UNIVERSIDAD TECNICA DE BABAHOYO

Facultad de Administración, Finanzas e Informática.

Tesis

Tema:

Sistema de gestión comercial para mejorar las ventas en ferrecentro oñate s.a. de la ciudad de Babahoyo**".**

Autores:

Rita Roxana Ortega Bustamante.

Wilson Enrique Muñoz Ponce.

Babahoyo

2013

CERTIFICACIÓN DE AUTORÍA

La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis me corresponden exclusivamente, y el patrimonio intelectual de la misma a la Escuela de Sistemas e Informática de la Universidad Técnica de Babahoyo.

_____

Rita Roxana Ortega Bustamante

_____

Wilson Enrique Muñoz Ponce

FIRMAS DE PERSONALIDAD

El jurado calificador de la Escuela de Sistemas-Facultad de Administración Finanzas e Informática de la Universidad Técnica de Babahoyo le da al siguiente proyecto de tesis.

La calificación de _____

Equivalente a _____

Fecha: _____

Firma para corroborar su veracidad:

Presidente Tribunal de Defensa _____

Director de la Tesis _____

Lector de la Tesis _____

Secretario _____

INDICE GENERAL

CAPITULO I

EL PROBLEMA

CAPITULO II

MARCO TEÓRICO

CAPITULO III

MARCO METODOLÓGICO

CAPITULO IV

MARCO PROPOSITIVO

CAPITULO I

1.1 Planteamiento del Problema.

Con el paso del tiempo la tecnología avanza, las empresas se sienten en la necesidad de adquirir tecnología para el mejoramiento de sus sistemas y a la vez sus procedimientos, con el fin de garantizar un eficaz funcionamiento y así obtener una adaptación paralela de condiciones con las empresas líderes del mercado.

En tiempos pasados, se discutían las dificultades planteadas por los sistemas de información se refería a las personas, no a la tecnología de la información, luego se ha intentado despertar el interés ante las posibilidades que ella brinda también formular que se ha desmontado y que se puede resolver la mayoría de los problemas técnicos, pero para quienes comprenden que la innovación basada en la tecnología es al fin un proceso social, cuando, muchos años más tarde, se examinan las predicciones hechas con referencias al cambio tecnológico, resulta impresionante su penetración y la incorporación de la tecnología de la información a la vida cotidiana pues al fin, su influencia en el hombre no se ha hecho esperar y el ser humano ha tomado amplia conciencia de la profunda alteración que la utilización no sólo de computador sino del conjunto de técnicas de tratamiento de la información derivadas de uso ha llevado a construir los restos del futuro.

Se ha podido observar que durante todos los procesos que se efectúan en Ferrecentro Oñate S.A de la ciudad de Babahoyo, es deficiente puesto que, cuando un cliente solicita su mercancía el empleado debe ir al sitio a verificar si la mercancía existe o no, lo que ocasiona el descontento en los mismos al esperar mucho tiempo para saber su existencia, no existe un control de registro clientes, proveedores ni de empleados ya que su sistema es totalmente manual y para no acumular más documentos no lo hacen. Al momento de cotejar la mercancía existente para posterior efectúo de pedidos es muy tedioso, ya que tienen que contar una por una cada una de la mercancía y se pierde mucho tiempo valioso que se pueden invertir en otras operaciones de igual manera al hacer las facturas y presupuesto.

De continuar funcionando de esta manera las operaciones del Ferrecentro Oñate S.A. como sigue actualmente sus actividades no podrán ser aligeradas, al contrario se obstaculizara más el trabajo debido al aumento de la clientela y la cantidad de información que será aún mayor.

Debido a esto se propone sistematizar las actividades de compra y venta de mercancía de dicha empresa y así mejorar su eficacia al trabajar de manera más rápida y segura.

1.2 Formulación del Problema

¿Cómo mejorar las ventas en Ferrecentro Oñate S.A. de la ciudad de Babahoyo a través de un Sistema Informático?

1.3 Justificación

Ferrecentro Oñate S.A. una Empresa que centra su actividad en la formación integral de sus clientes en el ámbito científico-técnico articulando crítica, innovadora y creativamente sus métodos de trabajo, investigación y proyección social.

Ferrecentro Oñate debe contar con un sistema informático que al carecer no lo tiene, se plantea entonces la necesidad de construir, con base sólida, un proceso regular y continuo que trabaje con un software desarrollado en la plataforma Visual.net 2008, con la modalidad de ejecutar un motor de base de datos en SQL server, lo cual garantiza la seguridad de la información y su integración con el sistema a implementar.

Se pretende con esta investigación satisfacer, al 100% la problemática actual de control ingreso, egreso y de ventas de productos que expende Ferrecentro Oñate S.A. Se deberá realizar un estudio acerca del problema y realizar una investigación profunda acerca de los diferentes medios de captura de datos para el control del mismo, los cuales deben estar adecuados a la infraestructura de la Empresa y que sean adaptables a un Sistema de aplicación.

Al realizar el Diseño e Implementación de este sistema en Ferrecentro Oñate, nos permitirá llevar a cabalidad la realización de este proyecto que puede generar un valor adicional a los conocimientos adquiridos durante la investigación, establecer unas bases que permitan el desarrollo integral de un profesional que se desempeñe a nivel laboral en las diferentes áreas de la ingeniería que se muestran en la elaboración de este proyecto.

1.3 Delimitación

Esta investigación se la desarrollará en La empresa Ferrecentro Oñate de la Ciudad de Babahoyo está ubicada en la avenida 5 de junio y Pedro Carbo.

Objeto de estudio: Ingeniería de Sistemas.

Campo de acción: Sistema de información

Temporal

De Mayo del 2013 a Septiembre del 2013

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un Sistema de Gestión Comercial para mejorar las ventas en Ferrecentro Oñate S.A. de una manera rápida, organizada y sencilla beneficiando tanto al personal como a los clientes de la empresa.

1.4.2 Objetivos Específicos

- ✓ Fundamentar la base teórica y científica que permitan el desarrollo de esta investigación.
- ✓ Analizar y preparar información para conocer las mejores soluciones.
- ✓ Validar la investigación y resultados con la ayuda de un experto.

CAPITULO II

2.1 Marco Teórico

El presente capítulo se divide en dos partes, el objetivo de la primera es describirla necesidad de la Ferretería Ferrecentro Oñate S.A. de contar con un Sistema de Bases de Gestión Comercial para mejorar las ventas de esta empresa que le permita lograr la eficiencia en el control de los productos con los que comercializa, así como también llevar un control de su nómina, y el objetivo de la segunda parte es dar un marco teórico de las bases de datos y de la ingeniería de software.

2.2 La Empresa

La empresa denominada Ferretería Ferrecentro Oñate S.A. surge en el mes de Mayo del 2003. La misión es competir en el ramo ferretero, proporcionando al cliente artículos de calidad en plomería, material eléctrico, herramienta en general, etc. de marcas nacionales y extranjeras.

La función principal de la empresa para la cual será elaborada el sistema de ventas es la de realizar la venta de productos divididos en categorías de Herramientas Eléctricas, Herramientas Manuales, Plomería, Electricidad, Iluminación, Muebles, Organización, Hogar, Accesorios para baños y cocinas, Jardinería, Mascotas, Pintura, Cerrajería, Hogar y Limpieza.

2.3 Antecedentes

Desde su creación, la empresa ha tenido como uno de sus objetivos principales satisfacer las necesidades del cliente y hacer una cartera de clientes cada vez mayor. En relación al control de inventario, la información se elaboraba manualmente en archivos de papeles, y en consecuencia generaba una labor tediosa encontrar información, ocasionando demoras en los servicios.

Refiriéndose a la nómina, el control de los empleados se llevaba a cabo poniendo toda la responsabilidad en la capacidad de memoria del gerente, que como consecuencia, generaba conflictos en los pagos.

El control de la cartera de clientes se realizaba en un archivo de Excel en el cual se anotaba el nombre, teléfono, dirección y pedidos del cliente, con la dificultad de que la información continuaba dispersa y para poder surtir un producto se tenía que consultar a varias fuentes, de las cuales el acceso no era sencillo, ya que ésta computadora se encuentra en la planta alta del local de la empresa.

2.4 Visión de la Empresa

Ferrecentro Oñate será una Empresa moderna de reconocido prestigio y credibilidad, líder y rectora del desarrollo organizacional, constituyéndose en referente válido de la Gestión Pública, técnica y transparente por los servicios de calidad que presta a sus usuarios.

2.5 Misión de la Empresa

Ejercer la rectoría en el diseño y ejecución de políticas de desarrollo organizacional para generar servicios de calidad, contribuyendo a incrementar los niveles de competitividad, productividad, empleo y satisfacción de Atención a los usuarios.

2.6 Objetivo General de la Empresa

Satisfacer las necesidades del usuario con una buena atención, garantizando esta manera la calidad de los productos que ofrece la Empresa.

2.7 Organización de la Empresa

La empresa organiza a su personal de la siguiente manera:

1. GERENTE: Encargado de dirigir al personal y autorizar todas las operaciones dentro de la empresa y de administrar los diferentes recursos de la misma.

FUNCIONES

- **a)** Iniciar operaciones
- **b)** Revisar agenda de cobros y pagos
- **c)** Iniciar registro de caja
- **d)** Atención a proveedores
- **e)** Hacer o verificar el correcto corte de caja
- **f)** Elaborar cartera de clientes
- **g)** Realizar operaciones bancarias
- **h)** Supervisión de inventario

i) Revisión del ingreso de mercancía y su facturación

j) Autorización de movimientos materiales o financieros

2. **SECRETARIA**: Encargada de las labores administrativas.

FUNCIONES

a) Organizar agenda

b) Contactar proveedores

c) Revisar cobros de clientes y pagos a proveedores

d) Atención telefónica en general

e) Control de remisiones y facturas de efectivo y crédito

f) Relación de gastos por facturación de proveedores

g) Captura de información

3. **CAJERO**: Encargado de cobros en mostrador.

FUNCIONES

a) Iniciar registro de caja

b) Hacer corte de caja diario

c) Atención de clientes en el mostrador en caso de ser necesario

d) Control de remisiones y facturas

4. EJECUTIVO DE VENTAS: Encargado de ventas en general.

FUNCIONES

a) Venta de productos en mostrador o por teléfono

b) Seguimiento de cartera de clientes

c) Emisión de facturas y remisiones

d) Entrega de mercancía

e) Elaboración de inventario (manual)

f) Recibir y organizar mercancía de nuevo ingreso

g) Mantenimiento del lugar de trabajo y medios de transporte del mismo.

5. **COBRADOR**: Encargado de cobros a clientes

FUNCIONES

- **a)** Mantenimiento del lugar de trabajo
- **b)** Realizar ruta de cobros a clientes
- **c)** Compra y recibo de mercancía
- **d)** Reparto de mercancía a clientes
- **e)** Atención de clientes en mostrador en caso necesario
- **f)** Organización de productos en el almacén y en el mostrador

6. VIGILANTES: Encargado de la seguridad del lugar de trabajo.

FUNCIONES

- **a)** Asegurar el lugar de trabajo
- **b)** Revisar los dispositivos de seguridad (malla eléctrica, candados y puertas)
- **c)** Mantener el espacio destinado a estacionamiento.

2.8 Recursos y Materiales de la empresa

- ✓ Dos camionetas
- ✓ Una caja registradora
- ✓ Fax
- ✓ 4 líneas telefónicas
- ✓ Un conmutador
- ✓ Tres equipo de cómputo.
- ✓ Una impresora.
- ✓ Papelería en general.

2.9 Necesidad de controlar los procesos de negocio.

Una necesidad básica es dar al cliente una atención rápida, lo cual requiere de tener en una o varias computadoras una base de datos con los productos disponibles para evitar la pérdida de tiempo en las búsquedas de existencia de los productos así como sus características, que incluyen el precio, marca, color, etc.

2.10 Principales Procesos de Negocio de la empresa.

¿Qué son los procesos de negocio?

Los procesos de negocio consisten en las tareas que debe realizar una empresa si quiere seguir funcionando: realizar ventas, pagar a empleados y acreedores, controlar el inventario, etc.

- ✓ Ventas
- ✓ Pago de nómina
- ✓ Pagos a acreedores
- ✓ Altas a nuevos trabajadores
- ✓ Altas a nuevos clientes
- ✓ Control de Inventario

2.11 Beneficios de la base de datos a la empresa.

Las ventajas de un sistema de base de datos sobre los métodos tradicionales de mantener los registros en papel es que una base de datos es compacta, rápida, menos laboriosa y actual.

Con todo, existe una ventaja adicional: El sistema de Base de Datos ofrece a la empresa un control centralizado de su información.

La información es uno de los recursos más valiosos de una empresa. Así en un comercio sin un sistema de este tipo, cada aplicación tiene por lo regular sus propios archivos privados de manera que los datos están muy dispersos y con seguridad son difíciles de controlar en cualquier forma sistemática.

2.12 Bases de Datos

¿Qué es un Sistema de Base de datos?

Un Sistema de Base de Datos es básicamente un sistema computarizado para llevar registros. Es posible considerar a la propia Base de Datos como una especie de armario

electrónico para archivar, es decir, un depósito o un contenedor de una colección de archivos de datos computarizados.

2.12.1 Componentes de un Sistema de Bases de Datos.

A continuación se mencionarán los cuatro principales componentes en un sistema de base de datos:

- La información
- El equipo
- Los usuarios
- Los programas

La información. En general, la información en la base de datos estará integrada y además será compartida.

El equipo. Se considera que son componentes del equipo del sistema:

Medios de almacenamiento secundario: Dispositivos E/S asociados, Drives, Canales de E/S, etcétera, Procesador o procesadores y memoria principal.

Los usuarios: Es todo el personal del departamento que requiera usar el sistema de base de datos para implementar, consultar o realizar sus reportes. Se tienen diferentes tipos de usuarios, entre los cuales tenemos a los programadores de aplicaciones; los cuales son los responsables de escribir los programas de aplicación; los usuarios finales, quienes interactúan con el sistema desde estaciones de trabajo o terminales; y finalmente el Administrador de la Base de Datos (DBA), y es quien administra la base de datos.

Los programas. Existe una capa de programas entre la base de datos física misma y los usuarios del sistema: el Sistema de Administración de Base de Datos (DBMS, Data Base ManagmentSystem).

El **DBMS** maneja todas las solicitudes de acceso a la base de datos formuladas por los usuarios.

2.12.2 El Sistema de Administración de Bases de Datos (DBMS)

El software que permite a una o más personas el usar y/o modificar los datos de una base de datos se denomina Administrador de Base de Datos (DBMS).

Maneja todas las solicitudes de acceso a la base de datos formuladas por los usuarios.

Seguridad: no todos los usuarios tienen acceso a todos los datos.

Integridad: cierto tipo de "consistencia" deberá realizarse sobre los atributos y valores de los datos, para evitar la inconsistencia en los datos.

Sincronización: Cuando varios usuarios corren programas que accedan a la base de datos al mismo tiempo, el DBMS deberá dar protección de inconsistencias que puedan resultar de dos operaciones simultáneas a un mismo grupo de datos.

Protección de rupturas y recuperación: facilidades para realizar copias regulares de la base de datos y reconstruirla después de un error de hardware o software.

Uno de sus objetivos más importantes es proporcionar a los usuarios una visión abstracta de los datos, es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos, pero sin embargo se deben extraer eficientemente.

2.12.3 El Modelo Relacional

El modelo relacional de las bases de datos representa la tendencia dominante en el mercado actual debido al avance que ha tenido en el campo de las bases de datos principalmente en el modelo relacional, por esta razón la tecnología de bases de datos se orienta hacia los sistemas relacionales, por ende, en ésta parte del capítulo se explicará de manera general los conceptos fundamentales para los sistemas relacionales de bases de datos.

Una base de datos relacional es una base de datos percibida por el usuario como una colección de relaciones normalizadas [Boone 2000].

