con aplicaciones limitadas.

Tabla 3- 1 Comparación de los métodos de salida.

¿Cómo elegir la tecnología de salida?

Existen varias consideraciones, que se pueden interpretar como principios útiles que permanecen constantes en relación a los avances tecnológicos, las cuales son:

¿Quién usará la salida? Es importante identificar quién utilizará la salida, ya que los requisitos del puesto, permitirán definir el método apropiado de salida.

¿Cuántas personas necesitan la salida? Si varias personas requieren la salida, se pueden justificar copias impresas. Si solamente la necesita un usuario, bastará con un despliegue en pantalla, o una señal de audio.

¿En dónde se necesita la salida? Aquella información que permanecerá cercana a su punto de origen puede ser impresa. Una información que deba transmitirse a usuarios a gran distancia, puede ser de maneja electrónica y el receptor decide si lo imprime o lo despliega.

¿Cuál es el propósito de la salida? Si la salida se entregará a los directivos de una empresa mensualmente, debe ser impresa. Si indicará las variaciones que existen en una variable del sistema cada quince minutos, bastará con ser desplegada.

¿Con que velocidad se requiere la salida? Cuando se requiere una salida inmediata, está puede ser desplegada. Por el contrario, si la salida no se necesita inmediatamente, está puede ser impresa.

¿Con que frecuencia se requiere la salida? Si la salida se requiere frecuentemente, se debe considerar una salida de despliegue. Si la salida no se requiere con una frecuencia alta, esta puede ser impresa en papel. Si la salida tiene una frecuencia de consulta muy baja, puede ser impresa en microfilmes.

¿Durante cuanto tiempo se almacenará la salida? Si se necesita que la salida sea almacenada por un largo periodo de tiempo, se recomienda que esta sea impresa en microfilmes. Si la salida se requiere por unos cuantos días, es suficiente tenerla impresa en papel. Si solamente se requiere la salida por unas horas o minutos, puede ser desplegada o de audio.

¿Bajo que regulaciones particulares se produce la salida? Esto es, si la salida necesita una característica especial, como ser impresa para hacer un deposito bancario, se un comprobante de pago, etc., o no.

¿Cuáles son los costos iniciales y posteriores de mantenimiento? En este punto se debe considerar los costos iniciales que se necesitaran para adquirir los elementos requeridos para entregar una salida de cierto tipo, así como, los costos de mantenimiento que tendrán esos elementos a través del tiempo.

¿Cuáles son los requisitos ambientales para las tecnologías de salida? En este punto se deben considerar cosas como: La salida de audio requiere un ambiente relativamente silencioso. Las impresoras requieren de un ambiente seco y fresco para poder operar correctamente. Los monitores requieren de espacio y cableado para poder conectarse a la computadora con que trabajan. Etc.

Al terminar de contestar las preguntas anteriores, se debe tener una idea y una visión de la salida o salidas que puede tener un sistema de información y así, se podrá elegir la salida más conveniente.

Diseño del formato de salida.

La elección del medio de salida más conveniente y la seguridad de calcularla en forma correcta no garantizan que los informes, documentos o pantallas de despliegue sean útiles para el usuario. El formato de salida se encarga de ello.

Un formato de salida es la colocación de diferentes aspectos en la salida impresa o en la pantalla de despliegue. Cuando se diseña un formato de salida, se debe contemplar como aparecerá después de que el sistema esté en operación. El formato de salida se revisara en forma repetida durante el diseño lógico y físico del sistema.

Diseño de la salida por impresión.

Se empieza el diseño determinando que aspectos se incluirán en el mismo. El análisis de requerimientos proporciona esta información, y el diccionario de datos contiene la información descriptiva necesaria: tipo de dato y su longitud. El análisis como el diccionario de datos son elementos que se revisaron el capitulo I.

Las partes de un informe impreso se pueden generalizar en tres grupos:

1. *Encabezados.* Deben de ir en la parte superior del informe y normalmente incluyen información como el titulo del informe, la fecha de impresión, titulo de las columnas de los datos mostrados, datos de la organización emisora, etc. Los encabezados deben repetirse en cada página que sea generada en el informe.
2. *Datos y Detalles.* Aquí es donde va la información que describe los datos obtenidos del sistema.
3. *Resúmenes.* En esta parte se coloca la información de resumen, como pueden ser los totales o subtotales de columnas, algunas notas para la interpretación del informe, etc., si estas existieran.

La Figura 3-1 muestra la ubicación de las partes en un informe.

Figura 3- 1 Partes de un informe.

También se puede englobar todas las características de un reporte impreso en dos bloques, los cuales son:

1. *Características Funcionales.* San aquellas que incluyen el encabezado o titulo del reporte, el número de la página, la fecha de preparación, los rótulos de las columnas, el agrupamiento de los datos relacionados y el uso de elementos de control.
2. *Características Estéticas.* Estas se refieren al uso de márgenes, de espacios en blanco adicionales para alinear el texto, el enmarcado de resultados, el uso de colores. Todas estas características se emplean para lograr que el reporte sea atractivo, tenga legibilidad, y sea útil para el usuario.

Diseño de salida por despliegue.

Estas son temporales, es decir, una imagen en un monitor no es permanente. Pueden estar dirigidas en forma más específica hacia el usuario. Tienen un formato con mayor

flexibilidad. No son portables. En algunas ocasiones, las salidas por pantalla pueden ser modificadas mediante una interacción directa con el usuario.

Los usuarios deben saber que teclas presionar si desean consultar pantallas adicionales, como concluir la presentación, y como interactuar con el desplegado de la pantalla. El acceso a las presentaciones por pantalla puede controlarse a través del uso de un código confidencial (contraseña).

Existen cuatro lineamientos que facilitan el diseño de pantallas, estos son:

1. *Mantener una pantalla sencilla.* La pantalla debe mostrar solamente la información de la acción que se llevó acabo. Una opción puede ser el uso de ventanas, las cuales cubrirán parcial o totalmente la pantalla activa con nueva información después de realizar alguna instrucción.
2. *Mantener una presentación consistente de la pantalla.* La consistencia de la pantalla se mantiene, si la información se localiza en la misma área cada vez que se realiza un acceso una nueva pantalla. La información que tenga alguna relación lógica entre si, debe presentarse en forma agrupada.
3. *Facilitar el movimiento del usuario entre pantallas.* Es la factibilidad de desplazarse con facilidad entre una pantalla y otra.
4. *Crear una pantalla atractiva.* Las pantallas deben atraer al usuario y mantener su atención, esto se puede lograr, utilizando áreas vacías que rodeen los datos a manera de que la pantalla no se vea sobrecargada, utilizando un flujo lógico en la organización de la información, y facilitando la ubicación en la pantalla.

Al igual que un informe, una pantalla se puede dividir en tres partes:

1. *Encabezados.* Esta es la parte superior de la pantalla, la cual debe indicar al usuario donde se encuentra dentro de la aplicación o paquete, o cualquier información relevante.
2. *Cuerpo.* Es la parte intermedia de la pantalla y muestra el detalle de los campos y los datos, o cualquier información resultante de alguna acción realizada.
3. *Comentarios e Instrucciones.* Esta corresponde a la parte inferior de pantalla, puede contener comentarios o instrucciones que ayuden a la interpretación o manejo de la pantalla en general.

La Figura 3-2 muestra las partes de una pantalla.

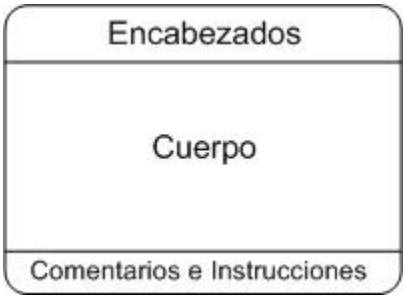


Figura 3- 2 Partes de una pantalla.

En el diseño de pantallas también se pueden considerar otros detalles como son: el uso de diferentes tipos de letras (los estilos diferentes logran la categorización de los datos), el uso de imágenes (las imágenes simbolizan ciertas acciones de computo que los usuarios pueden seleccionar con el ratón), el uso de color (el uso de color es una manera atractiva y comprobada para facilitar el uso de la computadora).

Implantación del diseño de salida para la solución del problema.

Tomando en cuenta las características del sistema para el cual se desarrolló la aplicación, las cuales se mencionaron en el primer capitulo, se obtuvo una salida por despliegue en pantalla.

Las respuestas a las preguntas para la selección de tecnología de salida quedaron de la siguiente manera:

¿Quién usará la salida? El usuario que tenga los privilegios suficientes en el sistema.
¿Cuántas personas necesitan la salida? Únicamente el usuario que la solicite.
¿En dónde se necesita la salida? En el cliente del sistema.
¿Cuál es el propósito de la salida? Entregar información sobre la aplicación de carga o descarga al usuario del sistema que las ejecute.
¿Con que velocidad se requiere la salida? En tiempo real.
¿Con que frecuencia se requiere la salida? Esporádica.
¿Durante cuanto tiempo se almacenará la salida? Por unos minutos.
¿Bajo que regulaciones particulares se produce la salida? No aplica.
¿Cuáles son los costos iniciales y posteriores de mantenimiento? Los costos iniciales y de mantenimiento no existen, ya que van implícitos con los requeridos por el sistema donde trabaja la aplicación.
¿Cuáles son los requisitos ambientales para las tecnologías de salida? No aplica.

Las pantallas de salida que presentan la Aplicación, tienen diferente contenido dependiendo de los resultados obtenidos en el proceso; sin embargo, todas las pantallas utilizan alguno de los dos patrones siguiente:

1. *Pantalla con mensaje.* Es una pantalla del sistema que despliega una ventana, la cual muestra un mensaje hacía el usuario. La Figura 3-3 muestra el patrón de esta pantalla.

Figura 3- 3 Patrón de pantalla con mensaje.

2. *Pantalla de descarga de archivos.* Es una pantalla que indica al usuario que se descargarán archivos. La Figura 3-4 muestra el patrón de esta pantalla.

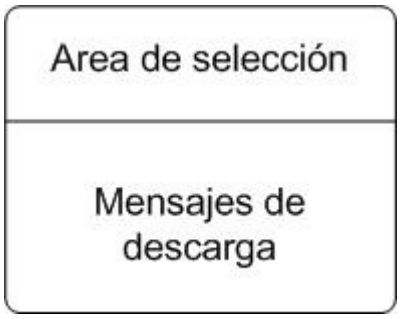


Figura 3- 4 Patrón de pantalla de descarga de archivos.

Diseño de Entrada.

"La entrada es la liga que une el sistema de información al mundo de los usuarios." [14]

"La calidad de la salida del sistema está predeterminada por la calidad de su acceso o entrada" [15]

El diseño de la entrada consiste en desarrollar especificaciones y procedimientos para la preparación de datos, aquellos pasos necesarios para poner los datos de la transacción en una forma utilizable para su procesamiento, e introducción de datos, es someter los datos a

[14] JAMES A. Senn. "Análisis y Diseño de Sistemas de Información."
[15] KENDALL Kenneth E, KENDALL Julie E. "Análisis y Diseño de Sistemas"

la computadora para su procesamiento. La entrada de datos puede llevarse a cabo instruyendo a la computadora para que lea los datos de un documento o archivo, o puede ocurrir cuando el usuario teclee los datos directamente al sistema.

Un buen diseño de los formatos y las pantallas de entrada debe satisfacer los siguientes objetivos:

Eficacia. Significa que las formas y las pantallas de entrada satisfagan propósitos específicos del sistema.

Precisión. Se refiere a un diseño tal que asegure una realización satisfactoria.

Facilidad de uso. Esto significa que las formas y las pantallas serán explicitas y no requerirán de tiempo adicional para descifrarse.

Consistencia. Significa que las formas y las pantallas ordenen los datos de manera similar de una aplicación a otra.

Sencillez. Se refiere a mantener en un mínimo los elementos indispensables que centren la atención del usuario.

Atracción. Esto implica que el usuario disfrutará del uso o transito a través de las formas y las pantallas.

Datos de Entrada.

Existen dos tipos de datos que deben ser entrada cuando se procesan las transacciones en un sistema, estos tipos son:

1. *Datos variables.* Son aquellos datos que cambian con cada transacción que se hace, ejecución que se realiza, o decisión que se lleva acabo en el sistema.
2. *Datos de identificación.* Son los datos que identifican a lo que se esta procesando en el sistema.

De igual manera, también es muy importante lo que no se debe introducir al sistema para que este pueda funcionar. Entre las cosas que se pueden enumerar están:

1. *Datos constantes.* Son los datos que permanecen igual para cada entrada o ejecución que se realice en el sistema.
2. *Detalles que el sistema pueda consultar.* Don datos que se encuentran almacenados en algún lugar, y que el sistema pueda consultar.
3. *Detalles que el sistema pueda calcular.* Son resultados que pueden ser calculados por el sistema a partir de los datos almacenados e introducidos.

Diseño de las formas.

Las formas son los documentos fuentes en los cuales se captan inicialmente los datos; también, hacen surgir la información que los miembros de la organización requieren.

En el diseño de formas se deben observar cuatro lineamientos con el fin de lograr que sean útiles. Estos lineamientos son:

1. *Diseñar formas fáciles de llenar.* Este lineamiento se basa en lograr una empatía muy grande con el usuario, con el fin de reducir el error, agilizar su llenado y facilitar la captura de datos.
2. *Cubrir el propósito para el cual se diseñan.* Las formas se deben crear para satisfacer uno o más de los objetivos de registro, proceso, almacenamiento o consulta de la información. En ocasiones, cuando se comparte cierta información básica es deseable proporcionar información diferente a distintos departamentos o usuarios.
3. *Asegurar un llenado preciso.* El diseño es importante para que la gente se percate de la manera correcta de llenado, sin importar que sea la primera o la décima vez que la utiliza.
4. *Mantener las formas atractivas.* Las formas estéticas motivan a la gente y hacen que se les de importancia, se sentirá más satisfecha y además llenará las formas en toda su extensión. Se les debe de dar una apariencia de organización y lógica.

Existe la posibilidad de que los datos sean introducidos a través de una estación de trabajo o de una terminal inteligente. Si se utilizan terminales inteligentes para captación de datos, puede eliminarse la necesidad de utilizar documentos fuentes, a menos que se requiera también un registro en papel de la transacción.

Diseño de pantallas.

Lo que se ha mencionado para el diseño de formas, puede aplicarse al diseño de pantallas; sin embargo, existen cuatro lineamientos para el diseño de pantallas, estos son: Mantener una pantalla sencilla. Mantener una presentación consistente de la pantalla. Facilitar el movimiento del usuario entre pantallas. Crear una pantalla atractiva. Estos lineamientos fueron explicados en la sección “Diseño de salida por despliegue” en este capítulo.

Validación de la entrada.

Los diseños de entrada se enfocan para reducir la posibilidad de errores, estos errores se deben encontrar durante la entrada y corregir antes de que los datos sean almacenados o procesados. El término que se les da en general a los métodos que se encargan de la detección de errores en la entrada es *validación de entrada.*

Existen dos métodos para realizar la validación de la entrada, estos son: Verificación de la transacción y Validación de los datos de la entrada.

Verificación de la transacción.

La verificación de la transacción se da en buena medida a través del software, y consta de varias partes, las cuales son:

Validación de la transacción. Identifica cualquier transacción que no sea válida. Las transacciones pueden ser inválidas debido a se introducen datos incorrectos, que están incompletas, sin autorización e incluso desordenadas.

Pruebas de secuencia. Las pruebas de secuencia utilizan los datos para probar una de dos condiciones diferentes: el orden de las transacciones, es cuando las transacciones necesitan procesarse en cierto orden, y aspectos faltantes, es cuando se ha omitido alguna transacción que es esperada.

Prueba de llenado completo. Esta se refiere a omitir datos para producir una información incompleta y errónea para almacenarse. Si se detecta, se podrá evitar errores causados por datos requeridos.

Validación de los datos de la entrada.

Las transacciones válidas pueden contener datos inválidos; por eso se debe asegurar, en el diseño del sistema, la especificación de métodos que validen los datos cuando son introducidos.

Existen varios métodos que permiten dicha validación:

Prueba de existencia. Las pruebas de existencia examinan los campos esenciales para verificar que contengan datos.

Prueba de límite o rango. Las pruebas de límite validan la cantidad mínima o máxima aceptable para un dato.

Prueba de combinación. La prueba de combinación valida que algunos campos de datos tengan valores aceptables en conjunto, esto es, el valor para un elemento dato determina si otros valores de datos son correctos.

Prueba de evaluación del tipo o composición. Esta prueba se encarga de verificar que los datos proporcionados se integren exclusivamente de caracteres válidos.

Prueba de longitud. La prueba de longitud se encarga de verificar que la entrada cuente con una longitud de campo correcta.

Implantación del diseño de entrada en la solución del problema.