El modelo relacional se ocupa de tres aspectos:

- ✓ ***Estructura de datos:*** El usuario percibe la información de la base de datos como tablas y nada más que tablas. La estructura de datos es la terminología que se utiliza como notación y forma de representar la información en el modelo relacional.
- ✓ ***Integridad:*** Las tablas deben satisfacer ciertas restricciones de integridad.

  La integridad se basa en dos reglas en el modelo relacional:
- ✓ Regla de la Integridad de las Entidades: Ningún componente de una relación base puede tener valores nulos.
- ✓ La base de datos no debe contener valores de llave ajena sin concordancia.
- ✓ ***Manipulación:*** Los operadores disponibles para que el usuario manipule estas tablas son operadores que derivan tablas a partir de tablas. En particular, tres de estos operadores son importantes: restringir, proyectar, juntar.

- ✓ La operación restringir extrae las filas especificadas de una tabla.
- ✓ La operación proyectar extrae las columnas especificadas de las tablas.
- ✓ La operación juntar reúne dos tablas con base en valores comunes de al menos una columna en común.

2.12.4 Ingeniería de Software

La Ingeniería del Software se podría definir como el establecimiento y aplicación de principios de la Ingeniería para obtener software. Teniendo en cuenta factores tan importantes como el coste económico, la fiabilidad del sistema y un funcionamiento eficiente que satisfaga las necesidades del usuario.

El modelo del ciclo de vida es un factor principal para conseguir los objetivos buscados, una mala elección del modelo de ciclo de vida puede hacer que se nos retrase el trabajo enormemente o que tengamos una planificación perfecta del trabajo.

2.12.5 Visionamiento del Software

Modelo de cascada

En un modelo en cascada un proyecto progresa a través de una secuencia ordenada de pasos que son:

- ✓ Concepto del software.
- ✓ Análisis de requerimientos.
- ✓ Diseño global.
- ✓ Diseño detallado.
- ✓ Codificación y depuración.
- ✓ Prueba del sistema.

El modelo contiene una serie de etapas que no se solapan, y el proyecto se va revisando tras cada una de las etapas. Para poder pasar a la siguiente etapa se tiene que haber conseguido todos los objetivos de la etapa anterior, es un proceso secuencial.

Tiene una buena aplicación cuando el problema es estable y cuando se trabaja con metodologías técnicas conocidas. Este modelo será apropiado para la migración de una aplicación o a una versión de mantenimiento bien definida.

Con este modelo se tiene un seguimiento de todas las fases del proyecto y del cumplimiento de todos los objetivos marcados en cada etapa tanto de costes, fechas de

entrega y lo más importante que pueden comprobar al final de cada etapa si el proyecto cumple todas las necesidades del usuario.
A su vez esto es un problema ya que si el usuario se da cuenta de que falta una tarea de la empresa en el proyecto una vez pasada esta etapa, el trabajo que hay que realizar se retrasa en fechas de entrega y el coste es mayor. Por lo tanto esto produce un fracaso en la industria ya que es reacio a las modificaciones de última hora.
Por este motivo se puede modificar el modelo en cascada pudiendo pasar de una etapa a la anterior, pero suele ser difícil ya que hay que rehacer la etapa anterior, este modelo es el ciclo de vida del salmón. Por lo tanto este es un modelo poco apropiado para proyectos con fecha de entrega corta, pero su rendimiento puede mejorar notablemente variando el modelo de la cascada pura.

Modelo Espiral

El modelo de la espiral es un modelo orientado a riesgo que divide el proyecto software en mini proyectos. Cada proyecto se encargará de resolver uno o varios riesgos hasta que estén todos controlados. Una vez que estén los riesgos más importantes controlados se finaliza igual que el ciclo de vida en cascada.
En el ciclo de vida en espiral localizan los riesgos, genera un plan para manejarlos y se establece una aproximación a la siguiente iteración. Con cada iteración se produce una aproximación al producto final.

2.13 Herramientas de Desarrollo

2.13.1 Microsoft SQL Server 2005

En la actualidad, las organizaciones deben afrontar numerosos retos relacionados con los datos; por ejemplo, la necesidad de toma de decisiones más rápidas y controladas por datos, la necesidad de aumentar la productividad y flexibilidad del personal de desarrollo y la presión para reducir los presupuestos generales relacionados con la tecnología de la información, a la vez que se exige escalar la infraestructura para que se adapte a exigencias cada vez mayores.

La siguiente versión de Microsoft® SQL Server™ se ha diseñado para ayudar a las empresas a enfrentarse a estos retos. Microsoft SQL Server 2005 es el software de última generación para el análisis y la administración de datos. Aporta un mayor grado

de seguridad, escalabilidad y disponibilidad a los datos de la empresa y a las aplicaciones de análisis, al mismo tiempo que simplifica su creación, implementación y administración.

Basado en las características de solidez de SQL Server 2000, SQL Server 2005 constituye una solución de análisis y administración de datos integrados que ayudará a las organizaciones de cualquier tamaño a:

- ✓ Crear, implementar y administrar aplicaciones empresariales que resulten más seguras, escalables y confiables.
- ✓ Maximizar la productividad de la tecnología de la información al reducir la complejidad de los procesos de creación, implementación y administración de aplicaciones de bases de datos.
- ✓ Compartir datos en varias plataformas, aplicaciones y dispositivos para facilitar la conexión de sistemas internos y externos.
- ✓ Controlar los costes sin poner en peligro el rendimiento, la disponibilidad, escalabilidad o seguridad.

2.13.2 Herramientas de la Plataforma de datos SQL Server.

SQL Server constituye una completa solución de datos de extremo a extremo que aporta a los usuarios de su organización una plataforma segura, confiable y productiva para las aplicaciones de datos de empresa e inteligencia empresarial (BI). SQL Server 2005 ofrece herramientas conocidas y de gran eficacia para los profesionales de TI, así como para aquellos que trabajan con la información. Estas herramientas reducen la complejidad que supone el proceso de crear, implementar, administrar y utilizar datos empresariales y aplicaciones analíticas en distintas plataformas que abarcan desde dispositivos móviles hasta sistemas de datos de empresas. Gracias a un extenso conjunto de características, interoperabilidad con los sistemas existentes y automatización de las tareas rutinarias, SQL Server 2005 aporta una completa solución de datos para las empresas de todos los tamaños. En la figura 1 se muestra el diseño de una plataforma de datos SQL Server 2005.

La plataforma de datos SQL Server 2005

La plataforma de datos SQL Server incluye las siguientes herramientas:

- ✓ **RelationalDatabase** (Base de datos relacional): motor de base de datos relacional seguro, confiable, escalable y de alta disponibilidad con mejoras en el rendimiento y compatibilidad con datos (XML) estructurados y sin estructurar.
- ✓ **ReplicationServices** (Servicios de duplicación): duplicación de datos para aplicaciones de procesamiento de datos distribuidos o móviles, alta disponibilidad de los sistemas, concurrencia escalable con almacenes de datos secundarios para soluciones de creación de informes empresariales e integración con sistemas heterogéneos, incluidas las bases de datos Oracle existentes.
- ✓ **NotificationServices** (Servicios de notificación): funciones avanzadas de notificación para el desarrollo e implementación de aplicaciones escalables que pueden enviar actualizaciones adecuadas y personalizadas de la información a una gran variedad de dispositivos conectados y móviles.
- ✓ **IntegrationServices** (Servicios de integración): funciones de extracción, transformación y carga para el almacenamiento de datos e integración de los datos en toda la empresa
- ✓ **AnalysisServices** (Servicios de análisis): funciones de procesamiento analítico en línea (OLAP) para el análisis rápido y sofisticado de conjuntos de datos complejos y de gran tamaño mediante el almacenamiento multidimensional.
- ✓ **ReportingServices** (Servicios de creación de informes): una completa solución para crear, administrar y entregar tanto los tradicionales informes en papel como los basados en Web interactivos.
- ✓ **Management Tools** (Herramientas de administración): SQL Server incluye herramientas de administración integradas para los procesos de ajuste y administración avanzados de bases de datos además de una estrecha integración

con herramientas como Microsoft Operations Manager (MOM) y Microsoft Systems Management Server (SMS). Los protocolos de acceso a datos estándar reducen de forma considerable el tiempo que se tarda en integrar datos en SQL Server con sistemas existentes. Además, se ha integrado en SQL Server la compatibilidad con servicios Web para garantizar la interoperabilidad con las demás aplicaciones y plataformas.

- ✓ **Herramientas de desarrollo:** SQL Server ofrece herramientas de desarrollo integradas para el motor de base de datos, extracción de datos, transformación y carga (ETL), modelos de minería, OLAP y creación de informes que están totalmente integradas en Microsoft Visual Studio® para proporcionar funciones de desarrollo de aplicaciones de extremo a extremo. Cada subsistema principal de SQL Server se suministra con su propio modelo de objetos y conjunto de API para ampliar el sistema de datos en cualquier dirección que sea exclusiva de su empresa.

La plataforma de datos SQL Server 2005 permite que organizaciones de todos los tamaños puedan disfrutar de las siguientes ventajas:

- ✓ **Aprovechamiento de los activos de datos:** además de ofrecer una base de datos segura y confiable para aplicaciones analíticas y empresariales, SQL Server 2005 permite a los clientes obtener un mayor provecho de los datos al incluir funcionalidad incrustada como la creación de informes, análisis y minería de datos.
- ✓ **Aumento de la productividad:** gracias a las completas funciones de inteligencia empresarial e integración con herramientas conocidas como Microsoft Office System, SQL Server 2005 ofrece a los que trabajan con información de su organización información empresarial esencial y adecuada adaptada a sus necesidades específicas. El objetivo es ampliar el uso de BI a todos los usuarios de una organización y, en última instancia, permitir a los usuarios de todos los niveles de la organización tomar mejores decisiones para la empresa basándose en uno de sus activos de mayor valor: los datos.
- ✓ **Reducción de la complejidad de la tecnología de la información:** SQL Server 2005 simplifica el proceso de desarrollo, implementación y administración de aplicaciones analíticas y empresariales al constituir un entorno de desarrollo

flexible para los desarrolladores, así como ofrecer herramientas integradas y automatizadas para los administradores de las bases de datos.

- ✓ **Disminución del costo total de propiedad (TCO):** el enfoque integrador y centrarse en la facilidad de uso e implementación permiten que los costos iníciales, de implementación y mantenimiento sean los más reducidos del sector de modo que se obtienen rápidos beneficios por la inversión realizada en las bases de datos.
- ✓

2.13.3 Administración de datos empresariales

En el mundo conectado en el que vivimos hoy en día, los datos y los sistemas que administran dichos datos deben mantenerse seguros pero a la vez disponibles para los usuarios. Con SQL Server 2005, todos los usuarios y profesionales de TI de su organización se beneficiarán de la disminución del tiempo de inactividad de las aplicaciones, del aumento de la escalabilidad y rendimiento, así como de controles de seguridad exhaustivos a la vez que flexibles. SQL Server 2005 incluye también numerosas funciones nuevas y mejoradas para ayudar al personal de TI a ser más productivo. Asimismo, introduce mejoras esenciales para la administración de datos empresariales en las siguientes áreas:

- ✓ Facilidad de uso
- ✓ Disponibilidad
- ✓ Escalabilidad
- ✓ Seguridad

Facilidad de uso

Con SQL Server 2005, la implementación, administración y optimización de las aplicaciones analíticas y de datos empresariales resultan más simples y sencillas. Al ser una plataforma de administración de datos empresariales, proporciona una única consola de administración que permite que los administradores de datos de cualquier área de la organización puedan controlar, administrar y ajustar todas las bases de datos y servicios relacionados de la empresa. Ofrece una infraestructura de administración extensible que se puede programar fácilmente con SMO (SQL

Management Objects), lo que permite a los usuarios personalizar y ampliar su entorno de administración y a los proveedores de software independientes (ISV) crear herramientas y funcionalidades adicionales para extender aún más las funciones ya incluidas.

Administración de datos mediante SQL Server Management Studio

SQL Server 2005 simplifica el proceso de administración al incluir una consola de administración integrada para supervisar y controlar la base de datos relacional SQL Server, así como los servicios IntegrationServices, AnalysisServices, ReportingServices, NotificationServices y SQL Mobile en un amplio número de servidores y bases de datos distribuidos. Los administradores de bases de datos pueden realizar varias tareas al mismo tiempo que incluyen: creación y ejecución de una consulta, visualización de objetos del servidor, administración de un objeto, supervisión de la actividad del sistema y visualización de la ayuda en línea. SQL Server Management Studio aloja un entorno de desarrollo para la creación, edición y administración de secuencias de comandos y procedimientos almacenados a través de Transact-SQL, Multidimensional Expressions (MDX), XMLA y SQL Server Mobile Edition. Management Studio se encuentra ya integrado con control de código fuente e incluye herramientas para programar trabajos de SQL Server Agent y administrar planes de mantenimiento a fin de automatizar las tareas diarias de mantenimiento y funcionamiento. La integración de las tareas de administración y creación en una sola herramienta junto con la capacidad de administrar todo tipo de servidores aportan una productividad mejorada a los administradores de bases de datos.

"Disponemos de miles de procedimientos almacenados y con SQL Server 2000 solía tener que usar una herramienta distinta para controlar el código y, después, abrir el analizador de consultas para editarlo. Con SQL Server 2005 todo este proceso se integra en Management Studio. Ahora puedo realizar tareas rutinarias con un 20% más de rapidez con Management Studio."

Joyce Behrendt, Director de desarrollo, planeación y análisis de estrategias corporativas de TI (InformationTechnologyCorporateStrategyPlanning and Analysis), Microsoft

Optimización y supervisión del rendimiento proactivos

Con SQL Server 2005 se exponen más de 70 nuevas medidas para el rendimiento de bases de datos internas y utilización de recursos desde la memoria, el bloqueo y la programación hasta transacciones o E/S de discos y redes. Estas vistas de administración dinámicas (Dynamic Management Views, DMV) aportan una mayor transparencia y visibilidad a la base de datos y constituyen una eficaz infraestructura para la supervisión proactiva del estado y rendimiento de las bases de datos.

2.13.4 SQL Management Objects

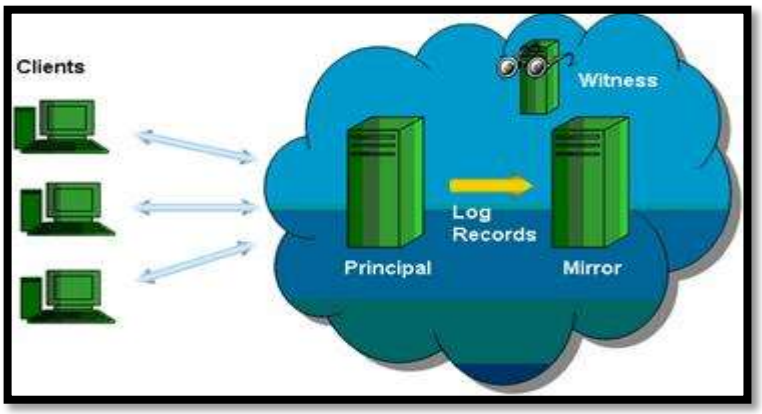
SMO (SQL Management Objects) es un nuevo conjunto de objetos de programación que cuenta con toda la funcionalidad de administración de la base de datos de SQL Server. De hecho, Management Studio se creó con SQL Management Objects. SMO se implementa como un ensamblado de Microsoft .NET Framework. Se puede utilizar SMO para automatizar las tareas administrativas habituales de SQL Server, como la recuperación mediante programación de los valores de configuración, la creación de nuevas bases de datos, la aplicación de secuencias de comandos de Transact-SQL, la creación de trabajos de SQL Server Agent y la programación de copias de seguridad. El modelo de objetos SMO es una sustitución más segura, confiable y escalable de los Objetos de administración distribuida (DMO), que se incluyó en las versiones anteriores de SQL Server.

2.13.5 Disponibilidad

Las inversiones en tecnologías de alta disponibilidad, funciones adicionales de copia de seguridad y restauración, así como mejoras en la duplicación permitirán a las empresas crear e implementar aplicaciones de alta disponibilidad. Las características innovadoras de alta disponibilidad como la creación de reflejos de bases de datos, clúster de conmutación por error, instantáneas de bases de datos y operaciones en línea mejoradas reducirán al mínimo el tiempo de inactividad y ayudarán a garantizar que los sistemas esenciales de las empresas permanecerán siempre accesibles. En esta sección repasaremos de forma detallada estas mejoras.

2.13.6 Creación de Reflejos de Bases de Datos

Esta característica permite el flujo continuo del registro de transacciones desde un servidor de origen hasta un único servidor de destino. En el caso de un error del sistema principal, las aplicaciones pueden volver a conectarse de inmediato a la base de datos del servidor secundario. La segunda instancia detecta el error del servidor principal en segundos y acepta de inmediato las conexiones a la base de datos. La creación de reflejos de bases de datos funciona en el hardware de servidores estándar y no exige almacenamiento o controladores especiales. La figura 2 muestra la configuración básica de la creación de un reflejo de una base de datos.



Configuración básica de la creación de un reflejo de una base de datos

2.13.7 Clúster de Conmutación por Error

Esta característica constituye una solución de alta disponibilidad que aprovecha Microsoft Windows® ClusteringServices para crear servidores virtuales con tolerancia a errores que ofrecen una conmutación por error rápida en el caso errores en el servidor de la base de datos. En SQL Server 2005, la compatibilidad con el clúster de conmutación por error se ha ampliado a AnalysisServices, NotificationServices y ReplicationServices de SQL Server. El número máximo de nodos del clúster ha aumentado a ocho. El clúster de conmutación por error de SQL Server constituye ahora una solución completa de servidor con tolerancia a errores.

Característica de disponibilidad	Creación de reflejos de bases de datos	Clúster de conmutación por error
Conmutación por error automática	Sí	Sí
Redirección de clientes transparente	Sí, redirección automática	Sí, nueva conexión a la misma IP
Impacto en el rendimiento general	Sin impacto o mínimo	Sin impacto
Pérdida de trabajo cero	Sí	Sí
Requiere hardware certificado	No	Sí
Proporciona datos redundantes	Sí	No

2.13.8 Opciones Instantáneas de bases de datos

SQL Server 2005 introduce la capacidad para que los administradores de datos puedan crear vistas inmediatas de sólo lectura de una base de datos. La instantánea de la base de datos ofrece una vista estable con lo que se ahorra el tiempo o almacenamiento que supone crear una copia completa de la base de datos. A medida que la base de datos principal se separa de la instantánea, esta última agrega su propia copia de las páginas que se van modificando. Por tanto, se puede utilizar la instantánea para una recuperación rápida ante un cambio realizado de forma accidental en una base de datos mediante el sencillo proceso de volver a aplicar las páginas originales de la instantánea a la base de datos principal.

Opción 1: Recuperación rápida

SQL Server 2005 mejora la disponibilidad de las bases de datos de SQL Server con la aportación de una nueva opción de recuperación más rápida. Los usuarios pueden volver a conectarse a la base de datos de recuperación después de que se haya mostrado

el registro de transacciones. Las versiones anteriores de SQL Server requieren que los usuarios esperen hasta que hayan finalizado las transacciones incompletas, incluso si los usuarios no necesitaron tener acceso a las partes afectadas de la base de datos.

Opción 2: Conexión de administrador dedicada

SQL Server 2005 introduce una conexión de administrador dedicada para tener acceso a un servidor en ejecución incluso si el servidor no responde o no está disponible. De este modo, permite ejecutar funciones de diagnóstico o instrucciones de Transact-SQL para solucionar los problemas en un servidor. La conexión la activan los miembros de la función del servidor fijo sysadmin y sólo está disponible a través de la utilidad de símbolo del sistema SQLCMD localmente o desde un equipo remoto.

Opción 3: Operaciones en línea (operaciones de índice y restauración)

La capacidad de crear, regenerar o eliminar un índice en línea constituye una característica mejorada de SQL Server 2005 que aumenta las funciones de indización de versiones anteriores de SQL Server. La opción de índices en línea permite modificaciones simultáneas (actualizaciones, eliminaciones e inserciones) en la tabla subyacente o datos de índices agrupados y en cualquier índice asociado durante la ejecución del lenguaje de definición de datos (DDL) de índices. Al admitir las operaciones de índices en línea, puede agregar índices sin interferir en el acceso a las tablas o a otros índices existentes. Además, la carga de trabajo del servidor permite operaciones de índice para aprovechar el procesamiento paralelo

SQL Server 2005 también introduce la capacidad de realizar una operación de restauración mientras está en ejecución una instancia de SQL Server. Las capacidades de restauración en línea mejoran la disponibilidad de SQL Server ya que solamente los datos que se están restaurando no están disponibles. El resto de la base de datos permanece en línea y se puede consultar. Las versiones anteriores de SQL Server requieren que la base de datos se encuentre fuera de línea para poder restaurarla.

Opción 4: Duplicación

La duplicación está diseñada para aumentar la disponibilidad de los datos al distribuirlos en varios servidores de base de datos. La disponibilidad aumenta al

permitir que las aplicaciones escalen la carga de trabajo de lectura de SQL Server en las bases de datos. SQL Server 2005 ofrece un servicio de duplicación mejorado a través de un nuevo modelo de igual a igual que proporciona una nueva topología en la que las bases de datos se pueden sincronizar mediante transacciones con una base de datos idéntica.

Opción 5: Escalabilidad

Los avances en la escalabilidad como la creación de particiones de tablas, el aislamiento de instantáneas y la compatibilidad con 64 bits permitirá crear e implementar las aplicaciones más exigentes mediante SQL Server 2005. La creación de particiones en tablas e índices de gran tamaño mejora de forma considerable el rendimiento de las consultas en bases de datos muy extensas.

Opción 6: Creación de particiones de tablas e índices

La creación de particiones de tablas e índices facilita la administración de enormes bases de datos al permitir el control de las mismas en partes más pequeñas que permiten una mejor administración. Aunque el concepto de creación de particiones en datos de las tablas, bases de datos y servidores no es nuevo en el mundo de las bases de datos, SQL Server 2005 ofrece una nueva capacidad para crear particiones de tablas en grupos de archivos de una base de datos. La creación de particiones horizontal permite la división de una tabla en agrupaciones más pequeñas basadas en un esquema de partición. La creación de particiones en tablas se ha diseñado para bases de datos muy extensas, que abarcan desde cientos de gigabytes hasta terabytes.

Opción 7: Aislamiento de instantáneas

Una vez que los datos se copian, transforman y archivan en una base de datos orientada al análisis, se deben mantener y/o regenerar de forma periódica. Los usuarios se benefician ciertamente de consultar una versión de la base de datos coherente mediante transacciones; sin embargo, la versión de los datos que están viendo deja de ser la actual. Puede llevar muchas horas crear e indizar los datos y esto podría no ajustarse a lo que el usuario realmente necesita. En este punto es cuando entra en escena el aislamiento de instantáneas. El nivel de aislamiento de instantáneas permite a los usuarios tener acceso a la última fila confirmada a través de una vista coherente

transaccional de la base de datos. Este nuevo nivel de aislamiento ofrece las siguientes ventajas:

- ✓ Aumento de la disponibilidad de los datos para las aplicaciones de sólo lectura.
- ✓ Se permiten operaciones de lectura sin bloqueos en un entorno OLTP.
- ✓ Detección automática obligatoria de conflictos para las transacciones de escritura.
- ✓ Migración simplificada de aplicaciones desde Oracle a SQL Server.

Opción 8: Monitor de duplicación

El Monitor de duplicación es una herramienta que establece un nuevo estándar en la facilidad de uso al administrar complejas operaciones de duplicación de datos con su interfaz de usuario intuitiva y riqueza de métrica de datos.

2.13.9 Ventajas que ofrece SQL Server 2005

Seguridad

SQL Server 2005 realiza mejoras importantes en el modelo de seguridad de la plataforma de base de datos, con la intención de ofrecer un control más preciso y flexible que permita una seguridad mayor de los datos. Se ha realizado una considerable inversión en una serie de características a fin de proporcionar un alto nivel de seguridad para los datos de su empresa que incluyen:

- ✓ Aplicación de directivas para las contraseñas de inicio de sesión de SQL Server en el espacio de la autenticación.
- ✓ Incorporación de mayor granularidad en términos de especificación de permisos en varios ámbitos en el espacio de la autorización.
- ✓ Capacidad de separación de propietarios y esquemas en el espacio de la administración de seguridad.

Autorización

Un nuevo modelo de seguridad en SQL Server 2005 permite a los administradores administrar permisos a un nivel granular y con un ámbito designado, permitiendo que la

administración de permisos resulte más sencilla a la vez que se garantiza el mantenimiento del principio de los privilegios mínimos. SQL Server 2005 permite especificar un contexto en el que se ejecutan las instrucciones de un módulo. Esta característica también actúa como un excelente mecanismo para la administración de permisos granulares.

Autenticación

Los clústeres de SQL Server 2005 admiten la autenticación Kerberos en un servidor virtual de SQL Server 2005. Los administradores pueden especificar directivas al estilo de Microsoft Windows en inicios de sesiones estándar para que se aplique una directiva coherente en todas las cuentas del dominio.

Cifrado nativo

SQL Server 2005 admite las funciones de cifrado dentro de la propia base de datos, totalmente integradas en una infraestructura de administración clave. De forma predeterminada, se cifran las comunicaciones cliente/servidor. Para centralizar la garantía de seguridad, se puede definir la directiva del servidor para que rechace las comunicaciones no cifradas.

SQL y Trustworthy Computing

- ✓ **Seguridad en el diseño.** El equipo de desarrollo de SQL Server 2005 realizó varias auditorías de seguridad y empleó más de dos meses en el estudio de los componentes de SQL Server 2005 y la interacción entre ellos. Ante cada posible amenaza a la seguridad, el equipo realizó un análisis de la misma para evaluar el problema y llevó a cabo tareas de diseño y pruebas adicionales para neutralizar los posibles problemas de seguridad. Como resultado de estos esfuerzos de diseño, SQL Server 2005 incluye muchas nuevas características de seguridad del servidor.
- ✓ **Seguridad predeterminada.** En la instalación, SQL Server 2005 elige el conjunto adecuado de valores de configuración para todas las opciones de instalación, con lo que garantiza que cuando se instala un nuevo sistema, se hará con un estado seguro de forma predeterminada.

- ✓ **Seguridad en la implementación.** Microsoft ha creado contenido para ayudar a las organizaciones a implementar SQL Server con las credenciales de seguridad adecuadas y a entender totalmente los pasos y permisos requeridos. Las herramientas de implementación de SQL Server 2005 aportan la información necesaria para entender las decisiones que es necesario tomar durante la implementación. Las actualizaciones de seguridad son fáciles de encontrar e instalar y, si elige la opción, las actualizaciones se instalan automáticamente. Las herramientas están disponibles para ayudarle a evaluar y administrar los riesgos de seguridad entre las organizaciones.

  La iniciativa de informática confiable Trustworthy Computing establece un marco que define los pasos necesarios para admitir una informática segura, así como las medidas que ayudan a implementar y mantener un entorno seguro. Estos pasos ayudan a proteger la confidencialidad, integridad y disponibilidad de los datos y sistemas en cada fase del ciclo de vida del software, desde el diseño hasta la entrega o el mantenimiento. Para sostener los cuatro principios de la iniciativa Trustworthy Computing, Microsoft y el equipo de SQL Server seguido estos pasos:

Productividad del desarrollador

SQL Server 2005 incluye un gran número de nuevas tecnologías que aportan un aumento considerable en la productividad del desarrollador. Desde compatibilidad con .NET Framework hasta la estrecha integración con Visual Studio®, estas características ofrecen a los desarrolladores la capacidad de crear de forma más sencilla aplicaciones de bases de datos sólidas y seguras a un bajo costo. SQL Server 2005 permite a los desarrolladores aprovechar sus habilidades existentes en una variedad de lenguajes de desarrollo a la vez que presenta un entorno de desarrollo de extremo a extremo para la base de datos. Las capacidades nativas de XML también permitirán a los desarrolladores crear nuevas clases de aplicaciones conectadas en cualquier plataforma o dispositivo.

Las mejoras para la productividad de los desarrolladores incluyen:

- ✓ Compatibilidad ampliada con lenguajes
- ✓ Herramientas de desarrollo mejoradas

- ✓ Extensibilidad
- ✓ Acceso a datos mejorado
- ✓ Servicios XML y Web
- ✓ Marco de aplicación

Compatibilidad ampliada con lenguajes

Gracias a que Common Language Runtime (CLR) se aloja en el motor de la base de datos, los desarrolladores pueden elegir entre una variedad de lenguajes conocidos para desarrollar aplicaciones de base de datos, incluidos Transact-SQL, Microsoft Visual Basic® .NET y Microsoft Visual C#® .NET. Además, el alojamiento de CLR ofrecerá a los desarrolladores una mayor flexibilidad a través del uso de funciones y tipos definidos por el usuario. CLR también brinda oportunidades para utilizar código de terceros en el desarrollo rápido de aplicaciones de bases de datos.

2.13.10 Integración de CLR/.NET Framework

Con la versión de Microsoft SQL Server 2005, los programadores de bases de datos pueden ahora aprovechar en gran medida la biblioteca de clases de Microsoft .NET Framework y los lenguajes de programación modernos para implementar la funcionalidad dentro del servidor. Con la integración de Common Language Runtime (CLR), puede codificar los procedimientos, funciones y desencadenadores almacenados en el lenguaje .NET Framework que elija. Microsoft Visual Basic .NET y el lenguaje de programación C# ofrecen ambos construcciones orientadas a objetos, control de excepciones estructurada, matrices, espacios de nombre y clases. Además, .NET Framework incluye miles de clases y métodos que disponen de capacidades integradas extensivas que puede utilizar fácilmente en el servidor. Muchas tareas que resultaban extrañas o difíciles de llevar a cabo en Transact-SQL se pueden realizar mejor con código administrado; además, se encuentran disponibles dos nuevos tipos de objetos de bases de datos: agregados y tipos definidos por el usuario. Ahora puede emplear mejor el conocimiento y las habilidades adquiridas para escribir código durante el proceso. En resumen, SQL Server 2005 le permite ampliar el servidor de base de datos para realizar más fácilmente los cálculos y las operaciones adecuadas en el servidor.

Esta integración entre SQL Server y CLR ofrece varias ventajas importantes:

- ✓ **Modelo de programación mejorado**: los lenguajes de programación que son compatibles con .NET Framework tienen, en muchos aspectos, una mayor variedad que Transact-SQL e incluyen construcciones y capacidades que anteriormente no se encontraban disponibles para los desarrolladores de SQL.
- ✓ **Seguridad mejorada**: el código administrado se ejecuta en un entorno CLR, alojado en el motor de la base de datos. Esto permite que los objetos de bases de datos de .NET Framework estén más seguros que los procedimientos almacenados extendidos disponibles en las versiones anteriores de SQL Server.
- ✓ **Tipos definidos por usuarios y agregados**: los dos nuevos objetos de base de datos que amplían las capacidades de almacenamiento y consulta de SQL Server se activan al alojar CLR.
- ✓ **Entorno de desarrollo común**: el desarrollo de base de datos se integra en el entorno de desarrollo de Microsoft Visual Studio 2005. Puede utilizar las mismas herramientas para desarrollar y depurar objetos y secuencias de comandos de base de datos que las empleadas para escribir componentes y servicios de .NET Framework de nivel intermedio o cliente.
- ✓ **Rendimiento y escalabilidad**: debido a que el código administrado se compila a código nativo antes de la ejecución, puede lograr un aumento significativo del rendimiento en algunos escenarios.