Para la implantación de la entrada en la solución del problema, se consideraron las características mencionadas en el primer capítulo de este trabajo, las cuales nos indican que la entrada la Aplicación debía ser por pantalla. La Aplicación puede recibir información de entrada de dos maneras diferentes:

1. *Por una pantalla de selección de datos.* Es una pantalla que le muestra al usuario las posibilidades de información que tiene para seleccionar. La opción seleccionada indicará a la Aplicación cual es la información que se debe procesar. La Figura 3-5 muestra éste patrón de pantalla.

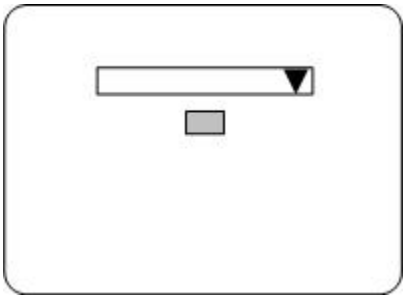


Figura 3- 5 Patrón de una pantalla de selección de datos.

2. *Por una pantalla de introducción de datos.* Es una pantalla donde se escribirá la información necesaria para la búsqueda de archivos, los cuales serán procesados de acuerdo a la petición solicitada. La Figura 3-6 muestra el patrón de la pantalla.

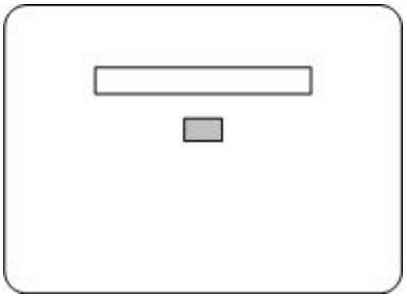


Figura 3- 6 Patrón de una pantalla de introducción de datos.

Una vez que la información de la entrada ha sido ingresada, la Aplicación automáticamente revisa que existan y valida los datos necesarios para el funcionamiento, entre esos datos tenemos:

Nombre de empresa. Es un dato de identificación el cual indicará la organización de la cual se necesita obtener la información, puede contener el valor de solamente una o de todas las empresas.

Nombre del archivo de información de salida. Este dato es un dato variable, y es propuesto por el sistema cada vez que se ejecuta la Aplicación, sin embargo el usuario lo puede modificar. Se refiere al nombre con el cual se generará el archivo que contenga la información solicitada.

Ruta de la carpeta de archivos para carga. Es un dato variable que indica la ruta de la carpeta que contiene los archivos que se cargarán a la BD del sistema a través de la Aplicación.
Nombre de los archivos de datos. Son datos constantes, que indican los nombres de los archivos que contienen la información que es generada y/o requerida por la Aplicación.

Datos de conexión a BD. Estos son datos constantes, provenientes de la configuración del sistema, que indican la información sobre la BD con la cual interactuará la Aplicación.

A todos estos datos se les realiza una prueba de existencia, ya que el hecho de la falta de ellos ocasiona que la Aplicación falle.

También se valida la transacción a través de la verificación de los permisos necesarios para la lectura y escritura de los archivos requeridos, y en su caso la existencia de los mismos archivos.

Cualquier falló encontrado en la validación de la entrada, tanto de los datos como de la transacción, se notifica al usuario.

Diseño de la Interfaz de usuario.

La interfaz cuenta con dos componentes principales: el lenguaje de presentación, que es parte de la relación computadora-hombre y el lenguaje de acción que caracteriza la parte hombre-computadora. Ambos conceptos cubren la fórmula y el contenido del término interfaz de usuario.

Tipos de interfaces.

Existen diferentes tipos de interfaces para el usuario.

Interfaz de lenguaje

La interfaz de lenguaje natural permite la interacción con la computadora por medio del lenguaje común o natural.

Los problemas de implantación y la demanda extraordinaria de recursos de cómputo han limitado el desarrollo de las interfaces de lenguaje natural.

Interfaz de preguntas y respuestas.

En este tipo de interfaces la computadora plantea sobre la pantalla una pregunta al usuario, el usuario por lo general proporciona una respuesta (a través del teclado) y la computadora responderá con base en tal información de entrada de una manera preprogramada; después de ello, el cursor se desplaza a la siguiente pregunta.

Menús.

Una interfaz de menú permite que el usuario elija las posibles opciones de una lista en pantalla.

Al responder el usuario al menú, se limita a las opciones que se le presentan. El usuario no necesita conocer el sistema pero sí necesita saber que tareas pueden realizarse.

Puede tener acceso a los menús a través del teclado, de ratones o de lápices ópticos.

Formas de entrada/salida.

Las formas de entrada/salida son formas en pantalla que despliegan campos que contienen datos o parámetros que requieren ser comunicados al usuario.

Los menús y las líneas de comandos pueden programarse de tal forma, que los usuarios identifiquen la aplicación deseada aún antes de aparecer la forma.

Las formas en pantalla muestran la información que deberá introducirse, así como su ubicación.

Interfaz en lenguaje de comandos.

La interfaz del lenguaje de comandos permite que el usuario tome el control de la aplicación mediante una serie de teclas, comandos, frases o cierta secuencia de los mismos.

Interfaz de manejo directo.

El manejo directo le confiere al usuario una forma gráfica. Permite el manejo directo de la representación gráfica en pantalla, el cual puede realizarse mediante el acceso por teclado, ratón o palanca. El manejo directo requiere de una mayor sofisticación del sistema.

El ratón.

El ratón no es en sí una interfaz común. Su característica principal radica en que permite que el usuario opere partes del sistema de cómputo de manera dinámica.

El ratón puede ser extremadamente eficaz para aplicaciones particulares. Aunque no sustituye la entrada de texto por el teclado, el ratón es un dispositivo de entrada bastante adecuado para implantar interfaces de manipulación directa que requieren de una presentación gráfica.

Otras interfaces del usuario.

El dispositivo de señalamiento (con frecuencia el lápiz óptico) se utiliza para señalar sobre la pantalla un elemento de la misma.

Las pantallas sensibles al tacto permiten que el usuario active la pantalla mediante un dedo (u objeto), cuando se acerca a la superficie de la pantalla.

El reconocimiento de voz es muy atractivo, pues intenta semejarse a la comunicación humana.

Retroalimentación para el usuario.

Todos los sistemas requieren de la retroalimentación con el fin de supervisar y modificar la conducta presente con las metas preestablecidas y devuelve información que describe el distanciamiento existente entre el desempeño real y el que se intenta.

La retroalimentación para el usuario del sistema es necesaria. Aquella retroalimentación que es escasa o inoportuna, carece de utilidad, ya que solo podemos procesar una cantidad limitada de información.

Existen diferentes situaciones en un sistema en las cuales es necesaria la retroalimentación.

Aceptación de la entrada.

Es cuando se necesita saber que la computadora aceptó la entrada. Por ejemplo, cuando un usuario captura un nombre, el cursor de la computadora avanza un carácter cada vez que se incorporan de manera correcta las letras.

Entrada correcta.

Entender que la entrada cuenta con la forma correcta, como ejemplo se puede tener un usuario que ejecuta un comando y la computadora lo retroalimenta con “LISTO”, conforme avanza al siguiente punto.

Entrada incorrecta.

Advertir al usuario que la entrada carece de la forma correcta. Cuando los datos son incorrectos, una forma de informarle al usuario es presentar una ventana que describa con brevedad el problema de la entrada y la manera en que el usuario puede corregirlo.

Retrazo en el proceso.

Consiste en informar al usuario que habrá un retrazo asociado al procedimiento de su solicitud. Los retazos mayores a 10 segundos o más, requieren de la retroalimentación para que el usuario se percate de que el sistema se encuentra en fase de proceso.

Petición terminada.

Los usuarios necesitan saber cuándo fue llevada a cabo por completo su petición para poder hacer nuevas peticiones. Una forma puede ser presentar el mensaje “EJECUTADO” al finalizar una petición.

Petición incompleta.

Indicar al usuario que la computadora es incapaz de llevar a cabo por completo una petición. Para esta situación al usuario, el sistema le puede presentar un mensaje como “PROCESO NO TERMINADO, INTENTE DE NUEVO”.

Mayor retroalimentación o ayuda.

Los usuarios necesitan asegurarse de que se dispone de una mayor retroalimentación, y enterarse de la manera de obtenerla. Una forma muy común es presentar un mensaje que indique la manera de obtener la ayuda sobre el sistema.

Planificar la manera de proporcionar la retroalimentación a los usuarios, de tal forma que ellos se percaten de su entrada fue aceptada; su entrada se encuentra en forma correcta; si el proceso se esta llevando a cabo, si las peticiones se pueden procesar o no, y si requiere de mayor información detallada, así como la manera de obtenerla, ayudan a que el sistema se bien aceptado por los usuarios.

Implantación del diseño de la Interfaz de usuario en la solución del problema.

La Aplicación al ser desarrollada para un sistema Web, utiliza por fuerza interfaces de usuario. Las interfaces que utiliza son la de “Menús”, la de “Formas de Entrada/Salida”, y el “Ratón”.

La interfaz de Menús se utiliza para ir seleccionando el tipo de proceso que se necesita que realice la Aplicación. También se utiliza para indicar a la aplicación cual es la información que se requiere sea procesada en la descarga.

La interfaz de Formas de Entrada/Salida se utiliza para que el usuario ingrese la información referente a la ubicación donde se realizará la entrega de los datos después del proceso de descarga, la ubicación de los archivos necesarios para realizar el proceso de carga, y para informarle al usuario de cualquier resultado que genere la Aplicación.

El ratón debe estar presente en todas interfaces de la Aplicación, ya que se emplea para ir seleccionando las diferentes opciones que se presentan, y también para indicar cuando se deben ejecutar los procesos solicitados.

Las Imágenes de las interfaces que se generaron para la Aplicación, la cual es motivo de este trabajo, se encuentran a continuación.

Imagen 3- 1 Menú de selección de tipo de proceso.

Imagen 3- 2 Menú de opciones para selección de información a descargar.

Imagen 3- 3 Menú con selección de información a descargar.

Imagen 3- 4 Forma para seleccionar la entrega de datos de la descarga.

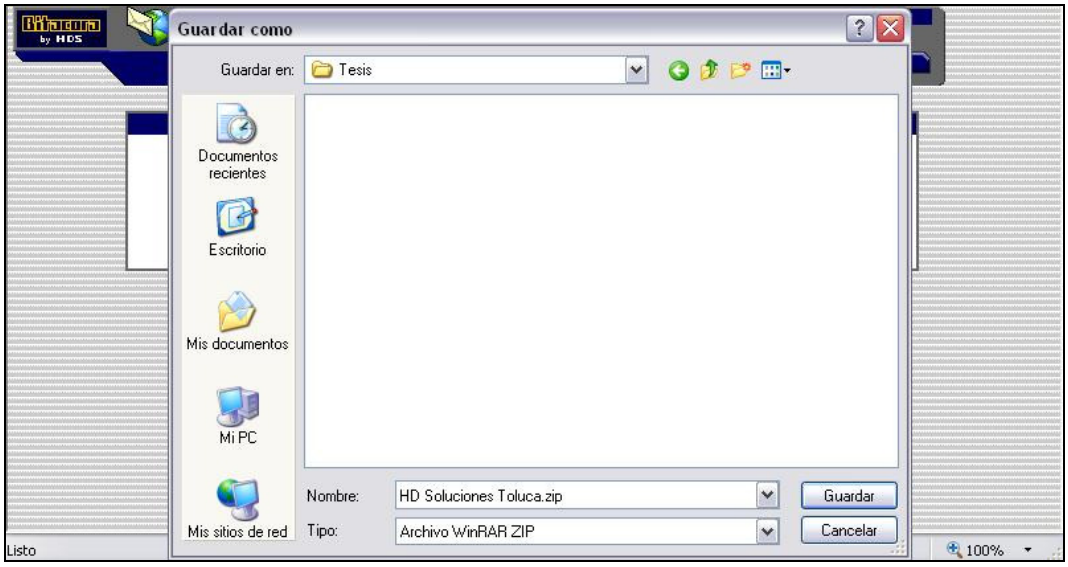


Imagen 3- 5 Forma para guardar la entrega de datos de la descarga.

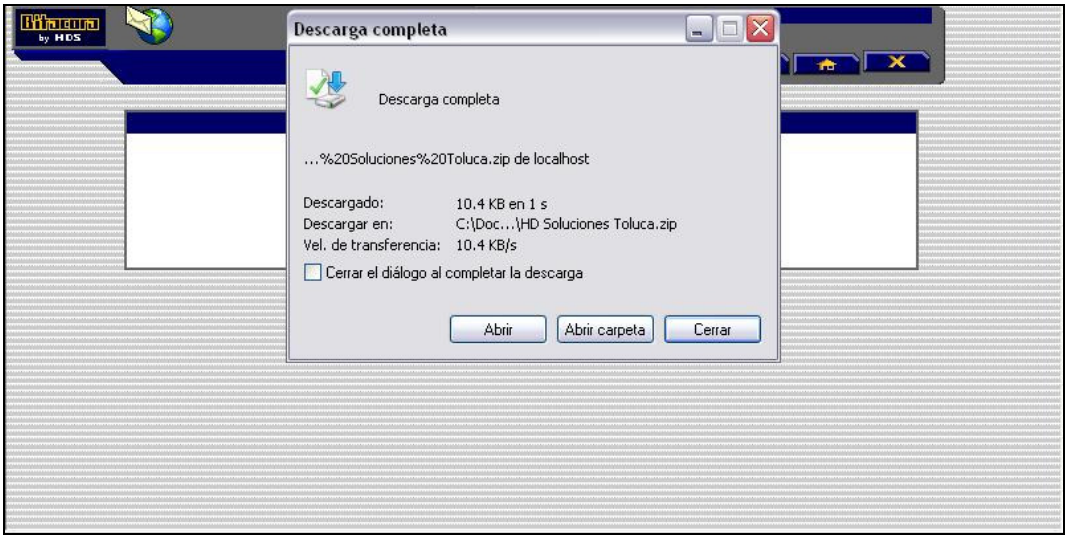


Imagen 3- 6 Forma para mostrar la finalización de la descarga de datos.

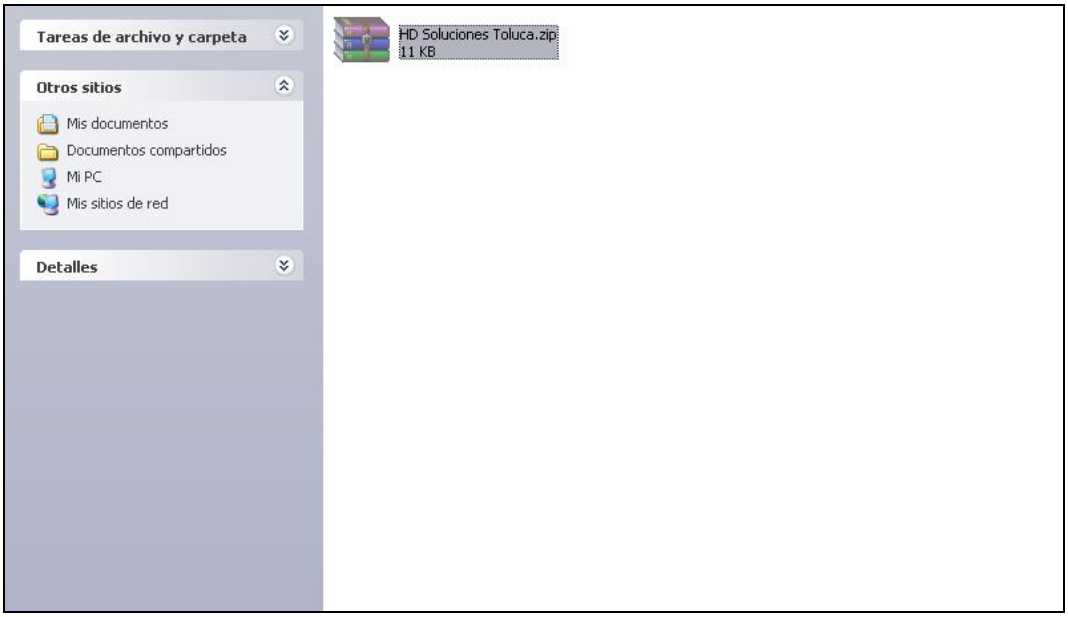


Imagen 3- 7 Datos entregados de una descarga de datos.