Al utilizar lenguajes como Visual Basic .NET y C#, puede aplicar las mayúsculas en la integración de CLR de modo que puede escribir código con una lógica más compleja y que sea más adecuado para las tareas de cálculo. Asimismo, Visual Basic .NET y C# ofrecen funciones orientadas a objetos como encapsulación, herencia y polimorfismo. Es posible organizar de forma sencilla código relacionado en clases y espacios de nombres, lo que significa que puede organizar y mantener más fácilmente inversiones de código cuando esté trabajando con grandes cantidades de él. La capacidad de organizar de forma lógica y física código en ensamblados y espacios de nombres constituye una enorme ventaja, ya que permite encontrar y relacionar mejor distintas partes de código en una implementación de base de datos de gran tamaño.

El código administrado resulta más eficaz que Transact-SQL en el procesamiento de números y administración de lógica de ejecución complicada. Además, permite el control de cadenas, expresiones regulares, etc. Asimismo, con la funcionalidad que está

disponible en la biblioteca de clases de .NET Framework, dispone de pleno acceso a miles de clases y rutinas creadas previamente a las que puede tener acceso fácilmente desde cualquier procedimiento, desencadenador o función definida por el usuario almacenado. Se puede tener un fácil acceso desde los procedimientos, funciones, desencadenadores y agregados almacenados a todo, desde funciones de control de cadenas mejoradas, funciones matemáticas, operaciones de fechas, acceso a los recursos del sistema, algoritmos de cifrado avanzados, acceso a archivos, procesamiento de imágenes y manipulación de datos XML.

Una de las principales ventajas del código administrado es la seguridad de los tipos. Antes de que se ejecute el código administrado, CLR realiza varias comprobaciones a través de un proceso conocido como verificación para asegurarse de que el código se ejecuta de manera segura. Por ejemplo, se comprueba el código para garantizar que no se lee la memoria que no se ha escrito.

Mejoras de Transact-SQL

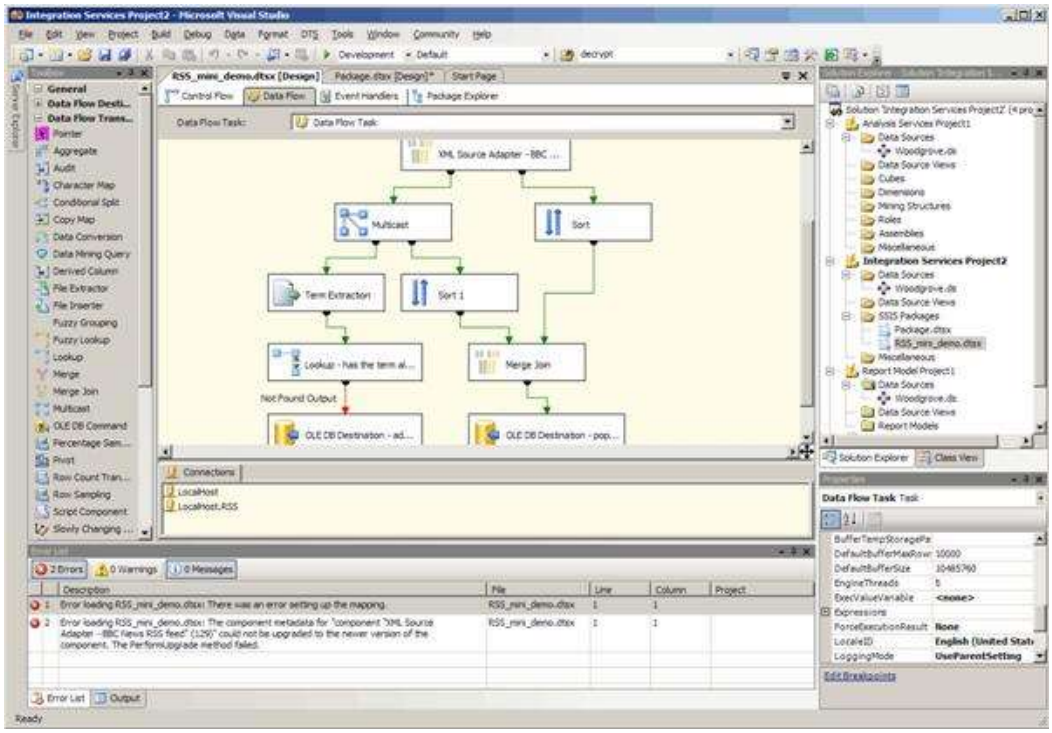
Transact-SQL ha sido durante mucho tiempo la base de toda la capacidad de programación de SQL Server. SQL Server 2005 incluye un gran número de capacidades de lenguaje nuevas para desarrollar aplicaciones de bases de datos escalables. Estas mejoras incluyen control de errores, nuevas capacidades de consulta recursivas y compatibilidad para nuevas capacidades del motor de bases de datos de SQL Server. Las mejoras de Transact-SQL en SQL Server 2005 aumentan la expresividad en la escritura de consultas, lo que le permite mejorar el rendimiento del código y ampliar las capacidades de administración. El continuo esfuerzo que se está poniendo en mejorar Transact-SQL demuestra la firme creencia en el importante papel que tiene SQL Server.

Herramientas de desarrollo mejoradas

Los desarrolladores podrán utilizar una herramienta de desarrollo para Transact-SQL, XML, Multidimensional Expressions (MDX) y XML para Analysis (XML/A). La integración con el entorno de desarrollo de Visual Studio ofrecerá un desarrollo y depuración más eficientes de las aplicaciones de línea empresarial e inteligencia empresarial (BI).

Business Intelligence Development Studio

Business Intelligence Development Studio es un entorno de desarrollo común para crear soluciones de BI basadas en Visual Studio que incluye un motor de base de datos, servicios de análisis y servicios de creación de informes. Utilice la interfaz gráfica de Business Intelligence Development Studio a fin de diseñar paquetes de servicios de integración de SQL Server (SSIS) para aplicaciones de administración de datos. Los paquetes SSIS se diseñan, desarrollan y depuran en Business Intelligence Development Studio mediante la operación de arrastre de tareas desde la caja de herramientas, la definición de sus propiedades y la conexión de tareas con limitaciones de precedencia.



Interfaz de Business Intelligence Development Studio en Visual Studio

2.13.11 Integración de Visual Studio

SQL Server 2005 y Visual Studio 2008 en combinación proporcionan niveles más profundos que nunca de integración entre la base de datos y el entorno de desarrollo de aplicaciones. Los desarrolladores cuentan ahora con la capacidad de crear procedimientos, funciones, tipos definidos por el usuario y agregados defir 37 ior el usuario directamente en el entorno de desarrollo de Visual Studio. Pueden implementar estos nuevos objetos de bases de datos directamente desde Visual Studio sin tener que

cambiar herramientas. Visual Studio 2008 admite todos los tipos nuevos de datos de SQL Server, como XML nativo directamente. También puede agregar sus objetos de bases de datos de CLR al mismo sistema de control de origen que utilizó para todos los proyectos de Visual Studio, aportando así un mayor nivel de integración y seguridad al proceso de desarrollo.

Depuración entre niveles y entre lenguajes

La combinación de SQL Server 2005 y Visual Studio 2008 ofrece una integración aún más profunda en la zona de depuración de la aplicación. Esta combinación permite depurar sin problemas tanto código de CLR como de Transact-SQL con la misma interfaz de depuración de Visual Studio y es posible depurar de CLR a Transact-SQL y viceversa, independientemente de la ubicación del código, tanto si está en el equipo del desarrollador o almacenado en la base de datos de SQL Server.

Tipos y agregados definidos por el usuario

Los tipos definidos por el usuario en SQL Server 2005 no constituyen un mecanismo de extensibilidad relacional. Son un modo de ampliar el sistema de tipos escalar de la base de datos. El sistema de tipos escalar incluye los tipos de columna que se suministran en SQL Server (tipos como **int**, **nvarchar**, **uniqueidentifier**, etc.). Con los tipos definidos por el usuario, se puede establecer un tipo propio que se utilice para las definiciones de columnas, por ejemplo. Cree un tipo definido por el usuario si su tipo es realmente un valor atómico que se puede modelar como una columna.

Utilice tipos definidos por el usuario si necesita definir su propio tipo escalar. Entre los escenarios de ejemplos para estos tipos se incluyen tipos de datos de fecha y hora en distintos calendarios y tipos de datos de moneda. Con los tipos definidos por el usuario, puede crear un único objeto que exponga todos los comportamientos que están disponibles en el tipo y encapsular u ocultar los datos subyacentes que ha almacenado el tipo. Cada persona que necesite tener acceso a los datos tiene que utilizar la interfaz mediante programación de los tipos definidos por el usuario. Si puede aprovechar la funcionalidad existente en .NET Framework (como la funcionalidad de

internacionalización o de calendario), sería otra buena razón para considerar la implementación del tipo como tipo definido por el usuario.

Existen una serie de casos en los que puede necesitar realizar agregaciones en los datos. Esto incluye realizar cálculos estadísticos, como avg, stddev, etc. Si la función de agregación deseada no se admite directamente como una función de agregación integrada, hay tres formas de realizar una agregación personalizada en SQL Server 2005:

- ✓ Escribir la agregación como un agregado definido por el usuario.
- ✓ Escribir el agregado con el procedimiento almacenado de CLR.
- ✓ Utilizar un cursor del servidor

SQL Management Objects (SMO)

SQL Management Objects (SMO) es el modelo de objetos de administración para SQL Server 2005. SMO representa mejoras importantes de diseño y arquitectura para el modelo de objetos de administración de SQL Server. Se trata de un modelo de objetos fácil de utilizar pero de una gran riqueza que está basado en el código administrado de .NET Framework. SMO es la herramienta principal para el desarrollo de aplicaciones de administración de base de datos con .NET Framework. Cada cuadro de diálogo de SQL Server Management Studio utiliza SMO y cada acción administrativa que pueda realizar en SQL Server Management Studio también la puede realizar mediante SMO.

El nuevo modelo de objetos SMO y las API de Microsoft Windows Management Instrumentation (WMI) sustituyen a SQL-DMO. Siempre que es posible, SMO incorpora objetos similares a SQL-DMO para que resulte fácil de usar. Puede seguir utilizando SQL Server 2005 con SQL-DMO, pero SQL-DMO no se actualizará para administrar características específicas de SQL Server 2005.

Analysis Management Objects (AMO)

Analysis Management Objects (AMO) permite a las aplicaciones cliente tener acceso a la variedad de comandos y capacidades administrativas disponibles en Analysis Services con una biblioteca de objetos que puede ofrecer capacidades de validación en el nivel de los objetos en lugar de tener que generar manualmente secuencias de

comandos DDL para los comandos de Analysis Services y los contenidos a menudo extensos del elemento ObjectDefinition de Analysis Services Scripting Language (ASSL). Las aplicaciones que utilizan AMO pueden conectarse o trabajar directamente con objetos en la instancia de Analysis Services o crear dichos objetos sin una conexión directa y continuar con los metadatos para su implementación posterior. AMO también "ajusta" los comandos y elementos de ASSL.

Acceso a datos y servicios Web mejorados

En SQL Server 2005, puede desarrollar servicios Web XML en el nivel de la base de datos, convirtiendo SQL Server en un detector HTTP. De este modo, se proporciona un nuevo tipo de capacidad de acceso a datos para aplicaciones centralizada en los servicios Web. En SQL Server 2005 puede utilizar HTTP para tener acceso directamente a SQL Server, sin utilizar un detector de nivel medio como Servicios de Internet Information Server (IIS) de Microsoft. SQL Server muestra una interfaz de servicio Web para permitir la ejecución de instrucciones SQL y la invocación de funciones y procedimientos. Los resultados de las consultas se devuelven en un formato XML y puede aprovechar la infraestructura de los servicios Web de Visual Studio.

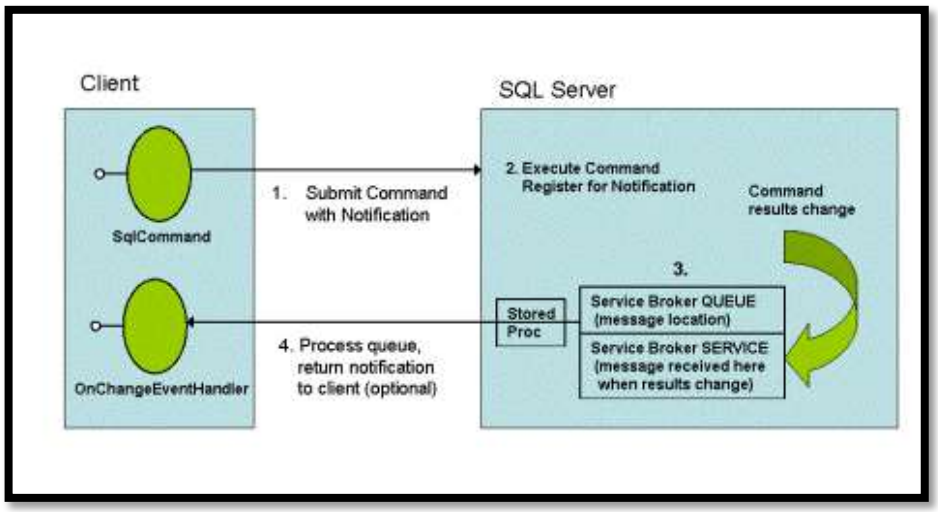
ADO.NET 2.0/ADOMD.NET

Hay muchas más novedades en la siguiente versión de ADO.NET. Desde nueva compatibilidad para notificaciones de cambio de consultas hasta conjuntos MARS (Multiple Active Result Sets), ADO.NET evoluciona en el acceso a los conjuntos de datos y la manipulación para conseguir una mayor escalabilidad y flexibilidad.

2.13.12 Notificación de consultas

SQL Server 2005 introduce un soporte de notificaciones para las consultas de SQL Server. Puede utilizar este soporte para enviar un comando a SQL Server y solicitar que se genere una notificación si se vuelve a ejecutar el mismo comando y se producen resultados distintos a los obtenidos inicialmente. Esto se consigue mediante un objeto de dependencia que detecta cuando se modifican los datos subyacentes. Los comandos que se envían al servidor a través de cualquier API de cliente como ADO.NET, OLE DB,

ODBC (Open Database Connectivity), Microsoft ActiveX® Data Objects (ADO) o SOAP pueden incluir una etiqueta que requiera una notificación. En cada instrucción que se ejecuta como parte de la solicitud, el servidor crea una suscripción de notificación que se activa una vez por cada instrucción que se incluye en la solicitud. Las notificaciones se entregan a través de la cola de SQL Service Broker que las aplicaciones pueden sondear y emplean los servicios de activación o las instrucciones de bloqueo que se devuelven cada vez que hay disponibles notificaciones. Las notificaciones de consulta son útiles para permitir el almacenamiento en caché de los resultados de las aplicaciones como los sitios Web destinados a bases de datos.



Notificación de consultas

MARS

MARS permite tener más de una solicitud pendiente por conexión, en especial tener más de un conjunto de resultados predeterminado abierto por conexión. Los conjuntos de resultados predeterminados son de sólo lectura y sólo reenvío. En los conjuntos de resultados predeterminados, los conductores de clientes recuperan de forma transparente los datos en grandes fragmentos (del tamaño de búfer en secuencias de datos tabular) para que las peticiones de aplicación se realicen sin un viaje de ida y vuelta al servidor (como en el caso de los cursores de servidor). La aplicación puede utilizar un modelo de programación de filas individuales sin poner en peligro el rendimiento. La característica de varios conjuntos de resultados activos elimina la restricción actual en la que un

conjunto de resultados predeterminado abierto bloquea el controlador para que no envíe peticiones al servidor hasta que se consuma todo el conjunto de resultados.

2.13.13 Compatibilidad con XML, XQuery y otros servicios.

Avances como el tipo de datos nativos XML y XQuery ayudan a las organizaciones a conectar sin problemas sistemas externos e internos. SQL Server 2005 admitirá tanto los datos XML como los relacionales de forma nativa, por lo que las empresas pueden almacenar, administrar y analizar los datos con el formato que mejor se adapte a sus necesidades. La compatibilidad con los estándares abiertos existentes y nuevos como el protocolo de transferencia de hipertexto (HTTP), XML, SOAP (Simple Object Access Protocol), XQuery y lenguajes de definición de esquemas XML (XSD) también facilitarán la comunicación con los sistemas empresariales ampliados.

Tipo de datos XML

XML puede modelar datos complejos; no se limita a los tipos escalares que admite SQL Server. De este modo, un tipo de datos integrado basado en cadenas como char o varchar no es suficiente para utilizar de forma total y efectiva la eficacia y las numerosas ventajas de XML. Por ejemplo, si XML se almacena como cadena, puede insertar o seleccionar un documento entero, o incluso recuperar bytes contiguos, pero no se puede consultar el contenido del propio documento. Al incluir el tipo de datos XML, SQL Server 2005 permite consultar partes de un documento XML, comprobar que el documento cumple con el esquema XML e incluso modificar el contenido del documento XML en su sitio. También integra datos relacionales tradicionales con datos de documentos XML semiestructurados o sin estructurar en modos que no son posibles con SQL Server 2000. En SQL Server 2005, los datos XML se almacenan como grandes objetos binarios (BLOB) en una representación interna que permite volver a realizar un eficiente análisis y realizar alguna compresión.

Una colección de esquemas XML se puede asociar con una columna de tipo XML. De este modo, se cuenta con la validación para las restricciones, inserciones y actualizaciones y la escritura de valores dentro de los datos XML almacenados, así como con optimizaciones para el almacenamiento y el procesamiento de consultas. SQL

Server 2005 también ofrece varias instancias de DDL para administrar los esquemas en el servidor.

XQuery

El lenguaje de consultas XML, o XQuery, es un lenguaje inteligente y seguro que se optimiza para consultar todos los tipos de datos XML. Con XQuery se pueden ejecutar consultas con variables y columnas de tipos de datos XML a través de los últimos métodos relacionados. Al igual que con gran cantidad de estándares XML, World Wide Web Consortium (W3C) supervisa el desarrollo de XQuery. XQuery evolucionó de un lenguaje de consultas denominado Quilt, que a su vez estaba basado en una variedad de otros lenguajes de consultas como XML Path Language (XPath) versión 1.0, XQL y SQL. También contiene XPath 2.0 como subconjunto. Por tanto, si tiene experiencia en el uso de XPath 1.0, puede aprovechar sus habilidades y no tener que aprender un lenguaje de consulta totalmente nuevo. Hay, sin embargo, mejoras significativas que van más allá de XPath 1.0, como la escritura, funciones especiales y capacidad para una mejor iteración, clasificación de resultados y construcción.

SQL Server 2005 se distribuye con capacidades profundas de XQuery que permiten la manipulación de objetos XML en el nivel de datos. Admite un subconjunto escrito en forma de estadística de XQuery 1.0 Working Draft del 15 de noviembre de 2003.

Compatibilidad con servicios Web

En SQL Server 2005, puede desarrollar servicios Web XML en el nivel de la base de datos, convirtiendo SQL Server en un detector HTTP. De este modo, se proporciona un nuevo tipo de capacidad de acceso a datos para aplicaciones centralizadas en los servicios Web. En SQL Server 2005 puede utilizar HTTP para tener acceso directamente a SQL Server, sin utilizar un detector de nivel medio como Servicios de Internet Information Server (IIS) de Microsoft. SQL Server muestra una interfaz de servicio Web que permite la ejecución de instrucciones SQL y la invocación de funciones y procedimientos. Los resultados de las consultas se devuelven en un formato XML y se puede sacar provecho de la infraestructura de los servicios Web de Visual Studio.

XML para Analysis Services (XML/A)

XML para Analysis Services (XML/A) es el protocolo nativo basado en estándares para la comunicación con el servidor de Analysis Services. Se habilitan nuevos tipos de aplicaciones fáciles de desarrollar: aplicaciones que integran la analítica con operaciones en tiempo real. Con XML/A como protocolo nativo, los clientes de Analysis Services se pueden configurar para que tengan una superficie cero y cada servidor sea un servicio Web automáticamente. Hay disponible una capa Win32 de superficie ligera para la compatibilidad con versiones anteriores de herramientas que funcionan con Analysis Services 2000 en OLE DB para OLAP, ADOMD y ADOMD.NET. Muchos usuarios continuarán utilizando el modelo de objetos de ADOMD.NET para crear aplicaciones personalizadas en Analysis Services.

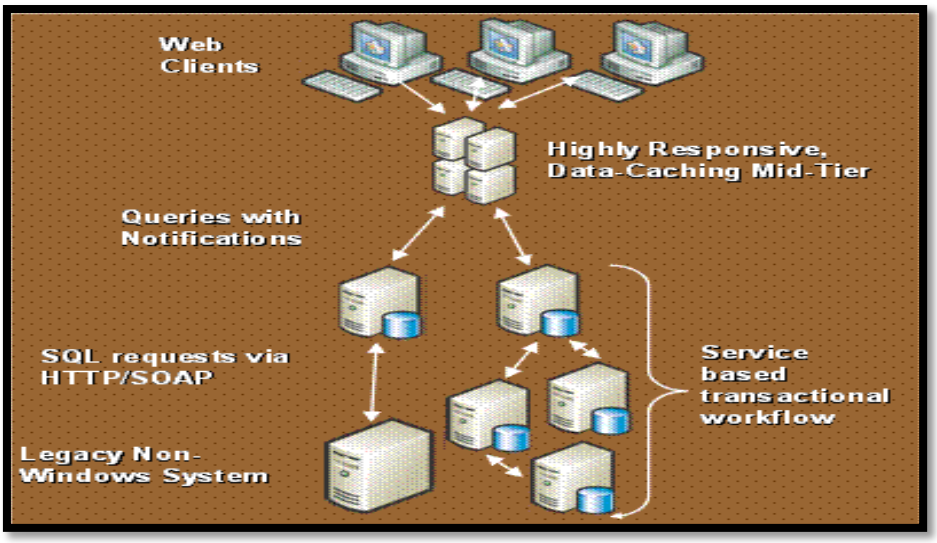
Marco de aplicación

SQL Server 2005 introduce un nuevo marco de aplicación de SQL Server que incluye: Service Broker, Notification Services, SQL Server Mobile y SQL Server Express. Service Broker es un marco de aplicación distribuida que presenta un sistema de mensajería asincrónica segura en la base de datos y a nivel de bases de datos.

Service Broker

En los últimos 10 años, con la proliferación de las aplicaciones de comercio electrónico ha surgido la necesidad de un aumento en la administración de flujos de trabajo en las aplicaciones de las bases de datos. Cuando un cliente en línea coloca el pedido de un libro, este pedido necesita realizar transacciones en los sistemas de inventario, envíos y tarjetas de crédito, así como enviar una confirmación del pedido con otra aplicación Web. La espera para que tenga lugar en orden cada uno de estos procesos no se escala bien. SQL Server 2005 presenta una nueva arquitectura escalable para crear un enrutamiento de mensajes asincrónico. En la figura 5 se muestra la arquitectura de Service Broker.

Arquitectura de Service Broker



La tecnología de Service Broker permite que procesos internos y externos envíen y reciban flujos de mensajes asincrónicos seguros mediante las extensiones del lenguaje de manipulación de datos normal de Transact-SQL. Los mensajes se envían a una cola en la misma base de datos que el remitente, a otra base de datos en la misma instancia de SQL Server, o bien a otra instancia de SQL Server en el mismo servidor o en otro servidor remoto.

"Service Broker en SQL Server 2005 ha aumentado en un 60 por ciento la velocidad de desarrollo en comparación con la escritura de trabajos en SQL."-- Ketan Patel, Desarrollador jefe de aplicaciones, planeación y análisis de estrategias corporativas de TI (Information Technology Corporate Strategy Planning and Analysis), Microsoft

Notification Services

Notification Services de Microsoft SQL Server es una plataforma para el desarrollo e implementación de aplicaciones que generan y envían notificaciones a los usuarios. Las notificaciones son mensajes oportunos personalizados que se pueden enviar a una amplia variedad de dispositivos.

Las notificaciones reflejan las preferencias del suscriptor. El suscriptor introduce una suscripción para expresar un interés por información. Por ejemplo, "notificarme cuando

el precio de las cotizaciones de Adventure Works llegue a $70.00" o "notificarme cuando el documento de estrategia que está escribiendo mi equipo se actualice".

Las notificaciones se pueden generar y enviar al usuario en cuanto se produce un evento de desencadenamiento, o bien se pueden generar y enviar según una programación predeterminada que especifique el usuario. La suscripción del usuario indica cuándo se debe generar y enviar la notificación.

Las notificaciones se pueden enviar a una amplia gama de dispositivos. Por ejemplo, se puede enviar a un teléfono móvil de un usuario, a un asistente digital personal (PDA), a Microsoft Windows Messenger o una cuenta de correo electrónico. Como el usuario siempre va acompañado de estos dispositivos, las notificaciones constituyen una forma ideal de enviar información de alta prioridad.

SQL Server Mobile Edition

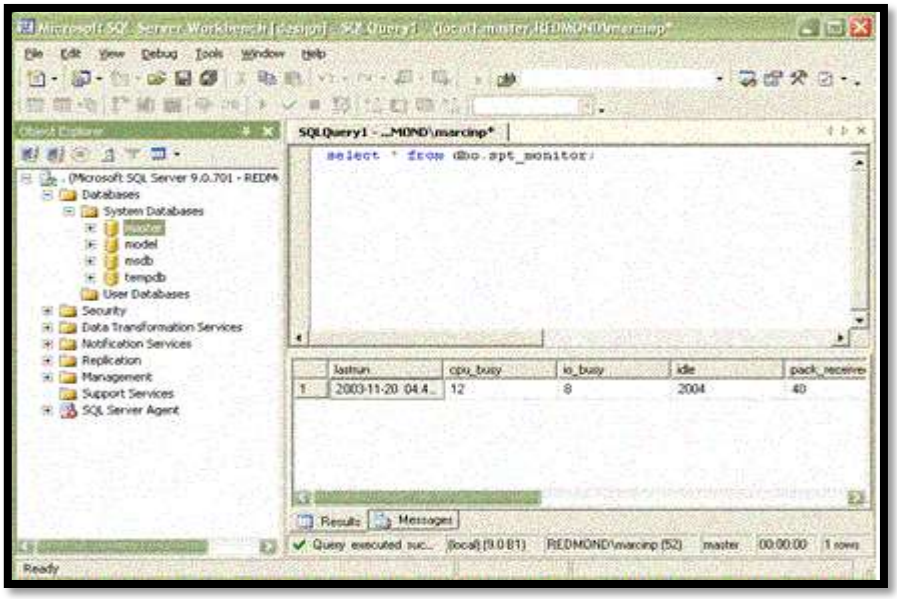
SQL Server 2000 incluido en SQL Server 2000 Windows CE Edition, que es ahora SQL Server Mobile Edition versión 3.0. Existe una serie de nuevas características clave en SQL Server Mobile Edition que está relacionada con los desarrolladores:

- ✓ Puede crear una base de datos SQL Server Mobile Edition en el escritorio o en el dispositivo, directamente desde SQL Server Management Studio. También puede manipular el esquema de la base de datos de SQL Server Mobile Edition directamente desde Management Studio, independientemente de si la base de datos reside en el dispositivo móvil o en el escritorio. Se puede utilizar SQL Server Management Studio para ejecutar consultas destinadas a la base de datos de SQL Server Mobile Edition en el dispositivo o en el escritorio. También puede aprovechar las nuevas características de SQL Server Mobile Edition que incluyen un plan de presentación de XML procesado en un formato de interfaz gráfica de usuario como SQL Server nativo y la capacidad de utilizar sugerencias de consultas para sustituir el optimizador de consultas de SQL Server Mobile Edition. Por primera vez, puede controlar el plan de optimización en un dispositivo.

- ✓ Ahora podrá codificar objetos de Integration Services de SQL Server (SSIS) para intercambiar datos.
- ✓ El nuevo conjunto SqlCeResult se deriva del conjunto SQLResult que están en SQL Server 2005. De este modo, permite que SQL Server Mobile Edition tenga un auténtico cursor con capacidad de arrastre y que se puede actualizar. También permite la vinculación a objetos de datos que están en los dispositivos.
- ✓ Puede codificar una aplicación para sincronizar los datos mientras se deja abierta la aplicación principal y es posible tener dos aplicaciones distintas con acceso a la misma base de datos en el dispositivo al mismo tiempo.
- ✓ Puede obtener notificaciones que podrá codificar en barras de estado que darán el estado de una sincronización. Anteriormente, no había forma de saber lo lejos que llegaba el estado de la sincronización a fin de notificar a los usuarios que un dispositivo no había dejado de responder.
- ✓ Puede mantener el tamaño reducido de la base de datos a través de una directiva de reclamación de páginas mucho más agresiva.
- ✓ Puede compartir código de consultas de parámetros con la sintaxis de SQL Server.

SQL Server Express

Ahora más que nunca los desarrolladores están aprovechando las bases de datos relacionales para ofrecer una enriquecedora experiencia al usuario final. La protección y administración de la información dentro de estas aplicaciones es crítica. Microsoft SQL Server Express ayuda a los desarrolladores a crear aplicaciones sólidas y confiables al proporcionar una base de datos libre, fácil de usar y segura. Demasiado a menudo los sistemas de bases de datos resultan muy complicados para crear aplicaciones sencillas. Microsoft Visual Studio 2005 y SQL Server Express reducen esta complejidad al proporcionar un entorno sencillo pero de gran eficacia para crear aplicaciones de datos. Los desarrolladores pueden diseñar esquemas, agregar datos y consultar bases de datos locales, todo dentro del entorno de Visual Studio 2005. Si los desarrolladores necesitaran más características avanzadas de base de datos, SQL Server Express se puede actualizar sin problemas a versiones más sofisticadas de SQL Server. En la figura 6 se muestra la interfaz del editor de consultas en SQL Server Express Manager.



El editor de consultas en SQL Server Express Manager (XM)

Una nueva herramienta de interfaz gráfica de usuario denominada SQL Server Express Manager (XM) se ofrece gratuitamente como una descarga Web independiente. XM permite una fácil administración de bases de datos y capacidades de análisis de consultas, tendrá un tamaño pequeño de descarga y se redistribuirá gratuitamente. XM admite conexiones a SQL Server Express y a otras ediciones de SQL Server 2005, SQL Server 2000 y MSDE 2000. Un cuadro de diálogo de conexión simplificada guía al usuario a través de la selección de la instancia y los métodos de autenticación que se emplearán. Tanto las conexiones locales como remotas son posibles con XM. El explorador de objetos enumerará y mostrará los objetos comunes utilizados, como la instancia, tablas, procesos almacenados, etc.,de una manera jerárquica y ayudará al usuario a visualizar el acceso a la base de datos.

Todas las funcionalidades de administración de bases de datos están disponibles al abrir el menú contextual con el botón secundario del *mouse* desde el explorador de objetos. Algunas de las opciones de administración de bases de datos que se van a mostrar incluyen la creación y modificación de bases de datos, tablas, inicios de sesión y usuarios. Muchas de estas operaciones de bases de datos frecuentes se encuentran disponibles como asistentes de tareas que guían al usuario a través del proceso mientras que otras están disponibles como documentos de ventanas con fichas. Por ejemplo, XM

proporcionará un documento New/Edit Database para crear nuevas bases de datos y editar las existentes.