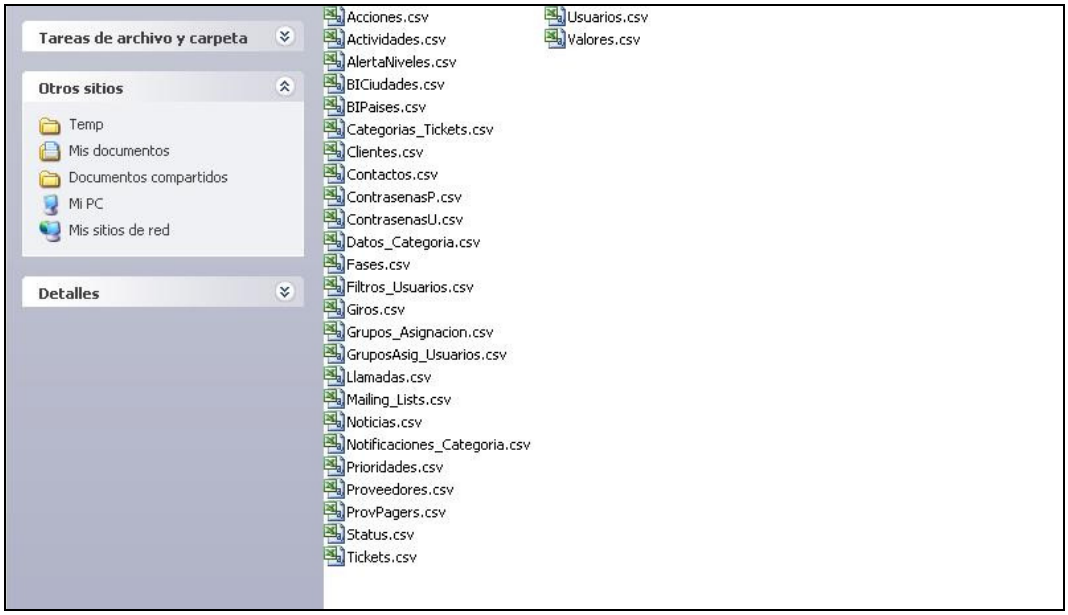


Imagen 3- 8 Archivos que serán usados en una carga de datos.

Imagen 3- 9 Forma para ingresar la ruta de los archivos de carga.

Imagen 3- 10 Forma para mostrar al usuario los resultados de la ruta nula de los archivos de carga.

Imagen 3- 11 Forma con entrada de ruta no nula.

Imagen 3- 12 Forma con resultados de falta de archivos necesarios.

Imagen 3- 13 Forma que indica al usuario que el proceso no termino de forma adecuada.

Imagen 3- 14 Forma que indica al usuario que el proceso termino correctamente.

Diseño de Archivos.

En un sistema de información basado en computadora se cuenta con dos enfoques para el almacenamiento de los datos. El primer método consiste en almacenar los datos en archivos individuales (El cual se tratará en esta sección), exclusivos para una aplicación en particular, y el segundo que involucra la elaboración de una Base de Datos.

Un archivo contiene grupos de registros que se utilizan para proporcionar información para operaciones, planeación, administración y toma de decisiones.

Tipos de archivos.

Los archivos pueden utilizarse para almacenar datos por un periodo indefinido de tiempo, o bien, pueden utilizarse como almacenes temporales para un propósito en particular. Existen cinco tipos principales de archivos: el archivo maestro y el archivo de tabla (ambos utilizados para almacenar datos por periodos de tiempo largos), el archivo de transacciones, el archivo de trabajo, y el archivo de impresión (Estos tres son archivos temporales). Estos se explican a continuación.

Archivo maestro.

Un archivo maestro es una colección de registros sobre un aspecto importante de las actividades de una compañía. Puede contener datos que describen el estado actual de acontecimientos específicos o de indicadores de negocios. Los atributos de los registros pueden actualizarse, pero los registros en si, se mantienen permanentes.

Archivo de tabla.

El archivo de tabla es un archivo permanente que contiene datos de referencia utilizados cuando las transacciones se procesan, se actualizan archivos maestros o se produce salida, es decir, contiene datos que se utilizan para calcular otros datos o más parámetros de desempeño. Como su nombre lo indica, estos archivos almacenan tablas de datos de referencia.

Archivo de transacciones.

Un archivo de transacciones es un archivo temporal que tiene dos objetivos: acumular datos sobre los acontecimientos conforme ocurren, y actualizar los archivos maestros para reflejar los resultados de las transacciones actuales. Una vez que se actualiza un archivo maestro, puede descartarse el archivo de transacciones. El termino transacción se refiere a cualquier suceso del negocio que afecte a la empresa y sobre el cual se captan datos.

Archivo de trabajo.

En ocasiones, un programa puede ejecutarse eficientemente si utiliza un archivo de trabajo; es decir, un archivo de trabajo se crea y manipula para ayudar al desempeño de una aplicación, este se destruye al terminar de utilizarlo.

Archivos de impresión.

El archivo de impresión almacena el contenido recolectado de los informes individuales de salida o documentos producidos por el sistema, los crea el mismo sistema donde se producen muchos informes, pero no hay tiempo de impresión o no se cuenta con una impresora en línea.

Métodos de organización de Archivos.

Los registros se almacenan en los archivos utilizando una organización que determina como se emplea el almacenamiento y como se ubicarán y consultarán los registros.

Organización secuencial.

En este tipo de organización, los registros se almacenan uno después de otro, sin importar el valor real de los datos en los registros. El primer registro almacenado se coloca al principio del archivo, el segundo se almacena a continuación, y así sucesivamente. Los registros se ordenan en base a un parámetro del registro.

Cuando se actualiza, el acceso a él es necesario hacerlo de manera completa. Los registros no pueden insertarse en la parte intermedia del archivo, generalmente se copia durante el proceso de actualización.

Para leer un archivo secuencial, el sistema siempre comienza al principio del archivo. Si el registro que se busca está en algún lugar dentro del archivo, el sistema lee todo el archivo hasta encontrarlo, de registro en registro.

Los archivos secuénciales se utilizan cuando lo requiere el hardware o cuando el acceso normal se requiere sobre la mayoría de los registros. Cuando se necesite leer o actualizar unos cuantos registros, será ineficiente utilizar un archivo secuencial, pero cuando se necesite leer o modificar numerosos registros, tendría mayor sentido una organización secuencial.

Organización de acceso directo.

Este método requiere que el programa le diga al sistema dónde se almacena un registro, de manera que pueda buscarlo ahí, es decir, que le indique la dirección.

Los archivos de acceso directo son archivos con llave. Asocian un registro con un valor de llave específico y una ubicación de almacenamiento en particular.

En contraste con la organización secuencial, el procesamiento de un archivo de acceso directo no requiere que el programa comience por el primer registro del archivo.

El tipo del acceso directo que utiliza la llave de registro como la dirección del almacenamiento se llama *direccionamiento directo*.

Otro tipo de acceso directo es el *direccionamiento algorítmico o atomizado*, este método se refiere al proceso (atomización o dispersión) de calcular una dirección de almacenamiento a partir de una llave de registro. Se emplea un algoritmo para cambiar el valor de la llave a otro valor que sirve como dirección para su almacenamiento. Existen muchas técnicas de dispersión.

Lista de enlace.

Los registros pueden ordenarse de manera lógica, en lugar de física, utilizando listas de enlace. Las listas de enlace se consultan para utilizar un conjunto de apuntadores que dirigen al siguiente registro lógico localizado en cualquier parte del archivo. Los apuntadores se almacenan en el mismo archivo que los datos.

Organización indexada.

Un índice es diferente a un apuntador, en el sentido de que se almacena en archivo independiente del archivo de datos al cual pertenece. En el archivo de índices, cada registro contiene dos datos: la llave del registro y una dirección del almacenamiento.

Para encontrar un registro específico, el índice primero se rastrea para encontrar la llave del registro que se desea. Cuando se encuentra, entonces el programa da acceso al registro directamente. El uso de archivos de índices permite que el rastreo de un registro sea mucho

más rápido, ya que lleva menos tiempo buscar en un índice que en un archivo de datos en su totalidad.

Organización de índices secuénciales.

Un método ampliamente utilizado para la organización de archivos es el denominado organización de índices secuénciales, o método de acceso secuencial indexado (ISAM, por sus siglas en ingles, Indexed Secuential Access Method). En un archivo ISAM, los registros se ordenan en bloque. Los registros dentro de los bloques se almacenan en un orden físico, pero los bloques o registros pueden presentarse en cualquier orden. Además, se necesita un índice para localizar el bloque del registro.

Implantación del diseño de archivos en la solución del problema.

Una de las características, que se mencionó en el capítulo I de este trabajo, que cumple la Aplicación que se diseño es la generación de archivos de información, los cuales contienen los datos obtenidos de la Base de Datos.

Para satisfacer este requerimiento, se diseñaron archivos de trabajo. La selección del tipo de archivo a utilizar se basó en la definición para cada tipo, con la cual podemos decir que, para la Aplicación que se desarrolló (en la parte de la descarga), es suficiente utilizar los archivos de trabajo.

La organización de los registros dentro de los archivos de trabajo es de un tipo secuencial. Esta decisión también se tomó en base a las definiciones, dadas en la sección anterior, para cada tipo de organización posible.

Los archivos se van creando conforme se vaya obteniendo la información de la Base de Datos. Para cada tabla de la BD que se lea, se crea un archivo único, a excepción de la tabla que contiene las contraseñas, la cual generará dos archivos. La Figura 3-7 muestra un ejemplo de la correspondencia tabla-archivo.

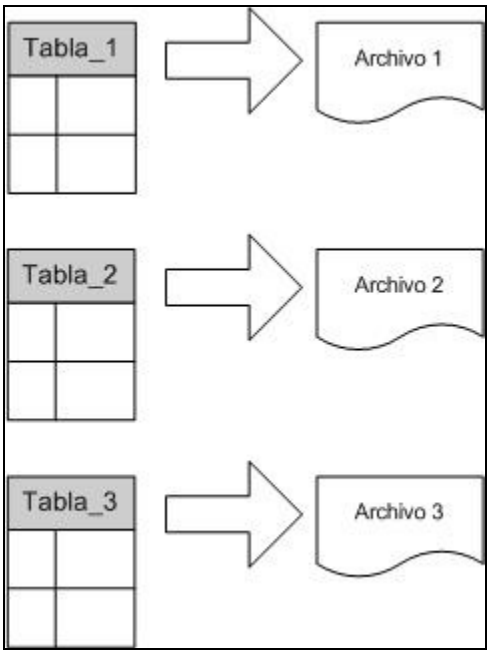


Figura 3- 7 Ejemplo de la correspondencia tabla-archivo.

Como ya se mencionó, los archivos tienen una organización secuencial ya que cuando la Aplicación los crea, en la parte de la descarga, los registros que contienen los datos se almacenan conforme se obtienen de la Base de Datos; esto garantiza, que los registros tienen un orden físico por alguno de sus campos.

En lo que se refiere a la parte de la carga, la organización secuencial de los archivos también cumple con las características necesarias para la Aplicación, ya que se lee cada uno de los registros almacenados en los archivos (sin excepción alguna) hasta completar la totalidad del mismo archivo.

En la Figura 3-8 se puede observar un ejemplo de la organización que tienen los archivos creados por la Aplicación.

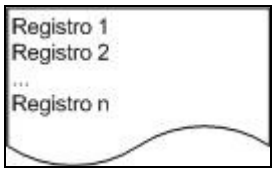


Figura 3- 8 Ejemplo de la organización de los archivos.

Dispositivos de Almacenamiento.

Tecnología magnética.

Esta tecnología se basa en la histéresis[16] magnética de algunos materiales y otros fenómenos magnéticos. Consiste en la aplicación de campos magnéticos a ciertos materiales cuyas partículas reaccionan a esa influencia, generalmente orientándose en unas determinadas posiciones que conservan tras dejar de aplicarse el campo magnético. Esas posiciones representan los datos.

La cinta magnética y el disco magnético, son los dispositivos más comunes de almacenamiento de esta tecnología.

Discos magnéticos.

Los discos magnéticos son los medios más comunes de almacenamiento, ya que se encuentran en casi todos los equipos de cómputo, y son utilizados por la mayoría de los sistemas para realizar el almacenamiento requerido. Los discos magnéticos son dispositivos de acceso directo, lo que significa que un registro puede ser escrito o leído en una localidad específica sobre el disco, sin necesidad de leerlo todo. Los discos magnéticos están divididos en dos categorías, generalmente llamadas como discos duros y discos flexibles.

Cintas magnéticas.

La cinta magnética es uno de los medios mejor conocidos y menos caros para almacenar los datos en los sistemas de cualquier tamaño. Para encontrar un registro, se debe leer toda la cinta hasta localizarlo. Una de las ventajas, por así llamarlo, que ofrece la cinta magnética sobre el disco magnético, es la mayor capacidad de almacenamiento de datos en un solo dispositivo, y la portabilidad que ésta ofrece.

Tecnología óptica.

Esta tecnología utiliza las propiedades del láser y su alta precisión para leer o escribir los datos. Los fundamentos técnicos que se utilizan son: un haz láser va leyendo (o escribiendo) microscópicos agujeros en la superficie de un disco de material plástico, recubiertos a su vez por una capa transparente para su protección del polvo.

El disco compacto (CD) es el dispositivo, con esta tecnología, que más se utiliza para almacenar datos.

[16] La histéresis es la propiedad de un material a conservar una de sus propiedades, en ausencia del estímulo que la ha generado.

Disco compacto.

Su primera aplicación comercial masiva fue el CD de música, esté guarda la información en formato digital (unos y ceros). La principal característica del dispositivo es su fiabilidad. No les afectan los campos magnéticos, apenas les afectan la humedad ni el calor y pueden aguantar golpes importantes.

Dispositivos de almacenamiento en la solución del problema.

El dispositivo de almacenamiento que se utilizó en la solución del problema es el disco magnético, en la categoría de disco duro, ya que su presencia en casi todas las computadoras facilita su empleo; además, de que actualmente cualquier sistema operativo, por lo menos, maneja un acceso al disco duro. Este dispositivo es empleado tanto por la parte de la carga como por la parte de la descarga, donde la primera escribe en el disco, y la segunda lee de él. La Figura 3-9 muestra un diagrama que señala la interrelación de la Aplicación con el disco duro.

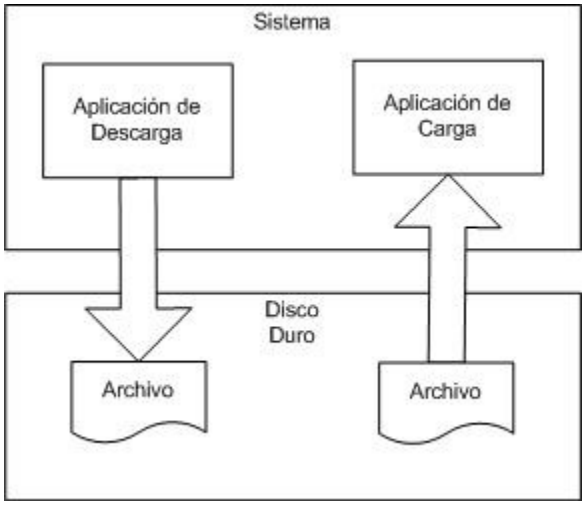


Figura 3- 9 Interrelación de la Aplicación con el disco duro.

Por otra parte, los permisos necesarios para el acceso de escritura y el acceso de lectura, son manejados por el sistema operativo.

El espacio requerido dentro del disco depende de la cantidad de información que se maneja, y esta cantidad solamente se determina en el momento que se ejecuta la Aplicación.

Ambientes en línea y distribuidos.

Los sistemas de información se están desarrollando en forma creciente para recibir y procesar datos utilizando métodos en línea.

Sistemas en línea.

Los sistemas en línea son los sistemas de información que permiten al usuario interactuar directamente con la computadora, mientras se lleva a cabo el procesamiento.

Los sistemas interactivos unen el sistema de cómputo con el usuario y permiten la introducción de datos, solicitudes de información o de procesamiento con retroalimentación directa de los resultados o respuestas para el usuario. En cierto sentido, la computadora y el usuario mantienen una comunicación en donde el individuo introduce un comando para la computadora y el sistema le responde con una solicitud para una clarificación adicional, ejecuta ésta o envía un mensaje. El dialogo interactivo define que hará el sistema. Muchos sistemas utilizan también un menú de alternativas.

Una de las ventajas para utilizar un sistema en línea es que proporcionan las respuestas rápidas que se necesitan con objeto de cumplir los requerimientos del usuario. Otra ventaja es la posibilidad de captar datos conforme los sucesos se presentan. Una tercera ventaja es que algunos sistemas en línea permiten a muchos usuarios utilizar el sistema al mismo tiempo.

Sistemas distribuidos.

Un sistema distribuido interconecta localidades que tienen capacidades de cómputo para captar y almacenar datos, procesarlos y enviarlos junto con la información a otros sistemas.

La característica principal de un sistema distribuido es, como su nombre lo indica, distribuir el trabajo en varios equipos para que este se realice con mayor rapidez. Se puede decir que es muy similar al concepto de *"Capas Cliente-Servidor"* comentado en el capitulo II de este trabajo; con la diferencia, que el concepto de *"Capas Cliente-Servidor"* se refiere a la parte lógica (software), y el concepto *"Sistema distribuido"* a la parte física (hardware) de un sistema.

Implantación en la solución del problema.

Tomando en cuenta la descripción realizada en la sección anterior, referente a un sistema en línea, y los requerimientos que se mencionan en el primer capitulo sobre el desarrollo que se realizó en este trabajo de tesis, podemos decir que la Aplicación se considerada dentro de esta categoría para su diseño; por esta razón, la Aplicación cubre todas las características correspondientes a un sistema en línea.