Hay muchos usuarios de bases de datos que prefieren administrar sus servidores con Transact-SQL, ya que este enfoque ofrece un control más granulado que la interfaz gráfica de usuario. El editor de consultas de XM permitirá a los usuarios desarrollar y ejecutar instrucciones y secuencias de comandos de Transact-SQL. Este editor tendrá una gran variedad de características como la codificación de colores de las palabras clave y un panel de resultados que devolverá los resultados en una cuadrícula de datos. Los mensajes de error, si los hay, también se mostrarán en el panel de resultados.

Minería de datos

La minería de datos de Microsoft SQL Server 2005 es la tecnología de inteligencia empresarial que le permite crear complejos modelos analíticos e integrarlos con las operaciones empresariales. Analysis Services de Microsoft SQL Server 2005 establece una nueva base para la minería de datos. Al crear una plataforma fácil de usar, ampliable, accesible y flexible, las funciones de minería de datos de Analysis Services de SQL Server 2005 introduce este concepto en organizaciones que anteriormente nunca se habrían planteado una solución de este tipo.

A través de una arquitectura de clases empresarial; una profunda integración con la familia SQL Server de herramientas de inteligencia empresarial y un variado conjunto de herramientas, API y algoritmos, SQL Server permite la creación de una nueva raza de aplicaciones inteligentes que mejoran la productividad, aumentan los beneficios y reducen los costos al aportar soluciones de datos personalizados a una gran variedad de problemas de las empresas.

Reporting Services

Reporting Services amplía la plataforma de IB de Microsoft para satisfacer las necesidades del trabajador de información que necesita tener acceso a los datos empresariales. Reporting Services es un entorno de creación de informes empresariales basados en el servidor y administrados a través de los servicios Web. Los informes se pueden entregar en una gran variedad de formatos, con una gama de opciones de

interactividad e impresión. Los análisis complejos pueden llegar a una amplia audiencia a través de la distribución de informes como origen de datos para bajar la tendencia de la inteligencia empresarial.

El componente integrado de SQL Server 2005, Reporting Services, ofrece:

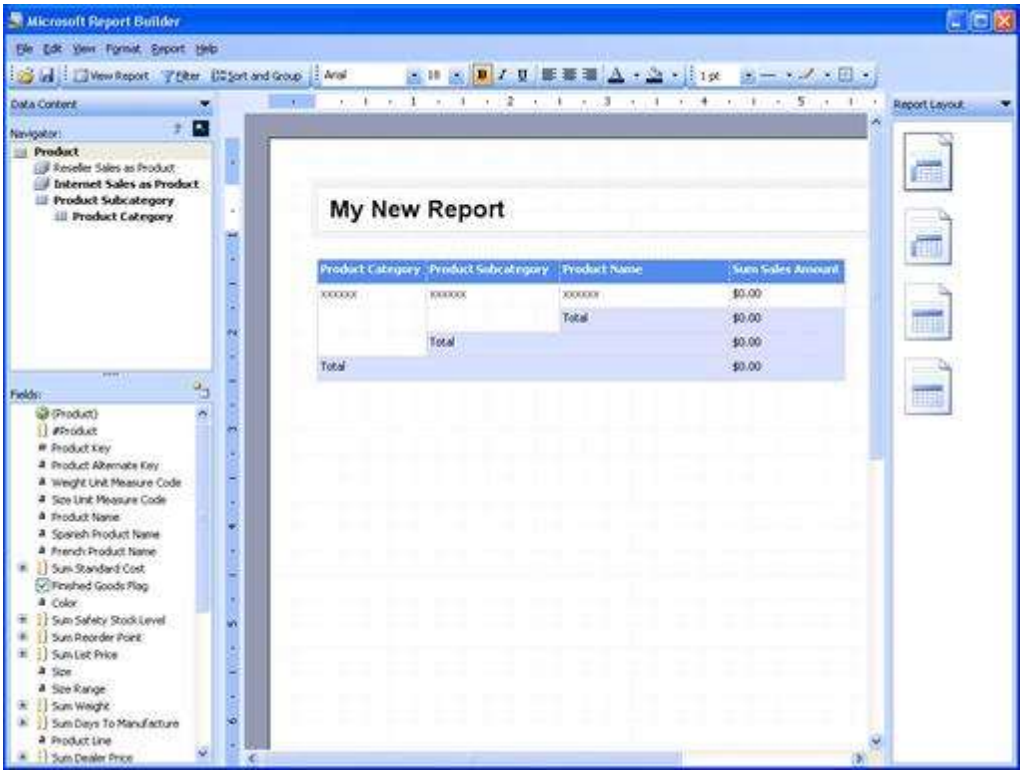
- ✓ Un motor de alto rendimiento para procesar y aplicar formato a informes.
- ✓ Un conjunto completo de herramientas para crear, administrar y ver informes.
- ✓ Una arquitectura extensible e interfaces abiertas para incrustar informes o integrar soluciones de creación de informes en diversos entornos de la TI.

Informes relacionales y OLAP

Los informes creados con datos relacionales resultan útiles pero la capacidad de agregar funciones de analítica adicionales aumentan considerablemente la eficacia de la creación de informes. Reporting Services permite crear fácilmente informes combinados o independientes. SQL Server 2005 admite los datos tanto relacionales como OLAP y proporciona un editor de consultas para incluir los editores de consultas de SQL y MDX.

Report Builder

Report Builder, un nuevo componente de Reporting Services de SQL Server 2005, permite a los usuarios de empresas crear sus propios informes mediante un modelo para el usuario de sus datos. Report Builder utiliza la plataforma de Reporting Services para acercar la creación de informes ad hoc a todos los usuarios finales. Los usuarios crean y editan informes con la aplicación cliente de Report Builder. La interfaz de usuario de Report Builder se ha creado según los paradigmas conocidos de Microsoft Office, como Excel y PowerPoint. En la figura 7 se muestra un informe de ejemplo de Report Builder.



Diseño de informes con Report Builder

Report Builder es una aplicación ClickOnce que se implementa a través del explorador. Los usuarios empiezan por seleccionar plantillas de diseño de informes que contienen secciones de datos predefinidas como tablas, matrices y gráficos. Deben arrastrar y soltar elementos de informe desde el modelo hasta la superficie de diseño y establecer los límites para filtrar los datos del informe. El modelo contiene toda la información necesaria para que Report Builder genere automáticamente la consulta origen y recupere los datos solicitados. Report Builder también permite a los usuarios:

- ✓ Agregar texto y aplicar formato a informes.
- ✓ Crear nuevos campos y cálculos definidos con el modelo.
- ✓ Obtener una vista previa, imprimir y publicar informes.
- ✓ Exportar datos de informes a formatos como Microsoft Excel.

Integración con Microsoft Office System

Los informes que se sirven con Report Server en Reporting Services se pueden ejecutar en el contexto de las aplicaciones de Microsoft SharePoint® Portal Server y Microsoft Office System como Microsoft Word y Microsoft Excel. Se pueden utilizar

características de SharePoint para suscribirse a informes, crear nuevas versiones de los mismos y distribuirlos. Asimismo, se pueden abrir informes en Word o Excel para ver las versiones HTML de los mismos.

2.13.14 Actualización a SQL Server 2005

A continuación, se muestran algunas sugerencias para actualizar a SQL Server 2005.

- ✓ Actualizar a SQL Server 2005 desde SQL Server 7.0 o SQL Server 2000.
- ✓ Ejecutar el asesor de actualizaciones antes de actualizar para determinar si se esperan cambios de algún producto que afecten a aplicaciones existentes.
- ✓ En la instalación se pueden actualizar a Database Engine, Analysis Services y Reporting Services.
- ✓ Integration Services de SQL Server, el sustituto de DTS, se instala totalmente con DTS. Puede ejecutar paquetes DTS mediante los componentes en tiempo de dimensión de DTS.
- ✓ Notification Services de SQL Server 2005 se instala de lado a lado con Notification Services 2.0. Debe migrar instancias de Notification Services a SQL Server 2005 al actualizar el motor de base de datos.
- ✓ Utilizar la herramienta Surface Area Configuration después de actualizar para habilitar o deshabilitar los servicios, protocolos de red y características de SQL Server 2005.

2.13.15 Conclusión de contenido

SQL Server 2005 ofrece la tecnología y las funciones con las que pueden contar las organizaciones. Con avances significativos en áreas clave de la administración de datos empresariales, la productividad de los desarrolladores y la inteligencia empresarial, las ventajas de SQL Server 2005 son considerables.

En este artículo se muestra que SQL Server 2005 puede beneficiar a su organización de las siguientes maneras:

- ✓ **Aprovechamiento de los activos de datos:** además de ofrecer una base de datos segura y confiable para aplicaciones analíticas y empresariales, SQL Server

2005 permite a los clientes obtener un mayor provecho de los datos al incluir funcionalidad incrustada como creación de informes, análisis y minería de datos.

- ✓ **Aumento de la productividad:** gracias a las completas funciones de inteligencia empresarial e integración con herramientas conocidas como Office, SQL Server 2005 ofrece a los que trabajan con datos en su organización información empresarial esencial y adecuada adaptada a sus necesidades específicas. Nuestro objetivo es ampliar BI a todos los usuarios de una organización y en última instancia permitir a los usuarios de todos los niveles de la organización tomar mejores decisiones para la empresa basándose en uno de sus activos de mayor valor: sus datos.
- ✓ **Reducción de la complejidad de la tecnología de la información:** SQL Server 2005 simplifica el proceso de desarrollo, implementación y administración de aplicaciones analíticas y empresariales al proporcionar un entorno de desarrollo flexible para los desarrolladores así como herramientas integradas y automatizadas para los administradores de bases de datos.
- ✓ **Disminución del costo total de propiedad (TCO):** nuestro enfoque integrador y centrarse en la facilidad de uso ofrece los costos iniciales, de implementación y mantenimiento más reducidos del sector a fin de conseguir rápidos beneficios en la inversión de las bases de datos.

2.14 Aplicación Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

A partir de la versión 2005 Microsoft ofrece gratuitamente las *Express Editions*. Estas son varias ediciones básicas separadas por lenguajes de programación o plataforma

enfocadas para novatos y entusiastas. Estas ediciones son iguales al entorno de desarrollo comercial pero sin características avanzadas. Las ediciones que hay son:

- ✓ Visual Basic Express Edition
- ✓ Visual C# Express Edition
- ✓ Visual C++ Express Edition
- ✓ Visual J# Express Edition (Desapareció en Visual Studio 2008)
- ✓ Visual Web Developer Express Edition (para programar en ASP.NET)
- ✓ Visual F# (Apareció en Visual Studio 2010, es parecido al J#)*

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de MS SQL Server llamada SQL Server Express Edition cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente utiliza un procesador y un Gb de Ram, y no cuenta con el Agente de SQL Server.

En el pasado se incluyeron los siguientes productos:

- ✓ Visual InterDev
- ✓ Visual J++
- ✓ Visual FoxPro
- ✓ Visual SourceSafe

Visual Studio 2005

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. Microsoft eliminó *.NET*, pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de *tipos genéricos*, similares en muchos aspectos a las plantillas de C++. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era

posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Se incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

Visual Studio 2005 también añade soporte de 64-bit. Aunque el entorno de desarrollo sigue siendo una aplicación de 32 bits Visual C++ 2005 soporta compilación para x86-64 (AMD64 e Intel 64) e IA-64 (Itanium). El SDK incluye compiladores de 64 bits así como versiones de 64 bits de las librerías.

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System. Éstas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los 4 principales roles de la programación: Architects, Software Developers, Testers, y Database Professionals. La funcionalidad combinada de las 4 ediciones Team System se ofrecía como la edición Team Suite.

Tools for the Microsoft Office System está diseñada para extender la funcionalidad a Microsoft Office.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft[2] se incluye una edición independiente para cada lenguaje: Visual Basic, Visual C++, Visual C#, Visual J# para programación .NET en Windows, y Visual Web Developer para la creación de sitios web ASP.NET. Las ediciones express carecen de algunas herramientas avanzadas de programación así cómo de opciones de extensibilidad.

Se lanzó el service Pack 1 para Visual Studio 2005 el 14 de Diciembre de 2006.

La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo es la 9.0.

Visual Studio 2008

Visual Studio 2008 fue publicado (RTM) el 17 de Noviembre de 2007 en inglés, mientras que la versión en castellano no fue publicada hasta el 2 de Febrero de 2008.[3]

El nuevo framework (.Net 3.5) está diseñado para aprovechar las ventajas que ofrece el nuevo sistema operativo "Windows Vista" a través de sus subsistemas "Windows Communication Foundation" (WCF) y "Windows Presentation Foundation" (WPF).El primero tiene como objetivo la construcción de aplicaciones orientadas a servicios mientras que el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento.

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades.

- ✓ La mejora en las capacidades de Pruebas Unitarias permiten ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos. Se incluye además un nuevo soporte para diagnosticar y optimizar el sistema a través de las herramientas de pruebas de Visual Studio. Con ellas se podrán ejecutar perfiles durante las pruebas para que ejecuten cargas, prueben procedimientos contra un sistema y registren su comportamiento; y utilizar herramientas integradas para depurar y optimizar.
- ✓ Con **Visual Studio Tools for Office (VSTO)** integrado con **Visual Studio 2008** es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario (UI) de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project. Una completa compatibilidad para implementación con ClickOnce garantiza el entorno ideal para una fácil instalación y mantenimiento de las soluciones Office.
- ✓ **Visual Studio 2008** permite incorporar características del nuevo Windows Presentation Foundation sin dificultad tanto en los formularios de Windows existentes como en los nuevos. Ahora es posible actualizar el estilo visual de las aplicaciones al de Windows Vista debido a las mejoras en Microsoft Foundation Class Library (MFC) y Visual C++. Visual Studio 2008 permite mejorar la

interoperabilidad entre código nativo y código manejado por .NET. Esta integración más profunda simplificará el trabajo de diseño y codificación.

- ✓ **LINQ (Language Integrated Query)** es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos, a través de extensiones para C++ y Visual Basic así como para Microsoft .NET Framework. Permite filtrar, enumerar, y crear proyecciones de muchos tipos y colecciones de datos utilizando todos la misma sintaxis, prescindiendo del uso de lenguajes especializados como SQL o XPath.
- ✓ **Visual Studio 2008** ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).
- ✓ .**NET 3.5** incluye biblioteca **ASP.NET AJAX** para desarrollar aplicaciones web más eficientes, interactivas y altamente personalizadas que funcionen para todos los navegadores más populares y utilicen las últimas tecnologías y herramientas Web, incluyendo Silverlight y Popfly.

Visual Studio 2010

Visual Studio 2010 es la versión más reciente de esta herramienta, acompañada por .NET Framework 4.0. La fecha prevista para el lanzamiento de la versión final ha sido el 12 de abril de 2010.[5]

Hasta ahora, uno de los mayores logros de la versión 2010 de Visual Studio ha sido el de incluir las herramientas para desarrollo de aplicaciones para Windows7, tales como herramientas para el desarrollo de las características de Windows7 (System.Windows.Shell) y la Ribbon Preview para WPF.

Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo. Además de esto, aparece una edición que compila las características de todas las ediciones comunes de Visual Studio: **Professional, Team Studio, Test**, conocida como **Visual Studio Ultimate.**

2.14.1 Windows Form

Los formularios son el elemento básico del interfaz de usuario (IU) en aplicaciones creadas para Microsoft Windows®. Proporcionan un marco de trabajo que puede utilizarse por toda la aplicación para crear un aspecto coherente. Los formularios de aplicaciones basadas en Windows se utilizan para presentar información al usuario y aceptar la introducción de datos por parte del mismo.