De igual forma la Aplicación puede ser considerada dentro de la categoría de un sistema distribuido, ya que las características del sistema para el cual se desarrolló, que se mencionan en el primer capitulo, y los requerimientos que está cubrió, hacen que cumpla con las características de dicha categoría.

La Figura 3-10 muestra un diagrama de la Aplicación interactuando con el usuario, y porque es catalogada como un sistema en línea y un sistema distribuido.

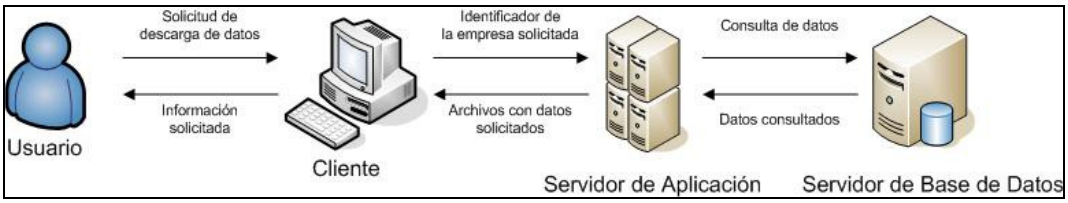


Figura 3- 10 Interrelación de la Aplicación con el usuario.

Desarrollo.

Existen varias formas de diseño y desarrollo de sistemas, entre la que se pueden mencionar el diseño ascendente, y el diseño descendente. También se puede mencionar el enfoque modular de la programación.

Diseño ascendente (bottom-up).

El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse conforme van apareciendo, su análisis como sistemas y su codificación. Los problemas que requieren computarizar, con mayor frecuencia se encuentran en los niveles inferiores de la organización, por ello en principio son los únicos en los cuales el cómputo puede ser costeable. En consecuencia, este enfoque se denomina ascendente, refiriéndose a que la computarización se implanta desde el nivel más bajo.

En un enfoque ascendente, es difícil integrar los subsistemas a grado tal de que el desempeño global sea fluido. Los problemas de interacción entre los sistemas son sumamente costosos y muchos de ellos no se solucionan. Cada subsistema parece ofrecer lo que se requiere, cuando se contempla al sistema como una entidad global, adolece de ciertas limitaciones por haber tomado un enfoque ascendente. Tal vez el más serio inconveniente del enfoque ascendente, es que los objetivos globales de la organización no fueron considerados, y en consecuencia no se satisfacen.

Diseño descendente (top-down).

El diseño descendente implica observar la gran imagen del sistema, con lo cual se debe cubrir los objetivos globales de la organización, y luego explosionarlo o desglosarlo en partes más pequeñas o subsistemas con sus requerimientos; es decir, comenzar con los niveles generales para obtener un entendimiento del sistema e ir descendiendo gradualmente a niveles de mayor detalle.

La Figura 3-11 muestra el enfoque descendente para alcanzar los objetivos globales de la organización.

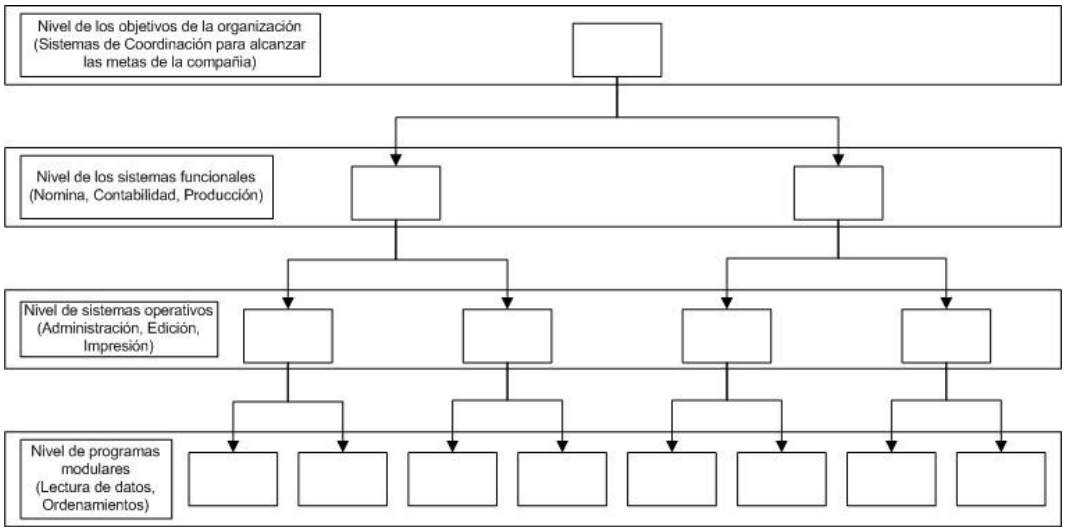


Figura 3- 11 El enfoque descendente.

Evitar el caos originado al tratar de diseñar el sistema "en un solo paso", es una de las ventajas que tiene el diseño descendente, otra es la posibilidad de contar con grupos trabajando por separado pero simultáneamente en subsistemas independientes, pero necesarios, una más es prevenir que se pierdan los objetivos centrales del sistema por adentrarse en los detalles.

Los inconvenientes pueden ser que el sistema se divida en subsistemas incorrectos, otro que una vez que se realizan las divisiones en subsistemas, sus interfaces pueden descuidarse o simplemente ignorarse.

Desarrollo modular.

Significa descomponer la programación en fracciones lógicas y manejables. De manera que cada módulo debe tener la funcionalidad de solamente satisfacer una sola función.

El desarrollo modular tiene tres ventajas principales, las cuales son:

1. *Facilidad de escritura.* Los módulos son más fáciles de escribir y revisar, ya que están virtualmente autocontenidos.
2. *Mantenimiento.* El mantenimiento a los módulos es más fácil, ya que las modificaciones se realizan a unos cuantos módulos y no al programa completo.
3. *Fácil entendimiento.* La problemática de los módulos es más fácil de entender.

De igual manera, existen cuatro lineamientos para la programación modular. Estos son:

1. Mantener cada módulo de un tamaño manejable (de manera ideal incluyendo sólo una función).
2. Prestar atención particular a las interfaces criticas (esto es, a los datos y a las variables de control que pasan entre módulos).
3. Minimizar el número de módulos que el usuario necesite modificar cuando haga cambios.
4. Mantener las relaciones jerárquicas establecidas en las etapas de descenso.

Los módulos de un sistema pueden ir agrupados de acuerdo a las siguientes categorías, las cuales determinarán su grupo:

- **Funcional.** Ocurre cuando todas las actividades de un módulo tienen el mismo propósito único o función.
- **Datos.** Es cuando en el módulo todos los elementos se refieren a los mismos datos o archivos.
- **Lógica.** Es cuando en el módulo, todos los pasos se llevan a cabo juntos o se manejan las mismas funciones.
- **Sin relación estrecha.** Es el tipo de agrupamiento menos deseable y consiste en pasos que no desarrollan ninguna función por completo o que lógicamente no van juntos.

De las cuatro agrupaciones posibles, la “Funcional” es la más óptima y la de “Sin relación estrecha” es la menos deseable.

Diagrama estructural.

Un sistema estructurado es aquel que se desarrolla en forma descendente y modular. Los módulos mismos son relativamente simples, lo que significa que tienen un efecto mínimo sobre otros módulos dentro del sistema. Las conexiones entre los módulos se limitan y la interacción de los datos es la mínima.

El instrumento recomendado para el diseño de un sistema modular descendente se denomina diagrama estructural. Un diagrama estructural simplemente es un diagrama que consta de rectángulos, los cuales representan a los módulos y se conectan por medio de flechas. Las flechas de conexión se dibujan con una dirección de arriba hacia abajo.

A los lados de las flechas de conexión se dibujan dos flechas más pequeñas. Tales flechas con círculos vacíos se denominan “parejas de datos”, mientras que las flechas con los círculos llenos se denominan “indicadores de control”. Estas flechas indican que algo pasa, ya sea hacia abajo del módulo superior o de regreso del módulo inferior.

Otro símbolo que se utiliza es un bucle o lazo (loop), este símbolo indica que los procedimientos encontrados en los módulos se repiten hasta el final.

Un símbolo más es un pequeño diamante. El diamante se coloca en la base de uno de los rectángulos y significa que se ejecutará solo uno de los módulos debajo del diamante.

La Figura 3-12 muestra los elementos de un diagrama estructural.

Figura 3- 12 Elementos de un diagrama estructural.

El diagrama estructural, por si solo, no puede mostrar el orden en que se deben ejecutar los módulos, ni tampoco puede mostrar bastante detalle.

Herramientas de documentación.

Actualmente no existe una sola técnica sencilla y estandarizada para la documentación y el diseño. Cada técnica tiene sus propias ventajas y desventajas. La Figura 3-13 muestra una grafica con varias técnicas, las cuales están ubicadas en base a dos atributos: primero, qué tan estructurada es, y segundo, qué tan visual es.

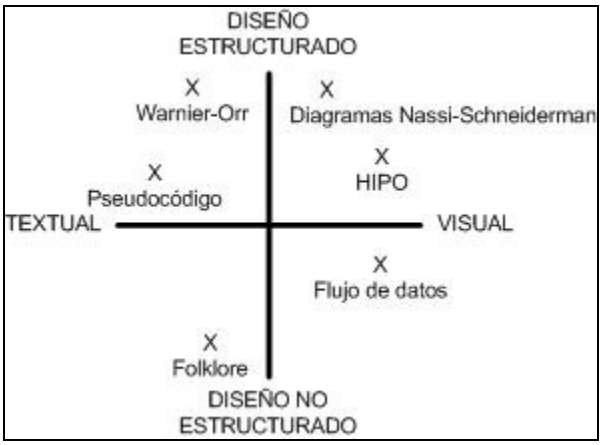


Figura 3- 13 Relación de técnicas de diseño y documentación.

Diagramas HIPO.

HIPO es una abreviatura de Hierarchy Input/Process/Output (Jerarquía Entrada/Proceso/Salida); Este método fue desarrollado por IBM.

Es una técnica jerárquica porque el sistema completo de programación se conforma de pequeños subsistemas. Soporta un enfoque de diseño descendente y también reduce la complejidad, ya que cada uno de los subcomponentes puede consultarse de manera separada.

Nos recuerda las tres partes principales de cualquier sistema: entrada, proceso y salida. Una vez que el diagrama jerárquico se completa, se elaboran otros diagramas HIPO en páginas divididas verticalmente entre las secciones, donde la sección de la izquierda corresponde a la entrada, la sección del centro al proceso, y la sección de la derecha a la salida.

Existen tres tipos principales de diagramas en los sistemas HIPO:

1. *Tabla Visual de Contenido*. También conocida como VTOC (por sus siglas en ingles, Visual Table of Contents) es el diagrama de jerarquías. Proporciona al lector un mapa que le permite localizar un módulo existente dentro del sistema principal. El diagrama parece ser similar a un diagrama de la estructura de una organización (organigrama), con la diferencia de que abajo del diagrama hay un espacio con una descripción más detallada de los cuadros.

2. *Diagrama general IPO*. Permite una visión global de la entrada, el proceso y la salida, y en consecuencia se refiere como diagrama panorámico. Se listan todas las entradas, los procesos y las salidas en las tres secciones de papel sin dibujar símbolos especializados.
3. *Diagramas detallados IPO*. Los diagramas generales se descomponen en cada uno de los módulos autocontenidos en él, los cuales nos dan como resultado los diagramas detallados. Aquí se lista la entrada, el proceso y la salida del módulo que se esta detallando, y se recomienda utilizar símbolos especializados para la elementos de la entrada y la salida.

La Figura 3-14 muestra el formato de una Tabla Visual de Contenido.

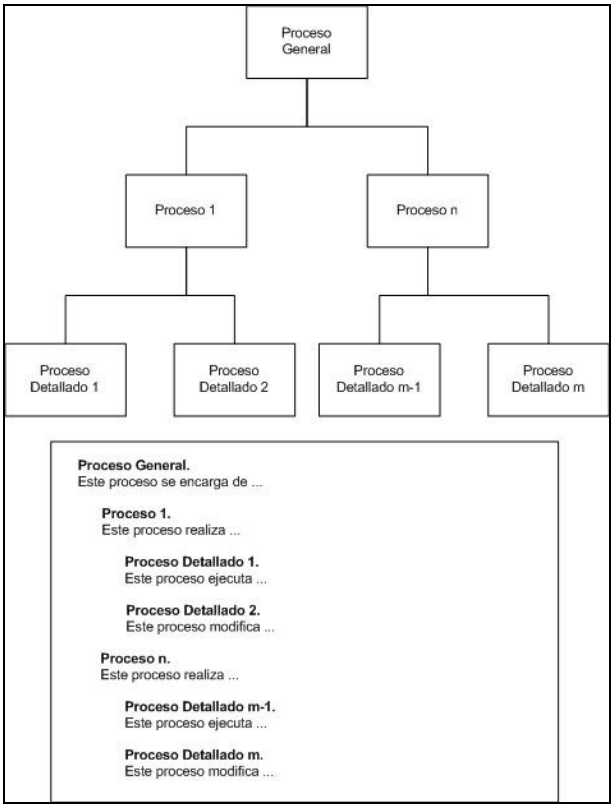


Figura 3- 14 Formato de una VTOC.

La Figura 3-15 muestra el formato de un diagrama general IPO.

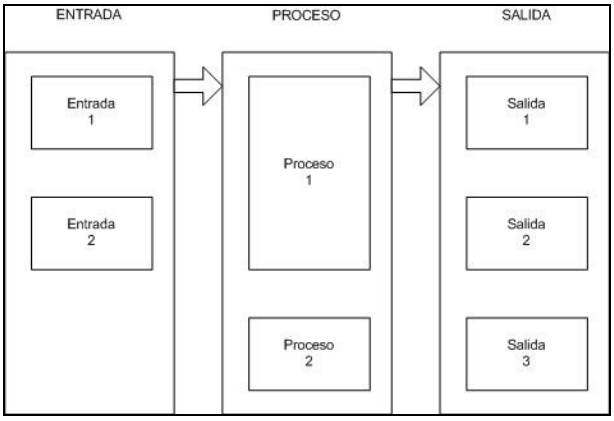


Figura 3- 15 Formato de un diagrama general IPO.

La Figura 3-16 muestra el formato de un diagrama detallado IPO.

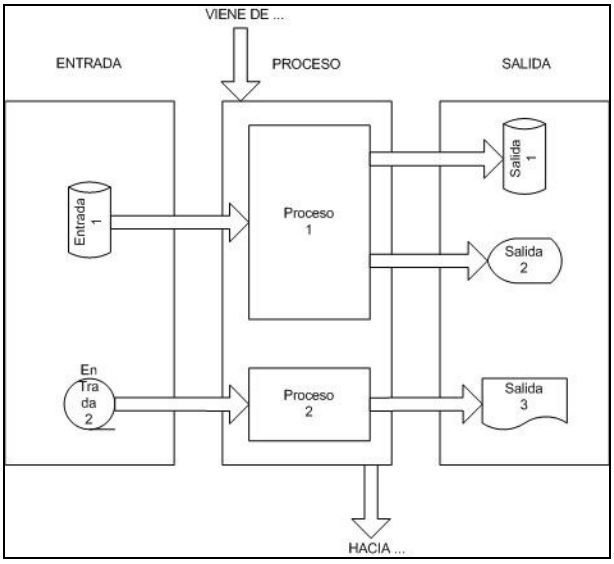


Figura 3- 16 Formato de un diagrama detallado IPO.

El HIPO es una técnica estructurada altamente visual para el diseño y la documentación, cuando se han familiarizado con los símbolos que usa.

Cuando no se ha logrado la familiarización con los símbolos, el HIPO se vuelve un instrumento demasiado especializado; también, se requiere de una cantidad de espacio gráfico considerado.

Diagramas de flujo.

Otro enfoque más estructurado para el diseño es el diagrama de flujo ordinario. Existen numerosas desventajas en su uso: no se elaboran con base en los principios de la programación estructurada, de tal forma que ilustran el flujo del programa, pero no su estructura; se requiere gran cantidad de espacio; es necesario revisar varias páginas para asimilar el contenido del programa. Cuenta con demasiadas ramificaciones, cada una de ellas proviene de cada decisión del diagrama de flujo.

La mejor razón para utilizar un diagrama de flujo es que han sido utilizados históricamente y con el tiempo llegan a entenderlo mejor que cualquier otra técnica más reciente.

La Figura 3-17 muestra los elementos de un diagrama de flujo.

Figura 3- 17 Elementos de un diagrama de flujo.

Diagramas Nassi-Schneiderman.

Un enfoque más estructurado es el diagrama Nassi-Schneiderman (N-S). La principal ventaja de un diagrama N-S es que adopta la filosofía de la programación estructurada; otra, es que utiliza un número limitado de símbolos, de tal forma que el diagrama ocupa menos espacio y puede leerse con cierta facilidad.

Existen tres elementos básicos utilizados en el desarrollo de un diagrama Nassi-Schneiderman, los cuales son:

1. *Proceso*. Es un cuadro que sirve para representar cualquier proceso o paso sencillo en un programa. Este símbolo representa la inicialización de valores, las actividades de entrada y salida y las llamadas para ejecutar otros procedimientos.
2. *Decisión*. Se representa por una columna dividida por un triangulo incorporado. Este símbolo representa las condiciones alternas que pueden ocurrir, es el equivalente de la estructura IF-THEN-ELSE.
3. *Iteración o Repetición*. Es un cuadro dentro de otro cuadro. El cuadro dentro del cuadro aparece con sangría en el diagrama global. Representa la anidación y repetición de las estructuras WHILE, UNTIL, y FOR.

La Figura 3-18 muestra los tres elementos de un diagrama Nassi-Schneiderman.

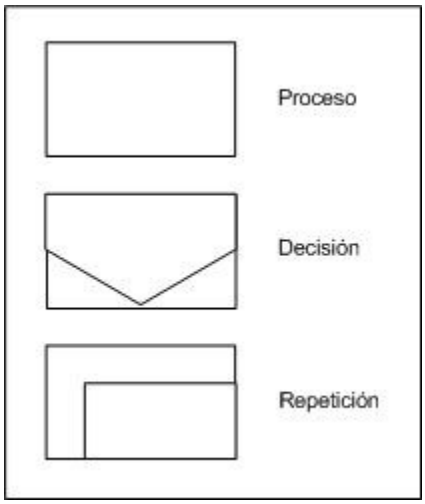


Figura 3- 18 Elementos de un diagrama Nassi-Schneiderman.

En la programación estructurada se utiliza un enfoque descendente. El analista comenzará dibujando primero las principales subrutinas y luego hará una sangría para completar más adelante las subrutinas internas.

Los diagramas deben estar completos y ser muy claros, con el fin de que se entiendan, esto es una desventaja. Si de manera regular se hacen cambios, los diagramas no serán apropiados, ya que deberán dibujarse nuevamente; y en consecuencia, su modificación no es sencilla.

Los beneficios son múltiples. Proporcionan un instrumento de ayuda para el diseño de programas y de su proceso de desarrollo, pues son compatibles con la programación estructurada. También, los diagramas son fáciles de leer porque no requieren del conocimiento de símbolos complejos.

Diagramas Warnier/Orr.

Esta técnica ayuda al diseño de las estructuras de un programa identificando la salida y los resultados del procesamiento y después trabaja retrocediendo para determinar los pasos y combinaciones de entrada que se necesiten para producirlos.

Los únicos símbolos que utiliza son las llaves, que se utilizan para representar conjuntos y subconjuntos, y las variables tales como M y N que representan el número de casos en una iteración. Cuando existe una condición, se cumpla o no, se utiliza la notación (0, 1) y un símbolo + significa que la lista de elementos que lo contenga son las posibles alternativas. Se utiliza PERFORM para dirigirse a otra parte del programa.

La Figura 3-19 muestra los elementos de un diagramaWarnier/Orr.

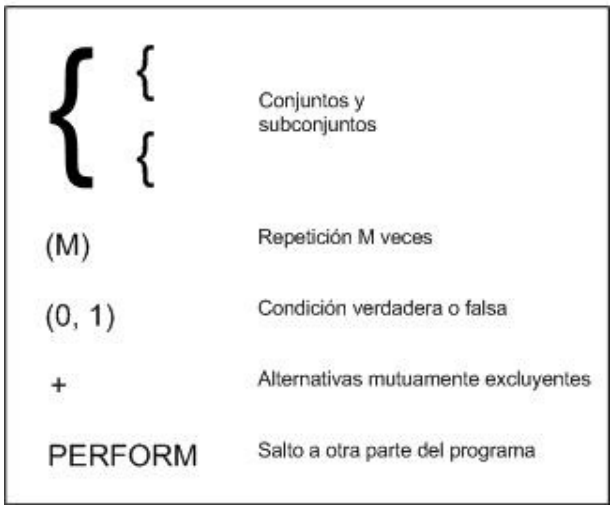


Figura 3- 19 Elementos de un diagrama Warnier/Orr.

El diagrama Warnier/Orr tiene la ventaja de mostrar agrupamientos de procesos y los datos que deben pasar de un nivel a otro; asimismo, la secuencia de trabajar retrocediendo asegura que el sistema estará orientado hacia los resultados. Otra ventaja es que los diagramas no necesitan estar terminados para empezar a utilizarse. Una más es que son fáciles de modificarse.

Pseudo código.

El pseudo código es similar al Lenguaje Estructurado, en el sentido de que no es un tipo particular de código de programación, pero puede utilizarse como paso intermedio en el desarrollo del código de programación.

El uso del pseudo código es común en la industria, pero al carecer de una estandarización, le confiere en general, poca aceptación. Ya que es tan similar a un código de programación, llega a ser común que lo utilicen los programadores.

Método Folklore.

Es una técnica de documentación de sistemas, que ha sido creada para completar algunas otras técnicas. La técnica del Folklore recompila la información que comparten los usuarios, pero que rara vez queda plasmada por escrito.

El objetivo es recopilar aquella información que se encuentre en cualquiera de las siguientes cuatro categorías:

1. *Costumbres y hábitos*. Se refiere a lo que los usuarios hacen para lograr que los programas funcionen sin problemas.
2. *Cuentos*. Los cuentos son historias que los usuarios platican con referencia a su manera de ver el sistema.
3. *Expresiones*. Son breves planteamientos que representan generalizaciones o recomendaciones.
4. *Manifestaciones artísticas*. Son los diagramas de flujo, las figuras y las tablas que desarrollan los usuarios.

El enfoque del Folklore funciona porque puede auxiliar a la falta de conocimiento creada cuando un autor se retira.

El riesgo de confiar en la técnica es que la información recopilada por los usuarios puede no ser objetiva, parcialmente correcta o peor aún, incorrecta.

Elección de una técnica de documentación.

Para elegir la técnica de documentación que se va ha adoptar para el diseño de un sistema, se recomienda considerar el siguiente conjunto de lineamientos que auxilian en la elección de una técnica adecuada:

- **Compatible.** Sea compatible con la documentación existente.
- **Comprensible.** Sea comprensible dentro de la organización.
- **Reutilizable.** Le permita regresar a trabajar en el sistema, una vez que se haya retirado por un buen intervalo de tiempo.
- **Adecuada.** Sea adecuada para el trabajo del sistema en el cual trabaja.
- **Estructurada.** Le permita un enfoque de diseño estructurado, si se considera que esto sea más relevante que los otros factores.
- **Modificable.** Permita una fácil modificación.

Implantación del desarrollo en la solución del problema.

La forma en que se diseñó la Aplicación es descendente, y se utilizó un desarrollo modular. Esta decisión se tomó debido a las ventajas que ofrecen ambas formas para realizarse, y principalmente a que el Sistema para el cual se desarrolló la Aplicación fue diseñado bajo estos conceptos.

Las Figuras 3-20, 3-21 y 3-22 muestran el diagrama estructural de la Aplicación.

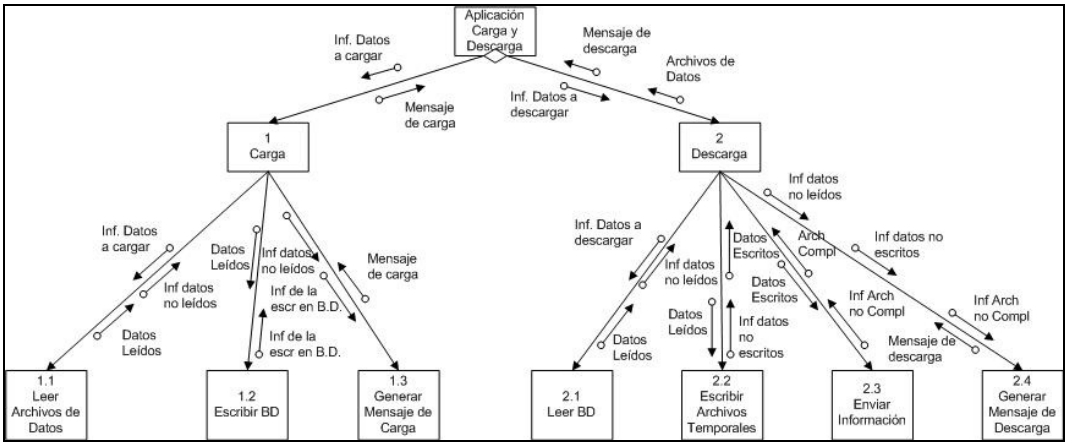


Figura 3- 20 Diagrama estructural de la Aplicación.

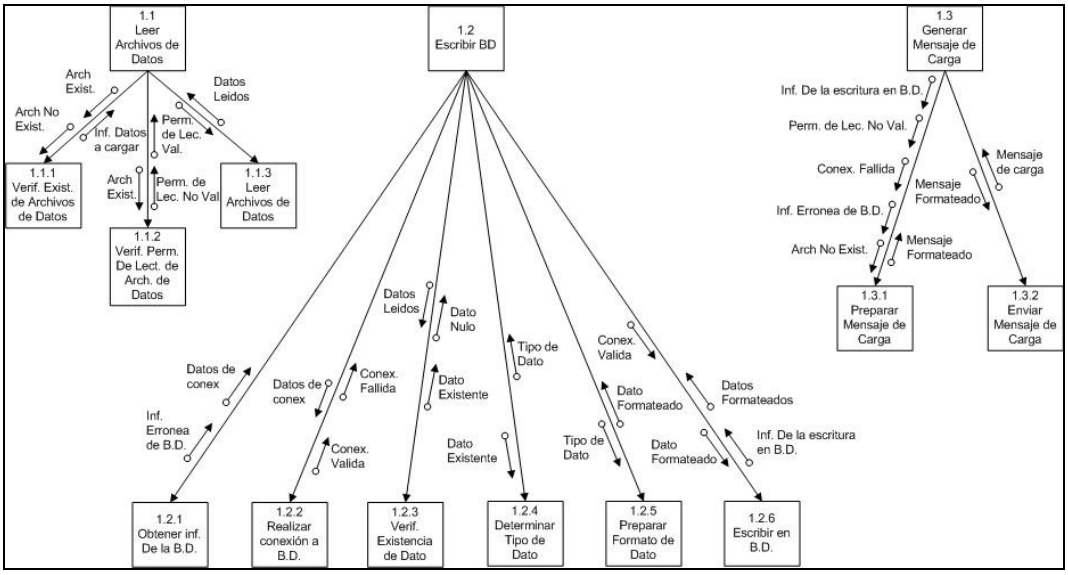


Figura 3- 21 Diagrama estructural de la Aplicación (Cont).

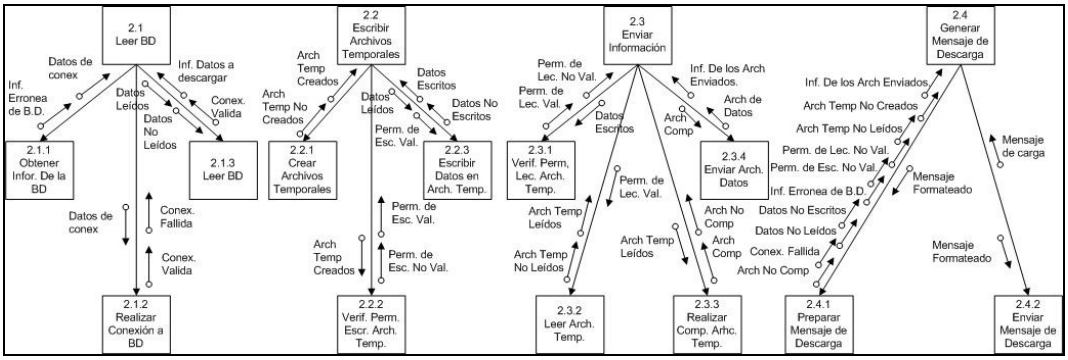


Figura 3- 22 Diagrama estructural de la Aplicación (Cont).

Los métodos de documentación del diseño que se utilizaron son el diagrama de flujo, y el pseudo código. Estos métodos se emplearon debido a que fue el predeterminado que se utilizó para documentar el diseño del Sistema con el que interactúa la Aplicación.

Después de realizar el diseño de la Aplicación se obtuvo el diagrama de flujo referente a ella, el cual se muestra en las Figuras 3-23, 3-24 y 3-25.

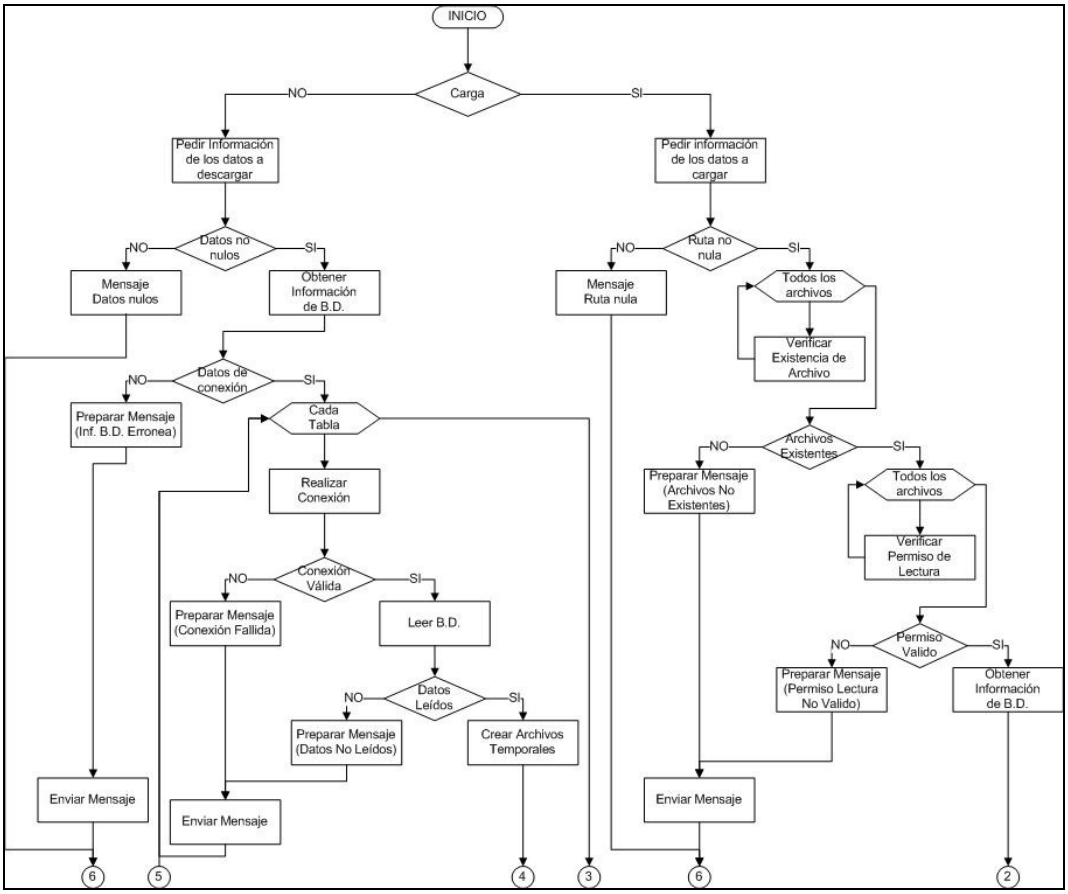


Figura 3- 23 Diagrama de flujo de la Aplicación.