Los formularios exponen propiedades que definen su apariencia, métodos que definen su comportamiento, y eventos que definen su interacción con el usuario. Estableciendo las propiedades y escribiendo código para responder a sus eventos, el formulario se personaliza para satisfacer los requerimientos de las aplicaciones. El formulario es un control derivado de la clase **Form**, que a su vez deriva de la clase **Control**. El marco de trabajo también permite heredar de formularios existentes para añadir más funcionalidades o modificar el comportamiento existente. Cuando se añade un formulario a un proyecto, se puede escoger si hereda de la clase **Form** proporcionada por el .NET Framework, o de un formulario creado con anterioridad.

Windows Forms frente a Web Forms

Característica	Windows Forms	Web Forms
Implantación	Puede ejecutarse sin alterar el Registro	No se requiere descarga
Gráficos	Incluye GDI+	Los gráficos interactivos o dinámicos requieren ida y vuelta al servidor para su actualización
Respuesta	Velocidad de respuesta más rápida posible para aplicaciones interactivas	Pueden aprovechar el HTML Dinámico del navegador y crear ricos IU
Plataforma	Requiere el .NET Framework ejecutándose en la máquina cliente	Sólo requiere un navegador
Modelo de programación	Basado en un modo de intercambio de mensajes Win32 en el lado cliente	Los componentes de aplicaciones se invocan mediante HTTP
Seguridad	Seguridad basada en código y basada en roles	Seguridad basada en roles

TABLA II.3: Características de los Windows Forms y Web Forms

Introducción

Cuando se diseñan aplicaciones que necesitan de un interfaz de usuario, existen dos opciones: Windows Forms y Web Forms. Ambos disponen de soporte completo en tiempo de diseño por parte del entorno de desarrollo y pueden generar un rico interfaz de usuario y funcionalidad avanzada de aplicaciones para solucionar problemas de negocio. Cuando se dispone de varias opciones, es importante saber qué opción utilizar en cada momento.

Web Forms

Web Forms de ASP.NET se utilizan para crear aplicaciones en las que el interfaz de usuario principal es un navegador. Incluyen aplicaciones pensadas para estar disponibles públicamente en la World Wide Web, como las aplicaciones de correo electrónico.

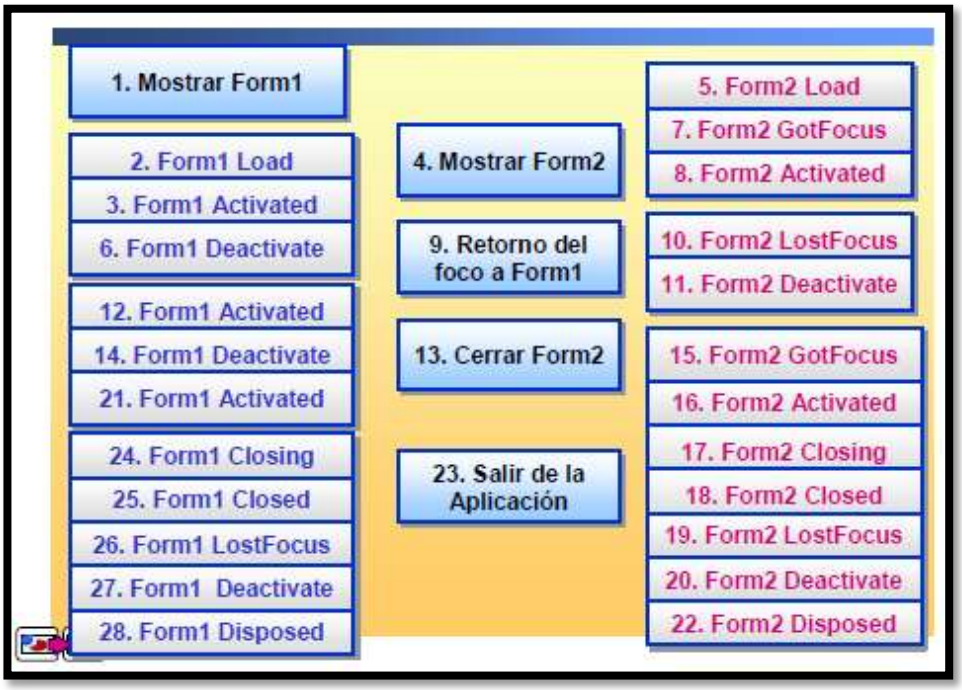
La siguiente tabla ofrece una comparativa de distintos criterios de comparación y cómo las tecnologías Windows Forms y Web Forms satisfacen estos criterios.

Característica / criterio	**Windows Forms**	**Web Forms**
Implantación	Las aplicaciones pueden descargarse, instalarse y ejecutarse directamente en el ordenador del usuario sin ninguna alteración del registro.	No tiene una implantación específica en el cliente; éste únicamente requiere un navegador. El servidor debe ejecutar Microsoft .NET Framework. Las actualizaciones de la aplicación se realizan actualizando el código en el servidor.

Gráficos	Windows Forms incluye GDI+ (Graphic Design Interface (GDI+), que permite utilizar gráficos sofisticados para juegos y otros entornos gráficos de gran riqueza.	Los gráficos interactivos o dinámicos requieren comunicación de ida y vuelta al servidor para actualizarse cuando se utilizan en Web Forms. Puede utilizarse GDI+ en el servidor para crear gráficos personalizados.
Respuesta	Windows Forms puede ejecutarse completamente en el equipo cliente; puede proporcionar mayor velocidad de respuesta para aplicaciones que requieren un alto grado de interactividad.	Si se sabe que los usuarios dispondrán de Microsoft Internet Explorer 5 o posterior, una aplicación Web Forms puede aprovechar las capacidades de HTML (DHTML) dinámico del navegador para crear una IU rica y con buena capacidad de respuesta. Si los usuarios tienen otros navegadores, la mayor parte del procesamiento (incluyendo tareas relacionadas con la IU, como la validación) requiere una comunicación de ida y vuelta al servidor Web, lo cual puede afectar a la capacidad de respuesta.
Plataforma	Windows Forms requiere que el .NET Framework se esté ejecutando en el equipo cliente.	Web Forms únicamente requiere un navegador. Los navegadores con capacidad DHTML pueden aprovechar

		las características adicionales, peroWeb Forms puede diseñarse para funcionar con todos los navegadores. El servidor Web debe estar ejecutando el .NET Framework.

2.14.2 Ciclo de vida de un formulario



2.14.3 Programación orientada a objetos

En la programación orientada a objetos, utilizamos la abstracción y la encapsulación para crear clases bien diseñadas.

Definición

Una *clase* es una plantilla o una estructura preliminar de un objeto. Esta estructura preliminar define atributos para almacenar datos y define operaciones para manipular esos datos. Una clase también define un conjunto de restricciones para permitir o denegar el acceso a sus atributos y operaciones.

Uso de la abstracción

Para crear una clase bien diseñada, utilizaremos la *abstracción.* Al implementar la abstracción, definiremos un concepto utilizando un mínimo conjunto de funcionalidades pensadas cuidadosamente que proporcione el comportamiento fundamental de la clase de un modo fácil de utilizar. Por desgracia, no es fácil crear buenas abstracciones de software. Normalmente, encontrar buenas abstracciones requiere un profundo conocimiento del problema que ha de resolver la clase y su contexto, una gran claridad de ideas y mucha experiencia.

Clases

Ejemplo de abstracción

El formulario Visual Basic .NET con el que hemos estado trabajando es un buen ejemplo de abstracción. Las propiedades esenciales de un formulario, como el título y color de fondo, se han abstraído en la clase **Form**. Algunas operaciones esenciales que se han abstraído son abrir, cerrar y minimizar.

Uso de la encapsulación

La abstracción se garantiza mediante la *encapsulación.* La encapsulación es el empaquetamiento de atributos y funcionalidades para crear un objeto que esencialmente es una "caja negra" (cuya estructura interna permanece privada). Empaquetamos los detalles de la abstracción y proporcionamos acceso sólo a los elementos que necesitan estar accesibles. Otros objetos pueden acceder a los servicios de un objeto encapsulado únicamente mediante mensajes que pasan a través de una interfaz claramente definida.

Ejemplo de encapsulación

Un ejemplo de encapsulación es un terminal de autoservicio (*automatic teller machine*, ATM). La interfaz de la ATM es simple para el cliente ATM, y el funcionamiento interno está oculto. Del mismo modo, una clase **BankAccount** encapsularía los métodos, campos y propiedades que describen una cuenta bancaria. Sin la encapsulación, deberíamos declarar procedimientos y variables distintos para almacenar y gestionar información de la cuenta bancaria, y sería difícil trabajar con más de una cuenta bancaria a la vez. La encapsulación permite a los usuarios utilizar los datos y

procedimientos de la clase **BankAccount** como una unidad, sin conocer el código concreto encapsulado en la clase.

Objetos

Podemos utilizar el Examinador de objetos para examinar los elementos de programación de un componente (espacios de nombres, clases, módulos, estructuras, etc.) y los miembros de esos elementos (propiedades, métodos, eventos, variables, etc.). Los componentes que examinemos pueden ser proyectos de nuestra solución, componentes referenciados en esos proyectos o componentes externos.

Uso del Examinador de objetos

Para abrir el Examinador de objetos, pulse F2 o, en el menú **Ver**, seleccione **Examinador de objetos**.

El Examinador de objetos está formado por tres paneles:

- El *panel Objetos* muestra todos los objetos contenedores del ámbito de búsqueda en una vista en árbol. Cuando expandimos un elemento haciendo doble clic en él o haciendo clic en el signo más (+) junto a su nombre, aparecen los elementos definidos en él.
- El *panel Miembros* muestra los miembros de un elemento cuando seleccionamos el elemento en el panel Objetos.
- El *panel Descripción* muestra información detallada sobre el elemento o miembro actualmente seleccionado.

Un icono específico representa cada elemento de programación en el Examinador de objetos. La siguiente tabla muestra algunos de los iconos que se utilizan en la programación:

Icono	Descripción	Icono	Descripción
{}	Espacio de nombres		Módulo
	Clase		Estructura

2.14.4 Estructura de Programación en Visual Studio.

Tipos de Datos

El sistema de tipos comunes define cómo se declaran, utilizan y gestionan los tipos en el Common Language Runtime. Cada tipo de datos utilizado en Visual Basic .NET corresponde directamente a un tipo definido en el sistema de tipos comunes.

Ventajas del sistema de tipos comunes

El sistema de tipos comunes tiene una gran importancia en la creación de aplicaciones para la plataforma Microsoft .NET. Hace posible que un desarrollador pueda crear un proyecto en Visual Basic .NET e integrarlo con un componente creado por otro desarrollador en Microsoft Visual C#™ y una función escrita por un tercer desarrollador en otro lenguaje compatible con .NET. Todas estas piezas pueden integrarse en una única solución. Los compiladores y herramientas de Microsoft Visual Studio® .NET y el Common Language Runtime dependen del sistema de tipos comunes para proporcionar:

- Integración entre lenguajes.
- Código con seguridad de tipos, lo que significa que únicamente se accede a los tipos de forma permisible y bien definida.
- Las herramientas que gestionan y permiten una ejecución del código de alto rendimiento.

Tipo valor vs. Tipo referencia

El sistema de tipos comunes soporta dos categorías generales de tipos: *tipos valor* y *tipos referencia*.

Una variable de tipo valor contiene directamente sus datos. Cada variable de tipo valor tiene su propia copia de datos, de modo que las operaciones en una variable de tipo valor no pueden afectar a otra variable.

Una variable de tipo referencia contiene una referencia o puntero al valor de un objeto. Dos variables de tipo referencia pueden referirse al mismo objeto, de modo que las

operaciones en una variable de tipo referencia pueden afectar al objeto referenciado por otra variable de tipo referencia.

Tipo de datos

Tipo Visual Basic .NET	Tamaño de almacenamiento	Rango de valores
Boolean	2 bytes	Verdadero o Falso
Date	8 bytes	0:00:00 del 1 de enero de 0001 a 11:59:59 PM del 31 de diciembre de 9999
Decimal	16 bytes	Hasta 29 dígitos significativos, con valores de hasta $7{,}9228 \times 10^{29}$ (con signo)
Double	8 bytes	-4,94065645841246544E-324 a +1,79769313486231570E+308 (con signo)
Integer	4 bytes	-2.147.483.648 a +2.147.483.647 (con signo)
Single	4 bytes	-3,4028235E+38 a 1,401298E-45 (con signo)
String	Varía	0 a 2.000 millones aproximadamente de caracteres Unicode

Tipos de datos y características

Escoger un tipo de datos

Escoger tipo de datos...	para gestionar...	Tipo CTS	Ejemplo
Boolean	Condiciones de Verdadero o Falso	Valor	Verdadero
Short, Integer, Long, Byte	Enteros	Valor	23 (Entero)
Single, Double, Decimal	Números con enteros y partes de fracciones	Valor	9456,72 (Decimal)
Date	Valores fecha y hora	Valor	02/12/2003 12:30:42 A.M.
String	Caracteres imprimibles y visualizables en pantalla	Referencia	"Casa"
Object	Un puntero al valor de un objeto	Referencia	myClass myPerson

Procedimientos: tipos, estructura e invocaciones.

Los procedimientos son las sentencias de código ejecutable de un programa. Las instrucciones de un procedimiento están delimitadas por una instrucción de declaración y una instrucción **End**.

Tipos de procedimientos

Existen tres tipos de procedimientos en Microsoft Visual Basic® .NET: procedimientos **Sub**, procedimientos **Function** y procedimientos **Property**.

- Los procedimientos **Sub** realizan acciones pero no devuelven un valor al procedimiento que origina la llamada. Los controladores de eventos son procedimientos **Sub** que se ejecutan en respuesta a un evento.
- Los procedimientos **Function** pueden devolver un valor al procedimiento que origina la llamada. La instrucción **MessageBox.Show** es un ejemplo de función.
- Los procedimientos **Property** devuelven y asignan valores de propiedades de clases, estructuras o módulos.

Uso de procedimientos

Un procedimiento puede ser invocado, o llamado, desde otro procedimiento. Cuando un procedimiento llama a otro procedimiento, se transfiere el control al segundo procedimiento. Cuando finaliza la ejecución del código del segundo procedimiento, éste devuelve el control al procedimiento que lo invocó.

Debido a esta funcionalidad, los procedimientos resultan útiles para realizar tareas repetidas o compartidas. En lugar de escribir el mismo código más de una vez, podemos escribir un procedimiento e invocarlo desde varios puntos de nuestra aplicación o desde otras aplicaciones.

Accesibilidad del procedimiento

Utilizamos un modificador de acceso para definir la accesibilidad de los procedimientos que escribimos (es decir, el permiso para que otro código invoque al procedimiento). Si no especificamos un modificador de acceso, los procedimientos son declarados *public* de forma predeterminada.

La siguiente tabla muestra las opciones de accesibilidad para declarar un procedimiento dentro de un módulo:

Modificador de acceso	**Descripción**
Public	Ninguna restricción de acceso
Friend	Accesible desde el programa que contiene la declaración y desde cualquier otro lugar del mismo ensamblado
Private	Accesible únicamente en el módulo que contiene la declaración

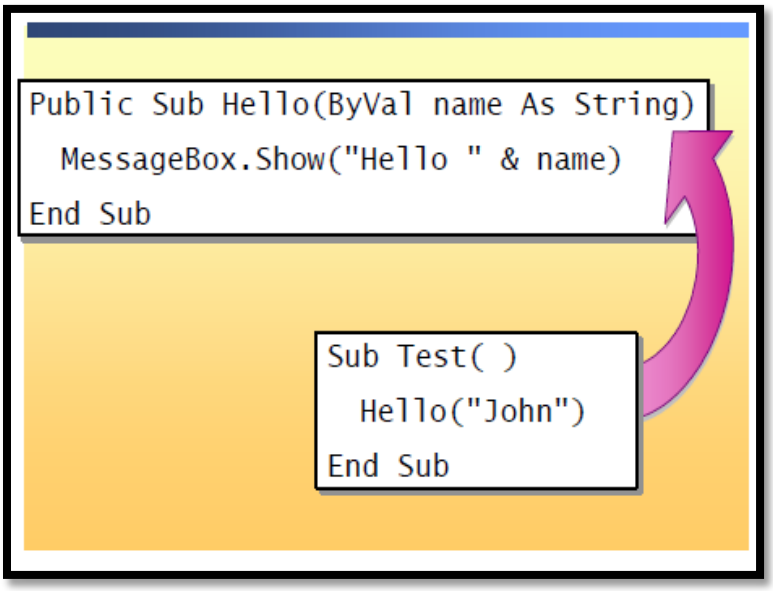
Accesibilidad de un procedimiento

Para utilizar un procedimiento **Sub**, lo invocamos desde otro procedimiento.