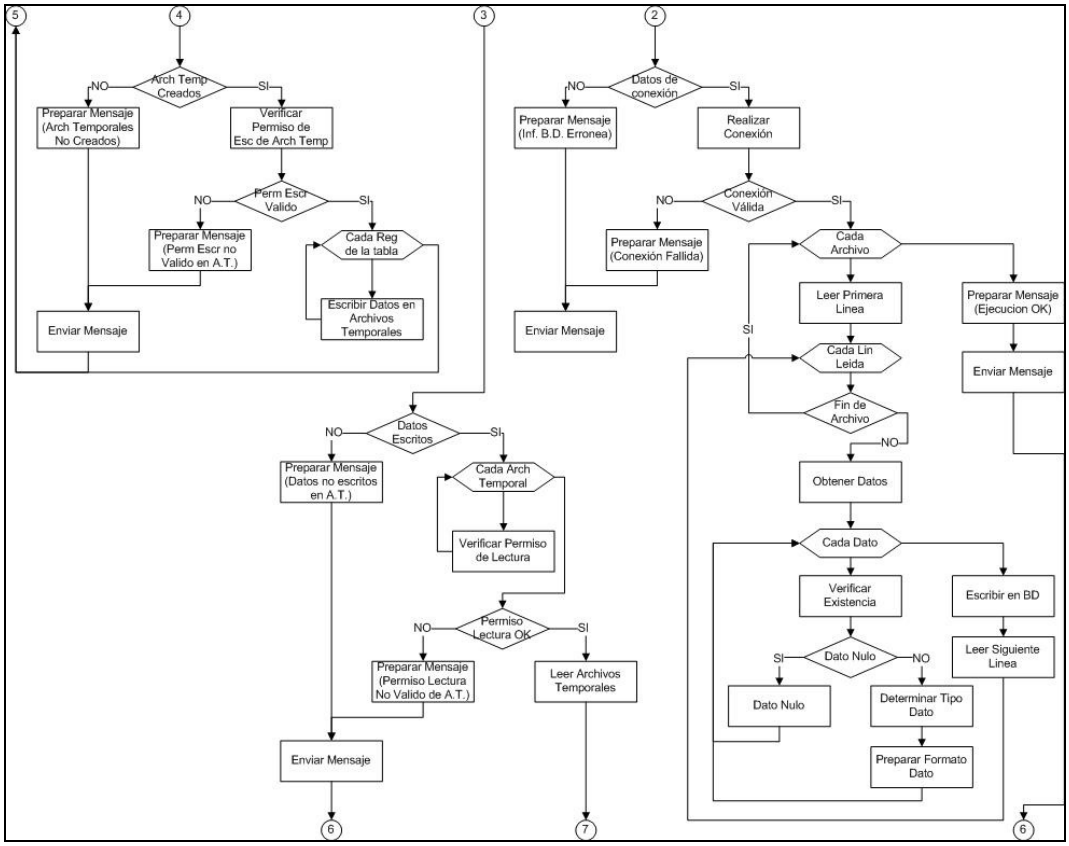


Figura 3- 24 Diagrama de flujo de la Aplicación (cont).

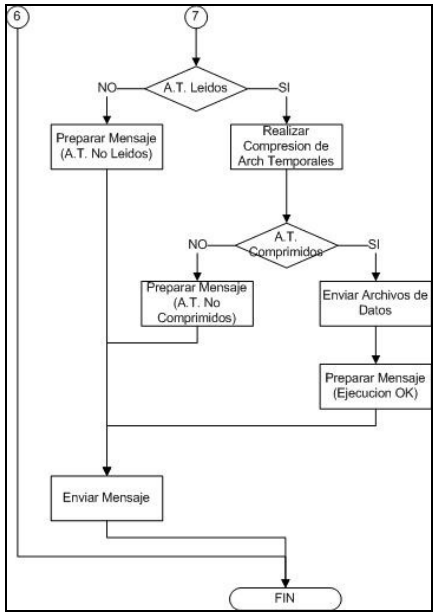


Figura 3- 25 Diagrama de flujo de la Aplicación (cont).

El pseudo código de la Aplicación se muestra a continuación:

```
if (carga)
{
    Pedir Información de los datos a cargar
    if (ruta no nula)
    {
        for (todos los archivos requeridos)
        {
            Verificar existencia de Archivo de Datos
        }
        if (Archivos Existentes)
        {
            for (todos los archivos)
            {
                Verificar permiso de lectura de Archivos de Datos
            }
            if (Permiso de Lectura Válido)
            {
                Obtener información de la BD
                if (Datos de conexión)
                {
                    Realizar conexión a BD
                    if (Conexión válida)
                    {
                        for (cada uno de los archivos)
                        {
                            Leer primera línea
                            for (cada línea leída)
                            {
                                if (línea leída es fin de archivo)
                                {
                                    Continuar con siguiente archivo
                                }
                                Obtener datos de línea leída
                                for (cada dato)
                                {
                                    Verificar Existencia de Dato
                                    if (Dato Existente)
                                    {
                                        Determinar Tipo de Dato
                                        Preparar Formato de Dato
                                    }
                                    else
                                    {
                                        Colocar Dato Nulo
                                    }
                                }
                                Escribir en BD
                                Leer siguiente línea de archivo
                            }
                        }
                        Preparar Mensaje de Carga: “Ejecución correcta, carga finalizada”
                        Enviar Mensaje de Carga
                    }
                    else
```

```
                    {
                        Preparar Mensaje de Carga: “Conexión fallida”
                        Enviar Mensaje de Carga
                    }
                }
                else
                {
                    Preparar Mensaje de Carga: “Información errónea de la base de datos”
                    Enviar Mensaje de Carga
                }
            }
            else
            {
                Preparar Mensaje de Carga: “Permiso de lectura no válido”
                Enviar Mensaje de Carga
            }
        }
        else
        {
            Preparar Mensaje de Carga: “Archivos no existentes”
            Enviar Mensaje de Carga
        }
    }
    else
    {
        Mostrar mensaje de ruta nula
    }
}
else
{
    Pedir Información de los datos a descargar
    if (datos a descargar no nulos)
    {
        Obtener Información de la BD
        if (Datos de conexión)
        {
            for (cada tabla de la BD)
            {
                Realizar conexión a BD
                if (Conexión válida)
                {
                    Leer la BD
                    if (Datos Leídos)
                    {
                        Crear Archivos Temporales
                        if (Archivos Temporales creados)
                        {
                            Verificar Permiso de Escritura de Archivos Temporales
                            if (Permiso de Escritura válido)
                            {
                                for (cada registro de la tabla leída)
                                {
                                    Escribir Datos en Archivos Temporales
                                }
                            }
                            else
```

```
                    {
                        Preparar Mensaje de Descarga: “Permiso de Escritura no válido”
                        Enviar Mensaje de Descarga
                    }
                }
                else
                {
                    Preparar Mensaje de Descarga: “Archivos Temporales no creados”
                    Enviar Mensaje de Descarga
                }
            }
            else
            {
                Preparar Mensaje de Descarga: “Datos no Leídos”
                Enviar Mensaje de Descarga
            }
        }
        else
        {
            Preparar Mensaje de Descarga: “Conexión fallida”
            Enviar Mensaje de Descarga
        }
    }
    if (Datos Escritos)
    {
        for (cada Archivo Temporal)
        {
            Verificar Permiso de Lectura de Archivos Temporales
        }
        if (Permiso de Lectura de Archivos Temporales)
        {
            Leer Archivos Temporales
            if (Archivos Temporales Leídos)
            {
                Realizar Compresión de Archivos Temporales
                if (Archivos Comprimidos)
                {
                    Enviar Archivos de Datos
                    Preparar Mensaje de Descarga: “Ejecución correcta, descarga finalizada”
                    Enviar Mensaje de Descarga
                }
                else
                {
                    Preparar Mensaje de Descarga: “Archivos no Comprimidos”
                    Enviar Mensaje de Descarga
                }
            }
            else
            {
                Preparar Mensaje de Descarga: “Archivos Temporales no Leídos”
                Enviar Mensaje de Descarga
            }
        }
        else
        {
            Preparar Mensaje de Descarga: “Permiso de Lectura no válido Archivos Temporales”
```

```
                    Enviar Mensaje de Descarga
                }
            }
            else
            {
                Preparar Mensaje de Descarga: "Datos no escritos en Archivos Temporales"
                Enviar Mensaje de Descarga
            }
        }
        else
        {
            Preparar Mensaje de Descarga: "Información errónea de la base de datos"
            Enviar Mensaje de Descarga
        }
    }
    else
    {
        Mostrar mensaje de datos a descargar nulos
    }
}
```

Pruebas.

Los analistas utilizan cuatro niveles para asegurar la calidad de un sistema: prueba, verificación, validación, y certificación. La prueba es el proceso de ejecutar un programa con la intención específica de encontrar errores, por lo que, una prueba exitosa es aquella en la que se encuentra un error. La verificación se utiliza con objeto de encontrar errores al ejecutar el programa en un ambiente simulado. La validación se refiere al proceso de utilizar el software en un ambiente real para encontrar errores. La certificación del software es la confirmación de que el programa es correcto. La etapa de prueba de programa se lleva a cabo para probar que no existen errores en un programa.

En general, la parte correspondiente a pruebas se refiere a que debe haber una evaluación total de todos los elementos del sistema. La evaluación se debe llevar a todo lo largo del desarrollo del sistema (no solo al final); cumple con el propósito de identificar aquellos problemas desconocidos. Aunque la evaluación es tediosa, conforma una serie esencial de pasos que ayudan a garantizar la calidad del sistema eventual. Es menos grave evaluar de antemano, que tener un sistema pobremente evaluado y que falle una vez instalado.

Existen varias etapas en lo que corresponde a las pruebas, dichas etapas son: Prueba del programa con datos de prueba, Prueba del enlace con datos de prueba, Prueba del sistema completo con datos de prueba, y Prueba del sistema completo con datos reales.

Así como existen etapas de prueba, también existen dos categorías generales para probar un software, la primera se llama Prueba de código, y la segunda Prueba de especificación. El uso de cada categoría va de acuerdo a la etapa de prueba que se esta ejecutando.

Un Caso de prueba es un conjunto de datos que el sistema procesará como una entrada normal; sin embargo, los datos son creados con la intención expresa de determinar si el sistema los procesará correctamente.

Prueba del programa con datos de prueba.

Los programadores deben revisar primero sus programas para verificar la manera en la que trabajará el sistema, con una evaluación de escritorio para verificar si las rutinas trabajan como se planteó.

Luego deben desarrollar datos de prueba, tanto válidos como inválidos, para ver si las rutinas básicas trabajan y también para generar errores. Los datos de prueba deben examinar los valores mínimos y máximos posibles, así como todas las variaciones que sean posibles en formato y en los códigos. La salida de datos de prueba del archivo debe verificarse con cuidado.

Prueba de enlace con datos de prueba.

Las evaluaciones de enlace verifican que los programas sean interdependientes y funciones integralmente tal y como fue planeado.

Para la prueba se utiliza una pequeña cantidad de datos de prueba. El examen de todas las combinaciones puede implicar varios pasos a través del sistema. Primero se procesan datos de prueba típicos para transacciones normales. Luego se agregarán variaciones, las cuales incluirán datos inválidos para asegurar que el sistema puede detectar los errores de manera adecuada.

Prueba del sistema completo con datos de prueba.

Debe examinarse el sistema como entidad completa. En esta etapa, los operadores y los usuarios finales se involucran activamente en tal operación.

Existe una serie de factores a considerar cuando se evalúan los sistemas con datos de prueba:

- **Documentación.** Verificar si los operadores cuentan con una documentación adecuada en los manuales de procedimientos para asegurar una operación correcta.
- **Manuales.** Verificar si los manuales de procedimientos son suficientemente claros como para comunicar la manera de preparar los datos de entrada.
- **Carga.** Asegurarse si el flujo de carga que debe tolerar el nuevo sistema o la modificación “fluye correctamente”.
- **Salida.** Determinar si la salida es correcta; esto es, que todos los usuarios comprendan en forma general como llegará a ser la salida.

Prueba del sistema completo con datos de reales.

Cuando la prueba de los sistemas con datos de prueba llega a ser satisfactoria, es una buena idea tratar que el sistema interactúe con lo que se llama “datos reales”, los cuales son datos que han sido procesados con éxito por el sistema existente. Esto permite una comparación precisa con lo que es una salida procesada correctamente, y una buena idea de cómo deben manejarse los datos reales.

Es un periodo importante para evaluar la manera en que los usuarios finales y operadores interactúan con el sistema. Los elementos a observar son: la facilidad de aprendizaje del sistema; el ajuste factores ergonómicos; las reacciones de los usuarios a la retroalimentación del sistema, incluyendo lo que ocurra cuando se presente en pantalla un mensaje de error. También se debe escuchar lo que los usuarios dicen acerca del sistema sobre cualquier problema real que necesita ser atendido antes de que el sistema entre en operación.

Pruebas especiales de sistemas.

Existen otras pruebas que están en una categoría especial, dado que no se enfocan a la forma normal de trabajo de un sistema. Seis pruebas son esenciales:

1. *Prueba de carga máxima.* En esta prueba se determinará si el sistema manejará el volumen de actividades que ocurren cuando éste se encuentra en el punto máximo de demandas de procesamiento.
2. *Prueba de almacenamiento.* Aquí debe determinarse la capacidad del sistema para almacenar datos de transacciones en un disco duro u otros archivos.
3. *Prueba de desempeño en tiempo.* Debe determinarse el tiempo que utilizó el sistema para el procesamiento de datos de transacciones.
4. *Prueba de recuperación.* Esta prueba debe determinar la posibilidad del usuario para recuperar datos o para iniciar el sistema nuevamente después de un error.
5. *Prueba de procedimiento.* Aquí debe determinarse la claridad de la documentación en operación y uso del sistema logrando que los usuarios hagan exactamente lo que los manuales indican.
6. *Prueba de factores humanos.* Debe determinarse la manera en que los usuarios utilizarán el sistema cuando procesen datos o preparen informes.

La aplicación de estas pruebas especiales queda a consideración de los desarrolladores; sin embargo, se recomienda que se apliquen al sistema siempre que se pueda, en la etapa de desarrollo.

Realización de las pruebas en la aplicación desarrollada.

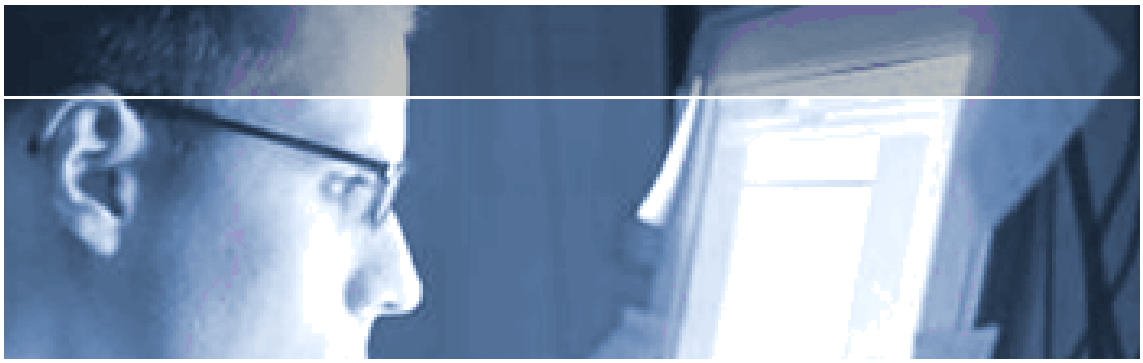
Una vez que el desarrollo de la Aplicación se concluyó, se pudo decir que en ese momento la Aplicación ya había pasado por una serie de pruebas como son: *prueba del programa con datos de prueba*, ya que como se indicó, esta prueba se llevó acabo mientras se estuvo desarrollando el programa; otra de las pruebas que fue cubierta es la *prueba de enlace con datos de prueba*, ya que también se realizó durante la programación cuando se fue verificando el correcto funcionamiento de la Aplicación, y también la correcta integración con el resto del sistema. La *prueba del sistema completo con datos de prueba* se realizó durante el periodo de "Aseguramiento de Calidad" del sistema con el que interactúa la Aplicación. La *prueba de sistema con datos reales*, en este caso, se realizó hasta que el sistema fue liberado, y la Aplicación fue utilizada, ya que no existían antecedentes para poder realizar una comparación previa.

A lo largo del proceso de pruebas se desarrollo una *tabla de pruebas*, para llevar un control de las pruebas realizadas, y de los resultados que se debían obtener. Esta tabla contiene la entrada que se alimentó a la Aplicación y el resultado que ésta entregó. La Tabla 3-2 muestra la tabla de pruebas utilizada.

Entrada	Salida
Ruta vacía de archivos a cargar.	Mensaje indicando que la ruta no puede ser vacía.
Ruta errónea de archivos a cargar.	Mensaje indicando que la ruta no existe.
Archivos incompletos en la carga.	Mensaje indicando que faltan archivos requeridos.
Selección de datos a descargar no valida.	Mensaje indicando que se debe seleccionar una opción valida.

Tabla 3- 2 Tabla de entradas y salidas usada en las prueba

Capítulo 4: Implantación y Solución.



Introducción.

En este capitulo se hablara de los temas Centros de Información, Adiestramiento de los Usuarios, Conversión del Sistema, Evaluación del Sistema y de la Implantación de la Aplicación Desarrollada.

En el primer tema Centros de Información se mencionan los objetivos del centro de información y la preparación del sitio.

En el segundo tema se habla de ¿a quién adiestrar? (operadores y usuarios), del personal que adiestra a los usuarios y de los lineamientos para el adiestramiento (objetivos, métodos, instalaciones y materiales).

El tercer tema menciona los métodos de conversión (reemplazo total, conversión en paralelo, conversión gradual, prototipo modular y conversión distribuida) y de las ventajas y desventajas de cada uno, el plan de conversión, la seguridad (lógica, física y conductual) y de otras consideraciones.

El cuarto tema habla sobre el enfoque de utilerías (de posesión, de forma, de lugar, de tiempo, de actualización y de objetivo) y de la evaluación del sistema.

El quinto tema habla de la implantación de la solución desarrollada para resolver el problema y de la evaluación que se le hizo a esta.

Implantación y Solución.

"Se denomina implantación o implementación al proceso que asegura la operatividad del sistema de información y que permite al usuario obtener beneficios por su operación."[17]

El analista dispone de diversos enfoques para la implantación. Estos enfoques incluyen la asignación de mayor capacidad en el uso del equipo de cómputo a los usuarios, vía un centro de información, el adiestramiento de los usuarios, la conversión de sistemas antiguos y la evaluación de los nuevos.

Centro de información.

Como manera de facilitarles a los usuarios el cumplimiento de sus necesidades de información a corto plazo, mientras recibe el soporte del departamento de sistemas, se propone la creación e implantación de un centro de información.

Objetivos del centro de información.

Un centro de información tiene como objetivo primario apoyar a los usuarios internos de la organización en el acceso de los datos, de tal forma que puedan formular, analizar y resolver sus propios problemas o preguntas de trabajo mediante el uso de las computadoras.

Preparación del sitio.

Los analistas con frecuencia trabajan con los vendedores del equipo para especificar las características del lugar de operación. El cliente o los ingenieros de sistemas presentarán una lista de especificaciones para disponer el cableado del sistema eléctrico, las tomas de corriente eléctrica, las necesidades del sistema de aire acondicionado, los controles de humedad, los requisitos de espacio, y cualquier otra característica que se requiera para preparar el lugar.

En caso de que una microcomputadora constituya el sistema, o sea suficiente para el funcionamiento del sistema que se ha diseñado o adquirido, se necesita poca preparación del sitio de trabajo.

Adiestramiento de los usuarios.

Los sistemas bien diseñados y técnicamente elegantes pueden tener éxito o fallar debido a la forma en se operan y utilizan. Las personas que trabajarán con el sistema o que se verán afectadas por éste deben conocer con detalle las funciones que desempeñarán, cómo utilizarán el sistema y lo que este hará o no. Tanto los operadores como los usuarios necesitan capacitación.

[17] KENDALL Kenneth E, KENDALL Julie E. "Análisis y Diseño de Sistemas"

Se denomina adiestramiento o capacitación al proceso educativo que involucra a los analistas de sistemas con los usuarios.

Las estrategias de adiestramiento determinan a quiénes se adiestrarán y quién los adiestrará.

¿A quién adiestrar?

Todos aquellos que tengan un uso primario o secundario del sistema deben ser adiestrados. Esto incluye a cualquier persona, desde el capturista (usuario primario) hasta quienes utilizarán la salida para la toma de decisiones (usuario secundario), aun sin que llegue a ver una terminal u otro equipo.

Adiestramiento de operadores de sistemas.

Muchos sistemas dependen del personal del centro de cómputo, quienes tienen la responsabilidad de mantener el equipo en buenas condiciones, así como de proporcionar el servicio de apoyo necesario. Su capacitación debe garantizar que están en condición para manejar todas las operaciones posibles, tanto las de rutina como las extraordinarias.

La capacitación incluye también procedimientos de operación, es decir, la manera de trabajar la secuencia de actividades necesarias para emplear u operar un nuevo sistema.

Adiestramiento de usuarios.

La mayor parte de la capacitación de los usuarios radica en la operación del sistema mismo. Las actividades del manejo de datos que reciben la máxima atención dentro de la capacitación de los usuarios son la entrada de datos, la edición de datos, la formulación de consultas, y el borrado de registros de datos.

La capacitación de usuarios puede incluir el empleo del equipo, por ejemplo, en caso de que se emplee una microcomputadora y el individuo sea simultáneamente operador y usuario. En este caso el usuario debe recibir instrucciones primero respecto a la operación del equipo.

Existen dos aspectos para la capacitación del usuario:

1. *Familiaridad.* La familiaridad con el sistema de procesamiento; es decir, con el equipo usado para los datos de entrada o su procesamiento.
2. *Empleo.* La capacitación en el empleo de la Aplicación; es decir, el uso del software que acepta los datos, los procesa y produce resultados.

Personal que adiestra a los usuarios.

Para un proyecto pueden utilizarse diferentes tipos de instructores, dependiendo de cuantos usuarios serán adiestrados y quiénes. Las posibles fuentes de adiestramiento incluyen:

- **Vendedores.** Los vendedores pueden capacitar a usuarios secundarios.
- **Instructores externos contratados.** Los instructores externos pueden capacitar a usuarios secundarios.
- **Analistas de sistemas.** Los analistas de sistemas pueden capacitar tanto a usuarios secundarios como a usuarios primarios.
- **Instructores internos.** Los instructores internos solamente puede capacitar a usuarios primarios.
- **Otros usuarios del sistema.** Cuando el instructor es un usuario del sistema, solamente puede capacitar a los usuarios primarios.

Lineamientos para el adiestramiento.

El analista debe apegarse a cuatro lineamientos principales para implantar el adiestramiento, dichos lineamientos son:

Objetivos del adiestramiento.

Los objetivos del adiestramiento deben plantearse en forma clara para cada uno de los grupos que lo recibirán. Los objetivos en gran medida son dictados por quien se adiestra.

Métodos de adiestramiento.

Cada usuario y cada operador necesitarán un adiestramiento ligeramente diferente, sus tareas determinan lo que ellos necesitarán saber.

Ciertos usuarios aprenden mejor al observar, mientras que otros al escuchar y otros al ejecutar. Ya que generalmente no es posible preparar el adiestramiento para un determinado individuo, con frecuencia una combinación de los tres métodos es la mejor manera de proceder.

Instalaciones para el adiestramiento.

El adiestramiento se lleva a cabo en localidades diferentes (instalaciones del proveedor, instalaciones de la compañía donde labora el empleado, o lugares contratados), algunas de ellas resultan más adecuadas que otras para el aprendizaje.

Con frecuencia el mejor instructor para capacitar al personal sobre el sistema proviene del vendedor que lo proporciona, es decir, de los distribuidores. Este entrenamiento es práctico, ya que los participantes emplean el sistema en presencia de los capacitadotes. Si

surgen preguntas, éstas se responden de manera rápida, ya que se pretende que el sistema se use para capacitación. Esta capacitación se puede realizar en dos lugares:

1. *Instalaciones de los proveedores.* Una de las ventajas de este lugar es que los proveedores cuentan con instalaciones especiales, donde se permite el manejo gratuito del equipo. Otra ventaja es que, en algunas ocasiones, incluyen al personal de muchas compañías que adquieren o emplean el mismo sistema, con lo cual, se comparten preguntas, problemas y experiencias. También existen desventajas, una puede ser que los usuarios se encuentren alejados del contexto de la organización.
2. *Instalaciones de la compañía.* La principal ventaja radica en que la instrucción se puede adecuar a las necesidades de la empresa donde se ofrece, y que se puede centrar en los procedimientos especiales que se emplean, en el crecimiento que se ha planeado y en los problemas que surgen. Una desventaja es, el sólo hecho de que los empleados se encuentren en su lugar común de trabajo constituye una distracción; otra, los adiestrados tienen cargos de conciencia por no cumplir sus obligaciones regulares de trabajo durante las sesiones de adiestramiento.

Otra forma de realizar la capacitación es la utilización de sitios externos de adiestramiento con capacitadores experimentados, los cuales son contratados por los proveedores. Normalmente estas capacitaciones se realizan en lugares propios para reuniones, tales como hoteles. La ventaja de esta forma de capacitación, es que el proveedor se puede desentender completamente de ella. La desventaja es que quizá los capacitados no contarán con un equipo de practica.

Materiales para el adiestramiento.

Este lineamiento incluye los manuales, los cuales pueden adoptar uno de dos enfoques.

El primer enfoque lleva al usuario a trabajar a diferentes actividades por etapas; por ejemplo, una lista de verificación con los pasos necesarios para que el sistema realice una operación específica.

El segundo enfoque constituye los ejemplos para el adiestramiento, en los cuales se les asigna trabajo a los usuarios a través de un caso de estudio que incorpora la mayoría de las interacciones o circunstancias, comúnmente encontradas en el sistema, así como los prototipos y bocetos de salida disponibles.

Conversión.

La conversión es el proceso de cambio del sistema antiguo al nuevo a través de la actualización física.

Métodos de conversión.

Existen cinco métodos para la conversión-actualización de un sistema antiguo a uno nuevo:

1. Reemplazo total.
2. Conversión en paralelo.
3. Conversión gradual.
4. Prototipo modular.
5. Conversión distribuida.

En general, la conversión de sistemas se debe realizar tan rápido como sea posible; los periodos largos de conversión aumentan la posible frustración y dificultad de la tarea para todas las personas que participan, incluyendo tanto a los analistas como a los usuarios.

Reemplazo total.

La conversión por reemplazo total significa que para una fecha específica, el sistema anterior se retira y el nuevo se pone en uso; es decir, el sistema viejo se emplea hasta el día de conversión planeado y entonces se reemplaza por el sistema nuevo.

Una ventaja del reemplazo total inmediato es que los usuarios no tienen posibilidad de utilizar el sistema anterior sino sólo el nuevo.

Lo anterior puede convertirse en una desventaja si surgen problemas serios con el sistema nuevo. Otra desventaja es que no hay manera de comparar los nuevos resultados con los anteriores.

Conversión en paralelo.

Esta se refiere al uso del sistema anterior y del nuevo al mismo tiempo, en paralelo. Ambos sistemas se usan de manera simultánea por un periodo y se examina la confiabilidad de los resultados. Cuando se llega a obtener los mismos resultados, el nuevo sistema se pone en uso y el viejo se descarta. Es el método más seguro de conversión.

Las ventajas que ofrece este método son: la posibilidad de verificar los datos nuevos contra los anteriores, ofrece cierta seguridad a los usuarios quienes no están forzados a cambiar bruscamente hacia el nuevo sistema.

Las desventajas que se pueden presentar son: el costo de operar dos sistemas al mismo tiempo, el agobio de los empleados por tener que duplicar virtualmente su trabajo durante la etapa de conversión, y el hecho de que los usuarios saben que pueden apoyarse en los viejos sistemas puede convertirse en una resistencia potencial para el cambio.

Conversión gradual.

La conversión gradual intenta combinar las ventajas de los dos planes anteriores, el volumen de transacciones se incrementa gradualmente conforme el sistema se va implantando. Este método se emplea cuando no es posible instalar un sistema nuevo en toda la compañía de manera simultánea, por lo cual algunos usuarios obtendrán ventajas del sistema nuevo antes que otros.

Las ventajas que se tienen son: permitir que los usuarios se involucren con el nuevo sistema de manera gradual; la posibilidad de detectar y de recuperarse de los errores; si el sistema funciona bien, los primeros usuarios comunicarán su entusiasmo a otros que esperan la puesta en marcha.

Las desventajas: requerir demasiado tiempo para la implantación del nuevo sistema y su incorrección para convertir sistemas pequeños y poco complicados; si existen problemas al inicio de la puesta en marcha de este método, los usuarios harán comentarios al respecto con otros.

Prototipo modular.

Cuando los sistemas nuevos implican también nuevas técnicas o cambios drásticos en el desempeño de la empresa, con frecuencia, se prefiere el enfoque de un prototipo. Este método considera la construcción de un prototipo modular operativo, para cambiar de manera gradual los viejos sistemas por unos nuevos. Conforme se modifica cada módulo y se acepta, se pone en operación.

Una ventaja es que cada módulo se prueba completamente antes de utilizarse. Otra es que los usuarios se familiarizan con los módulos conforme éstos se vuelven operativos.

Una desventaja es que tal vez los prototipos no sean factibles para la empresa, por lo cual se descarta este método. Otra es que se debe prestar atención especial a las interfaces, de tal forma que los módulos, al ser desarrollados independientemente, actúen como todo un sistema.

Conversión distribuida.

Esto se refiere a una situación en la cual se contemplan muchas instalaciones del mismo sistema. Una conversión se realiza en uno de los sitios (con cualquiera de los cuatro métodos anteriores). Cuando esta conversión se concluye con éxito, se realizan otras conversiones en los otros sitios.

El hecho de que los problemas puedan detectarse (y manejarse), más que inducirse a todos los demás sitios, es una ventaja.

Una desventaja puede ser que cada sitio tendrá sus propias peculiaridades, las cuales deberán resolverse.

La Figura 4-1 muestra gráficamente como sería la conversión de un sistema utilizando cada uno de los métodos.

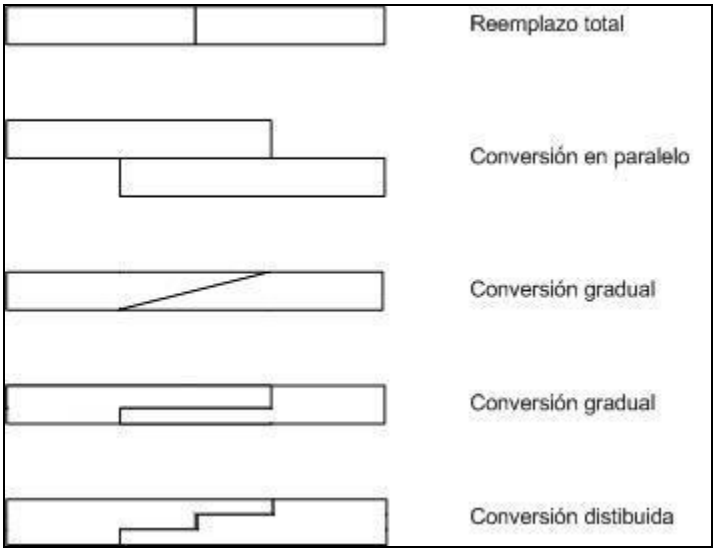


Figura 4- 1 Métodos de conversión para sistemas informáticos.

Plan de conversión.

El plan de conversión incluye una descripción de todas las actividades que deben llevarse a cabo para poner en práctica el sistema nuevo y ponerlo en operación; asimismo, identifica a las personas que son responsables de cada actividad e incluye una distribución de horarios que estipula cuando ocurrirá cada actividad.

El plan de conversión debe anticipar los posibles problemas y métodos para enfrentarlos.

La determinación de tiempos para la conversión es difícil, ya que existen muchos que varían desde la instalación del equipo hasta la recepción de formas y suministros.

Seguridad.

La seguridad de los servicios de cómputo, así como la seguridad de los datos almacenados y de la información generada forman parte de una implantación con éxito.

La seguridad es responsabilidad de todos aquellos que están en contacto con el sistema dentro de la organización. Tiene tres aspectos interrelacionados: físicos, lógicos y de conducta, los cuales debe operar en conjunto si se quiere que los estándares de calidad en seguridad permanezcan altos.

Seguridad física.

La seguridad física se refiere a las instalaciones de cómputo, a nuestro equipo y al software, a través de elementos físicos.

Las decisiones referentes a la seguridad física deben realizarse en el momento mismo en que el analista planee las instalaciones de cómputo y la adquisición del equipo.

Seguridad lógica.

El concepto de seguridad lógica se refiere a los controles lógicos dentro del software. Los controles lógicos son conocidos por la mayoría de los usuarios, como contraseñas o códigos de autorización. Cuando se utilizan, permiten que el usuario entre al sistema o a alguna sección en particular.

Conducta del usuario.

La conducta interna de los miembros de la organización es decisiva para el buen éxito de los esfuerzos de seguridad.

Una manera de supervisar si las personas no autorizadas intentan utilizar el sistema es hacer que el sistema registre el número de intentos frustrados de buscar acceso al sistema sin éxito; además, de investigar aquellos accesos al sistema en sesiones extremadamente largas o en horarios no usuales.

El control de la salida incluye las pantallas que pueden accederse a través de contraseñas; la clasificación de la información y el almacenamiento seguro de los documentos impresos o almacenados en material magnético.

Otras consideraciones.

La conversión también comprende otros detalles que deben incluirse:

- **Ordenar equipo**, hasta tres meses antes de la conversión.
- **Ordenar cualquier material,** necesario que provea externamente al sistema de información, tal como: cintas, papel, y formas preimpresas.
- **La asignación de un gerente,** para supervisar personalmente la preparación del sitio de la instalación.
- **La planeación,** programación y supervisión de los programadores y del personal de captura que deberá convertir todos los archivos y la base de datos.

Evaluación.

A todo lo largo del ciclo de vida del sistema, los analistas directivos y usuarios han evaluado la evolución del sistema informativo. La evaluación se realiza también para darle seguimiento a la implantación del sistema. Esta evaluación es importante para obtener información que sirve al mantenimiento del sistema, a través de la respuesta a ciertas preguntas como son: quién, qué, dónde, cuándo, cómo, y por qué.

Se han desarrollado múltiples técnicas de evaluación. Estas técnicas incluyen el análisis costo-beneficio; los modelos de estimación del valor de una decisión, con base en los efectos de la información, por medio de la aplicación de la teoría de la información; las evaluaciones del usuario que destacan los problemas de implantación; el compromiso del usuario y los enfoques de utilería de información que examinan las propiedades de la información.

Cada tipo de evaluación satisface un propósito diferente, si bien, contando también con ciertos inconvenientes implícitos. Los análisis costo-beneficio pueden ser difíciles de aplicar, ya que los sistemas de información proporcionan información de los objetivos por vez primera, haciendo imposible comparar el desempeño antes y después de la implantación del sistema. El enfoque de evaluación de decisiones presenta algunas dificultades ya que todos los valores involucrados en el diseño, desarrollo e implantación del sistema no pueden calcularse o cuantificarse. El enfoque del compromiso del usuario da cierta pauta para nuevos proyectos al proporcionar una lista de características deseadas. El enfoque de utilerías del sistema para la evaluación puede ser más comprensivo que otros si se expande y aplica de manera sistemática.

Enfoque de utilerías.

Las utilerías de la información incluyen la posesión, forma, lugar y tiempo. Con el fin de evaluar de manera comprensiva al sistema de información, estas utilerías deben ampliarse para incluir las utilerías de actualización y las utilerías de objetivos. Después, las utilerías pueden verse dirigidas adecuadamente a las preguntas de quién (posesión), qué (forma), dónde (lugar), cuándo (tiempo), cómo (actualización) y por qué (objetivo).

Utilería de posesión.

La utilería de posesión contesta la pregunta: ¿Quién debe recibir la salida? La información carece de valor en manos de alguien que carece de poder para llevar a cabo mejoras en el sistema o de alguien que carece de la habilidad para utilizar la información.

Utilería de forma.

La utilería de forma contesta la pregunta: ¿Qué tipo de salida se distribuye entre quienes toman las decisiones? El documento debe ser útil para un tomador de decisiones en particular, considerando el formato y el lenguaje que utilice.

Utilería de lugar.

La utilería de lugar contesta la pregunta: ¿Dónde debe distribuirse la información? La información debe llevarse al mismo sitio donde se realiza la toma de decisión.

Utilería de tiempo.

La utilería de tiempo contesta la pregunta: ¿Cuándo debe proporcionarse la información? La información debe llegar anticipadamente al momento de la toma de la decisión.

Utilería de actualización.

La utilería de actualización considera cómo se introduce la información y cómo se utiliza por el tomador de decisiones. Primero, el sistema de información cuenta con valor si posee la cualidad para ser implantado. Segundo, la utilería de actualización implica que un sistema de información tendrá valor si se mantiene después de que sus diseñadores hayan partido o si un uso único del sistema de información proporciona resultados satisfactorios y duraderos.

Utilería de objetivo.

La utilería de objetivo contesta el porqué del sistema de información al preguntar si las salidas tienen algún valor para auxiliar a la organización para alcanzar sus objetivos.

Evaluación del sistema.

Un sistema de información puede evaluarse como exitoso si posee las seis utilerías anteriores. Si el módulo del sistema se juzga como "pobre" al proporcionar una de las utilerías, el modelo completo esta destinado al fracaso. Un intento parcial de una utilería puede resultar en un módulo con un éxito parcial. Si el módulo del sistema de información se juzga como "bueno" al proporcionar cada utilería, el módulo tendrá éxito.

El enfoque de utilerías de sistemas de información representa una estructura operativa y bien definida para evaluar proyectos de gran escala de sistemas de información, y los resultados de los esfuerzos del centro de operación.

Como se puede apreciar, el interés fundamental durante la evaluación es contestar varias preguntas, las cuales nos permiten determinar si el sistema cumplió con su objetivo, la calidad de las salidas, la facilidad de uso de las salidas y que contengan la suficiente información, la facilidad de uso y la tendencia a la producción de errores en la entrada del sistema, y la confianza que el sistema genera en el usuario.

Implantación de la Aplicación desarrollada.

En lo que se refiere al centro de información para implantar la Aplicación, no fue necesario realizar ninguna actividad; ya que esta aplicación pertenece a un sistema mayor, con lo cual, al cubrir los requerimientos necesarios para implantar el sistema se cubren implícitamente los requerimientos de la Aplicación, por lo que no se requieren características especiales o determinadas, dentro del centro de información, para su funcionamiento.

El adiestramiento, para el uso de la Aplicación, se dirigió a los usuarios con perfil "Administrador del sistema", ya que son ellos los que únicamente tenían la facultad para poder realizar la descarga de información del sistema y la carga de información al sistema, en el curso de "Administrador de sistema". El curso "Administrador de sistema" se impartió por el personal que se encargo de desarrollar el sistema, ya que solo ellos tenían el conocimiento técnico necesario para impartirlo. El curso se realizaba por medio de un ponente, el cual iba dando una platica sobre las características de "la administración del sistema" y de todas las actividades que se podían realizar en esta parte del sistema de información, y lo llevaba a cabo utilizando una presentación, la cual contenía los conceptos necesarios para entender "la administración del sistema" y resaltar los puntos importantes de la misma administración, al igual que una practica después de la explicación de un tema, esta practica se realizaba inmediatamente sobre el sistema. El adiestramiento se impartía en las instalaciones del proveedor, ya que generalmente los administradores del sistema eran una o dos personas dentro de la organización del cliente, y no se justificaba la planeación y realización de un curso en las instalaciones del cliente; además, en las instalaciones del proveedor se planeaban e impartían los cursos no solo a un administrador de sistema de un cliente, sino a varios administradores de diferentes clientes al mismo tiempo, también ya se contaba con una sala preparada para dicha actividad. Dentro del curso se entregaba un manual sobre el mismo, que apoyaba la realización de éste; también, al entregarse el sistema se hacía entrega de un manual que solamente hacía referencia a las actividades y al módulo de "Administrador del sistema".

La conversión del sistema se realizó a través del método de "conversión distribuida", ya que cuando se termino de desarrollar la Aplicación, objeto de este trabajo de tesis, el sistema para el cual se hizo, se encontraba trabajando con varios clientes. La forma en que se implantó la Aplicación en los sistemas que se encontraban en funcionamiento, fue por medio del método de "reemplazo total", ya que se realizó una actualización de la versión del sistema que incluía la Aplicación por aquella que no la tenía. En el caso de las implantaciones del sistema, posteriores al término de la Aplicación, no fue necesario realizar ninguna conversión.

La evaluación que se realizó a la Aplicación desarrollada se basó en el enfoque de utilerías, con el cual se obtuvieron los siguientes resultados:

Utilería de posesión. La información que generaba la Aplicación es entregaba a un usuario "administrador del sistema", ya que eran ellos los que podían realizar un mejor uso y explotación de ella, debido a los conocimientos que se requerían para hacerlo.

Utilería de forma. El formato en que la información se entregaba, a quien la podía utilizar, era electrónico, por medio de archivos de datos con ciertas características que permitían su manipulación a través de otros programas.

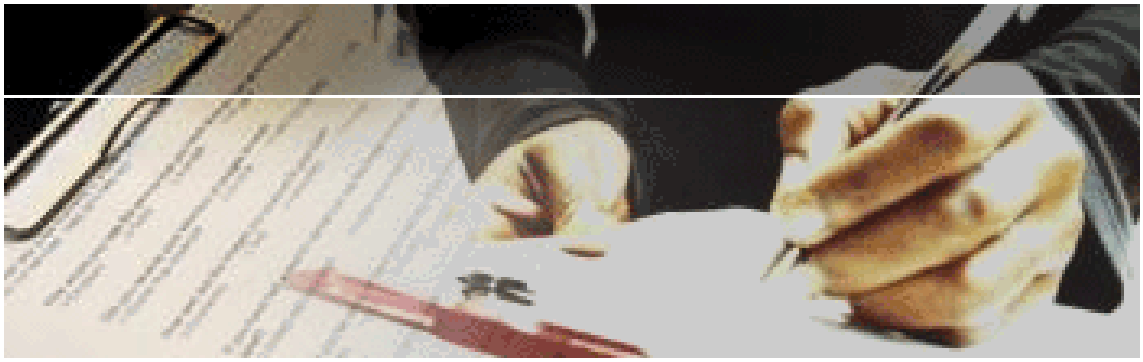
Utilería de lugar. La información se obtenía de la Aplicación, en el más alto nivel de administración del sistema, debido a que solamente algunas personas podían acceder a este nivel, y normalmente eran ellas quienes podían obtener provecho de ella.

Utilería de tiempo. La Aplicación podía entregar información en el momento en que ésta se ejecutaba, ya que se desarrolló para tal efecto.

Utilería de actualización. No fue necesario realizar una actualización a la Aplicación, ya que la versión del sistema en la que integro fue la última de su producción; por lo que, al no realizarse cambios en la estructura del sistema para el cual se desarrollo la Aplicación, tampoco fue necesario realizar ajustes o mantenimientos a ésta.

Utilería de objetivo. El objetivo por el cual se desarrolló la Aplicación fue satisfecho al 100%; con lo cual, la organización para la que se hizo el desarrollo logro obtener portabilidad en la información de su sistema, sin requerir de grandes esfuerzos o de instrucciones demasiado especializadas.

Conclusiones.



Al final, los objetivos que se plantearon y por los cuales se desarrollo este trabajo, fueron cumplidos en su totalidad. Lo anterior basado en que con las aplicaciones que fueron analizadas, diseñadas, desarrolladas e implantadas se logro darle la característica de la portabilidad al sistema de información que lo requirió.

Existieron algunas características, posteriores a la finalización del desarrollo de la única versión, que mejoraron la funcionalidad de los procesos de las aplicaciones desarrolladas. Estas características fueron obtenidas a través de la observación de la operación de las aplicaciones, de entrevistas con los usuarios que las emplearon, y del rediseño de las mismas aplicaciones. Estas características más la única versión de las aplicaciones desarrolladas sirvieron como bases y fundamentos para el desarrollo de nuevas aplicaciones y sus nuevas versiones que se incluyeron en nuevos sistemas a los cuales se les implantó la portabilidad. A continuación se listan las características más relevantes:

- La selección de varias empresas, sin necesidad de que sean todas.
- El uso de un archivo que describa la base de datos.
- Incluir en los archivos generados el tipo de dato que es cada columna.
- Indicar en un área de mensaje mayor detalle sobre los problemas ocurridos.
- Implementar la recursividad en los procesos de carga.

Como ya se mencionó, la vida útil de las aplicaciones desarrolladas fue mientras se empleó la versión del sistema de información al que se le implantó la portabilidad, ya que no hubo desarrollo de versiones posteriores para el dicho sistema. Sin embargo, la funcionalidad de la portabilidad fue requerida por otros sistemas, como se menciono en el párrafo anterior.

Con el desarrollo de nuevas tecnologías, actualmente, se tiene mayor opción para lograr el objetivo de este trabajo. Entre las nuevas tecnologías obviamente se puede mencionar las nuevas versiones de las JVM, con lo cual obtenemos más procesos desarrollados y lo cual hace que se empleen estos en lugar de desarrollar los propios. Otra de las tecnologías que se pueden emplear es el uso de archivos XML como descriptores de la base de datos, con esto se lograría una mejor estructuración de los propios elementos del archivo y también mayor entendimiento del diseño de la base y elementos que la conforman.

Por último se puede mencionar que la portabilidad de los sistemas (movimiento de datos) es cada día mas requerida por los usuarios de los mismos; es decir, que los usuarios, llámese empresa, persona, etc., de un sistema no tengan que tener relación forzosa con una sola tecnología de base de datos, y que sean ellos los que decidan cual les conviene y se ajusta más a sus necesidades y posibilidades. Por lo anterior, el sistema de información que tenga como una de sus características la portabilidad de datos tendrá mayor ventaja sobre sus competidores al momento de ser evaluado.

Bibliografía.

- [BERTINO Y MARTINO 95] **BERTINO, Elisa, y MARTINO, Lorenzo**, "Intelligent Database System", Madrid, España, Addison-Wesley, 1995.
- [DENNIS Y HALEY 00] **DENNIS, Alan**, **y HALEY Wixom, Barbara**, "System Analysis and Desing.", USA, John Wiley & Sons, 2000.
- [KENDALL Y KENDALL 91] **KENDALL, Kenneth E., y KENDALL, Julie E.,** "Análisis y Diseño de Sistemas.", Prentice Hall. 1991.
- [KULAK Y GUINEY 03] **KULAK, Daryl, y GUINEY, Eammon**, **"**Use Cases: Requirements in Context**"**, 2a ed. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2003. 272 p.
- [KULAK Y GUINEY 03:c2:s2.3] **KULAK, Daryl, y GUINEY, Eammon**, **"**Use Cases: Requirements in Context**"**, 2a ed. 2003. Cápitulo 2 Sección 2.3
- [LOPEZ Y QUERO 98] **LOPEZ HERRANDEZ, José, y QUERO CATALINAS, Enrique,** "Fundamentos de Programación.", Paraninfo.
  1998.
- [MARAKAS 00] **MARAKAS, George M.,** "System Analysis and Design.", Prentice Hall. 2000.
- [ROWLEY 94] **ROWLEY, Jennifer**,"The Basic of System Analysis and Design.", Clive Bingley London, 1990.
- [SEBESTA 04] **SEBESTA, Robert W.,** "Programming Languages." Addison Wesley. 2004.
- [SENN 98] **SENN, James A.**, "Análisis y Diseño de Sistemas de Información.",Mc. Graw Hill, 1998.
- [WHITTEN Y BENTLEY] **WHITTEN, Jeffrey L., y BENTLEY, Lonnie D.,** "System Analysis and Desing Methods.", Mc Graw Hill.1998.
- Acey. Evolución histórica de los lenguajes de programación [en línea]. España. 2006.
  Disponible en:
  http://pdf.rincondelvago.com/evolucion-historica-de-los-lenguajes-de-programacion.html
- Bases de datos [en línea]. Madrid, España: Universidad Complutense de Madrid. 2006
  Disponible en:
  http://pdf.rincondelvago.com/bases-de-datos_22.html
- Campo (informática) [en línea]. 2004
  Disponible en:
  http://es.wikipedia.org/w/index.php?title=Campo_%28inform%C3%A1tica%29&action=history
- Carola. Computadoras [en línea]. Venezuela. 2006
  Disponible en:
  http://apuntes.rincondelvago.com/computadoras.html
- CISTERNA Neira, Mario. Métodos de optimización de consultas para el lenguaje SQL [en línea]. Santiago, Chile: Universidad de Santiago de Chile. 2002
  Disponible en:
  http://macine.epublish.cl/tesis/
- CHEN, Doris. Servlets and JavaServer Pages [en línea]. 2006

Disponible en:
http://developers.sun.com/events/techdays/presentations/dchen/index.html

- Dato [en línea]. 2003
  Disponible en:
  http://es.wikipedia.org/wiki/Dato
- Declarative programming [en línea]. 2003
  Disponible en:
  http://en.wikipedia.org/wiki/Declarative_programming
- DUQUE Méndez, Néstor Dario. Bases de Datos [en línea]. Colombia: Universidad Nacional de Colombia, 2006.
  Disponible en:
  http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/index.html
- Functional programming [en línea]. 2001
  Disponible en:
  http://en.wikipedia.org/wiki/Functional_programming
- Gonis. Base de datos [en línea]. 2002.
  Disponible en:
  http://es.wikipedia.org/wiki/Base_de_datos
- JavaBeans FAQ: General Questions [en línea]. 2006.
  Disponible en:
  http://java.sun.com/products/javabeans/faq/faq.general.html
- Imperative programming [en línea]. 2003
  Disponible en:
  http://en.wikipedia.org/wiki/Imperative_programming
- LAGO García, Javier. Dispositivos de almacenamiento [en línea]. España. 2006.
  Disponible en:
  http://pdf.rincondelvago.com/dispositivos-de-almacenamiento_1.html
- Logic programming [en línea]. 2001
  Disponible en:
  http://en.wikipedia.org/wiki/Logic_programming
- Matrix. Informática administrativa [en línea]. México: U.A.E.M. 2006
  Disponible en:
  http://pdf.rincondelvago.com/infomatica-administrativa.html
- Normalizacion de una base de datos [en línea]
  Disponible en:
  http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_una_base_de_datos
- Programación lógica [en línea]. 2005
  Disponible en:
  http://es.wikipedia.org/wiki/Programaci%C3%B3n_l%C3%B3gica
- Registro (base de datos) [en línea]. 2006
  Disponible en:
  http://es.wikipedia.org/wiki/Registro_%28base_de_datos%29
- RODRIGUEZ, Héctor Julio. Historia de las bases de datos en Ciencia de la Información [en línea]. Bogotá, Colombia: Pontificia Universidad Javeriana, 2006.

Disponible en:
http://recursostic.javeriana.edu.co/wiki/index.php/Historia_de_las_bases_de_datos_en_Ciencia_de_la_Informaci%C3%B3n

- ROSSEL, Gerardo. Programación Lógica.
  Disponible en:
  http://www.amzi.com/articles/code07_whitepaper.pdf
- SpeedyGonzalez. Base de datos jerárquica [en línea]. 2003
  Disponible en:
  http://es.wikipedia.org/wiki/Base_de_datos_jer%C3%A1rquica
- SQL [en línea]. 2001
  Disponible en:
  http://en.wikipedia.org/wiki/SQL