Flujo de código

Cada vez que se invoca un procedimiento **Sub**, se ejecutan sus instrucciones, empezando por la primera instrucción ejecutable después de la instrucción **Sub** y finalizando con la primera instrucción **End Sub**, **Exit Sub** o **Return** encontrada. Después de que el procedimiento **Sub** ejecute nuestro código, devuelve la ejecución del programa a la línea de código que sigue a la línea que invocó el procedimiento **Sub**.

Invocar un procedimiento Sub



La sintaxis para invocar un procedimiento **Sub** es la siguiente:

[Call] *Subname [(Argumentlist)]*

- Debemos invocar el procedimiento **Sub** en una línea por sí mis*mo* en nuestro código (no puede invocarlo utilizando su nombre dentro de una expresión).
- La instrucción de llamada debe proporcionar valores para todos los argumentos que no son opcionales.
- Opcionalmente, podemos utilizar la instrucción **Call** para invocar un procedimiento **Sub**. El uso de la instrucción **Call** puede mejorar la legibilidad de nuestro programa.

Los procedimientos **Sub** no devuelven un valor a la instrucción de llamada. Sin embargo, un procedimiento **Sub** pasar información de retorno al código de llamada modificando argumentos pasados por referencia.

Ejemplo de una invocación simple

El siguiente código muestra un procedimiento de evento que invoca un procedimiento **Sub** denominado **SetData**:

Sub DataButton_Click(...)

SetData( )

End Sub

2.14.5 Funciones: Estructura e Invocaciones

Subrutinas y procedimientos

Ejemplo de uso de la instrucción Call

También puede utilizar el siguiente código para realiza la misma tarea:

Sub DataButton_Click(...)

Call SetData( )

End Sub

Ejemplo de invocación simple incorrecta

La siguiente invocación contiene un error:

Sub DataButton_Click(...)

MessageBox.Show(SetData( ))

End Sub

' Causes an error, because the Show method expects a String

' data type, not un procedure

Ejemplo de invocación con argumentos

Observe la siguiente definición para el procedimiento **Sub SetData**:

Public Sub SetData(ByVal cars As Integer, ByVal trucks As _ Integer, ByVal vans As Integer)

' Code for SetData Procedure

```
End Sub
```

La sintaxis para invocar este procedimiento incluye el nombre del procedimiento y la lista de argumentos en paréntesis, como se muestra en el siguiente código:

```
Sub DataButton_Click(...)

SetData(10, 20, 30)

End Sub
```

Ejemplo incorrecto de una llamada con argumentos

La siguiente llamada al procedimiento **SetData** definido en el ejemplo a continuación contiene un error.

```
Sub DataButton_Click( )

SetData(10, 20)

End Sub

' Causes an error, because there is no valor for the third

' parametre of the SetData procedure. The calling statement

' must provide values for all arguments that are not optional.
```

Invocar un procedimiento Function

Un procedimiento **Function** se diferencia de un procedimiento **Sub** en que el primero puede devolver un valor al procedimiento de llamada.

Invocar una función

Invocamos un procedimiento **Function** incluyendo su nombre y sus argumentos en el lado derecho de una instrucción de asignación o en una expresión. Piense en la siguiente función, que convierte una temperatura en Fahrenheit a una temperatura en Celsius.

```
Function FtoC(ByVal temperature As Single) As Single

' Convert Fahrenheit to Celsius

FtoC = (temperature - 32.0) * (5 / 9)

End Function
```

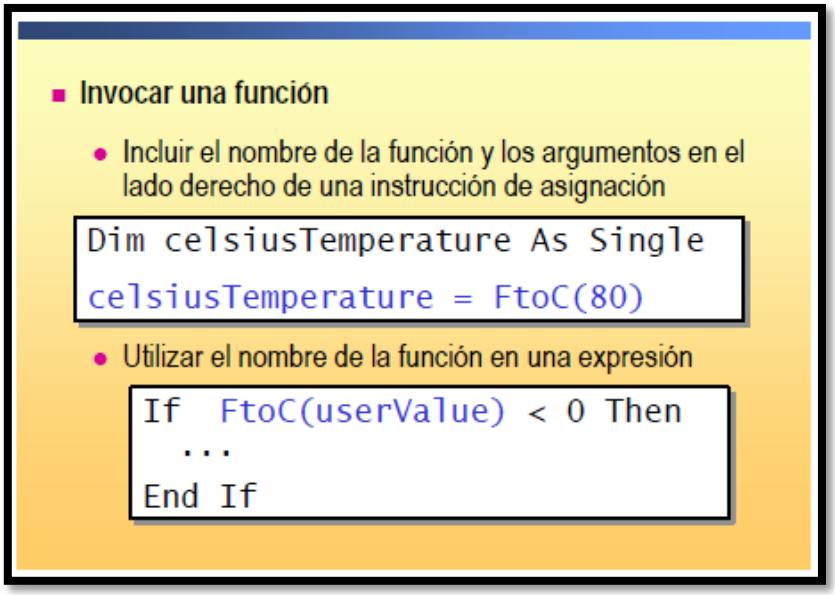
Las siguientes llamadas de ejemplo muestran cómo podríamos invocar esta función:

```
Dim celsiusTemperature As Single

celsiusTemperature = FtoC(80)

' Call the procedure by including its name and arguments on

' the right side of an assignment statement. In this call,

' the value 80 is passed to the FtoC function, and the

' value returned is assigned to celsiusTemperature.

If FtoC(userValue) < 0 Then . . .

' Call the procedure by using it in an expression. In this

' call, the FtoC function is used as part of an expression.

End If
```

Flujo de código

Cada vez que se invoca la función se ejecutan sus instrucciones, empezando por la primera instrucción ejecutable tras la instrucción **Function** y finalizando con la primera instrucción **End.**

Function, **Exit Function** o **Return** encontrada.



Ejemplo de invocar una Función

2.14.6 Arreglos o Matrices: estructura e invocaciones

Podemos pasar matrices como argumentos a un procedimiento igual que otros argumentos. Visual Basic .NET también proporciona la palabra clave **ParamArray** para declarar una matriz de parámetros en la definición de parámetros de un procedimiento.

Pasar matrices

Podemos pasar matrices unidimensionales o multidimensionales a procedimientos del mismo modo que pasamos otros argumentos.

El siguiente ejemplo muestra cómo pasar una matriz unidimensional a un procedimiento:

```
Sub PassArray(ByVal testScores As Integer( ))

...

End Sub
```

```
Dim scores( ) As Integer = {80, 92, 73}

PassArray(scores)
```

El siguiente ejemplo muestra cómo pasar una matriz bidimensional a un procedimiento:

```
Sub Pass2DArray(ByVal rectangle As Integer(,))

...

End Sub

Dim rectangle(,) As Integer = {{12, 1}, {0, 12}}

Pass2DArray(rectangle)
```

Uso de ParamArray

Normalmente, no podemos invocar un procedimiento con más argumentos de los especificados en su declaración. Cuando necesitamos un número indefinido de argumentos, podemos declarar una matriz de parámetros, que permite que un procedimiento acepte una matriz de valores para un argumento. No es necesario conocer el número de elementos de la matriz de parámetros cuando definimos el procedimiento. El tamaño de la matriz está determinado de forma individual por cada invocación al procedimiento.

Utilizamos la palabra clave **ParamArray** para denotar una matriz de parámetros. Esta palabra clave indica que el argumento de un procedimiento es una matriz opcional de elementos de un tipo especificado. Se aplican las siguientes reglas:

- Un procedimiento sólo puede tener una matriz de parámetros, y debe ser el último argumento de la definición del procedimiento.
- La matriz de parámetros debe pasarse por valor. Es una buena práctica de programación incluir explícitamente la palabra clave **ByVal** en la definición del procedimiento.

- El código dentro del procedimiento debe tratar la matriz de parámetros como una matriz unidimensional, siendo cada elemento de la misma el mismo tipo de datos que el tipo de datos **ParamArray**.
- La matriz de parámetros es automáticamente opcional. Su valor predeterminado es una matriz unidimensional vacía del tipo de elemento del parámetro de la matriz.
- Todos los argumentos que preceden a la matriz de parámetros deben ser obligatorios. La matriz de parámetros debe ser el único argumento opcional.

Invocar un procedimiento con un argumento de matriz de parámetros.

- Cuando invocamos un procedimiento con un argumento de matriz de parámetros, podemos pasar alguna de las opciones siguientes para la matriz de parámetros:
- Nada. Es decir, podemos omitir el argumento **ParamArray** En este caso, se pasa al procedimiento una matriz vacía. También podemos pasar la palabra clave **Nothing**, produciendo el mismo efecto.
- Una lista de un número indefinido de argumentos, separados por comas. El tipo de datos de cada argumento debe ser implícitamente convertible al tipo de elemento **ParamArray**.
- Una matriz con el mismo tipo de elemento que la matriz de parámetros.

Ejemplo de declaración ParamArray

El siguiente código muestra cómo podemos definir un procedimiento con una matriz de parámetros:

```
Sub StudentScores(ByVal name As String, ByVal ParamArray _

scores( ) As String)

' Statements for Sub procedure

End Sub
```

Ejemplos de invocaciones a un procedimiento con una matriz de parámetros

Los siguientes ejemplos muestran invocaciones posibles a **StudentScores**.

```
StudentScores("Anne", "10", "26", "32", "15", "22", "16")

StudentScores("Mary", "High", "Low", "Average", "High")

Dim JohnScores( ) As String = {"35", "Absent", "21", "30"}

StudentScores("John", JohnScores).
```

CAPITULO III

3.1 MARCO METODOLOGICO

3.1.1 Tipo de Investigación

En esta investigación se realizará un estudio experimental y correlacional, ya que se ha de manipular la variable independiente (causa) y se verán los resultados en la variable dependiente. Además es un diseño tecnológico ya que utiliza muchas herramientas conjuntas en el campo de la Ingeniería del Software conjuntamente con la Ingeniería de Sistemas.

Esto específicamente a lo que se ha desarrollado en la investigación y a lo que vamos a demostrar luego con el planteamiento de la hipótesis.

Experimental, porque esta investigación va más allá de simples conceptos y metodologías a utilizar, está dirigida a ver qué problemas existen, es decir este estudio se centra en explicar por qué ocurre la demora en la atención a los usuarios momento de hacer una compra.

Correlacionar, ya que el estudio se centra en manipular la variable independiente que es mejorar las ventas en Ferrecentro Oñate S.A. de la ciudad de Babahoyo para afectar la variable dependiente es decir mejorar la atención a los usuarios.

Se utilizarán en esta investigación los siguientes métodos:

Método Científico: Estudia la realidad descomponiéndola en sus elementos constitutivos mediante el análisis y la síntesis, obteniendo una visión global de la misma. Porque se basará en investigaciones que ya han sido comprobadas anteriormente por varios autores y además se tendrá la ayuda de los profesionales que tienen experiencia en este tema.

Método Deductivo: Porque analizaremos primero los contenidos teóricos para luego desglosarlos y pasarlos a la práctica para finalmente llegar a conclusiones y recomendaciones adecuadas.

Método Inductivo: A partir del análisis de un caso o de casos particulares y observaciones de la realidad se extraen conclusiones de carácter general. Comienza con

la recolección de datos, se categoriza las variables observadas, se prueban las hipótesis, se puede realizar generalizaciones para elaborar una teoría.

Método Bibliográfico: Realizaremos una Investigación Bibliográfica sobre lo que con lleva el Sistema a desarrollarse como son: Bases de Datos, Lenguajes de Programación, Plataformas de Soporte y manejo de Sistemas Operativos. Se agrupará todas las referencias de libros utilizados en la presente tesis organizando de alguna manera ya sea alfabética o sistemáticamente.

3.1.2 PLANTEAMIENTO DE HIPÓTESIS, VARIABLES

FORMULACIÓN DE HIPÓTESIS

El control en los procesos de ventas en Ferrecentro Oñate S.A .de la ciudad de Babahoyo mejorará si se dispone de un sistema informático que facilite el control de ingresos y egresos de mercadería.

OPERACIONALIZACION DE LAS VARIABLES

De acuerdo a la formulación de hipótesis se tienen identificadas claramente 2 variables:

Variable Independiente:

Sistema Informático

Variable Dependiente:

Mejorar las Ventas

3.1.2.1 Operacionalización Conceptual

VARIABLES	DEFINICION
Variable Independiente Sistema Informático .	Un sistema informático se basa en un conjunto de hardware, software y de un soporte humano con el fin de controlar, procesar y optimizar datos obteniendo resultados confiables, rápidos, y exactos.

Variable Dependiente Mejorar las Ventas	Actividad que permite dar a conocer resultados en el proceso de ventas y asi poder satisfacer las necesidades tanto de la empresa como del usuario.

Operacionalización Conceptual de las variables

3.1.2.2 Operacionalización Metodológica

HIPOTESIS	VARIABLES	INDICADORES	TECNICAS	FUENTES DE VERIFICACION
Desarrollo del sistema informático para mejorar las ventas de Ferrecentro Oñate S.A. de la ciudad de Babahoyo utilizando visual.net 2008 y sqlserver2005 se mejorará la atención a los Usuarios.	**VARIABLE INDEPENDIENTE.** Desarrollo del sistema informático para mejorar las ventas de Ferrecentro Oñate S.A. de la ciudad de Babahoyo utilizando visual.net 2008 y sqlserver2005	❖ Validación ❖ Optimización ❖ Exactitud	Entrevista Observación	Usuarios Empleados Director

	VARIABLE DEPENDIENTE Mejorar las Ventas	❖ Registros ❖ Control ❖ Excelencia	Revisión Documental Encuesta Entrevistas	Sistema Manual Sistema Informático Empleados Usuarios Directivos

Operacionalización Metodológica de las variables

3.2 POBLACIÓN Y MUESTRA

En el presente trabajo de investigación, la población la componen los Usuario, Empleados y Directivos concretamente aquellos que están involucrados en este proceso de control de Ventas de Ferrecentro Oñate.

Así tenemos:

Babahoyo	
Usuarios	900
Empleados	10
Administrativos	3
TOTAL	913

Población Global

La "población accesible" es la que reúne las mismas características que la anterior, pero con menor número de individuos, y por tanto susceptible de estudio; es la que delimita el investigador con los criterios de inclusión y exclusión.

La "población de estudio" es de la que realmente se recogen los datos; suele ser la muestra de estudio.

Cálculo del Tamaño de la Muestra

En este sentido se va a realizar el cálculo del tamaño de la muestra ya que no se puede encuestar a todos los involucrados en la atención a los usuarios, empleados y directivos, entonces se obtendrá el tamaño de la muestra de estudio por cada tipo de población para tener una idea de cómo se está llevando el manejo de ventas en Ferrecentro Oñate.

Cálculo del tamaño de la muestra cuando la población es finita

Si se conoce el tamaño de la población, es decir, la población es finita como en este caso de nuestra investigación y deseamos saber cuántos del total tendremos que estudiar, la fórmula es:

Formula:

n= Tamaño de la muestra.

z= Valor de confianza.

p= Población.

$$\boldsymbol{n = \frac{z * p}{(p.1)(z^2/2^2) + z^2}}$$

$$n = \frac{456{,}5}{(913 * 1)\left(\frac{0.5^2}{2^2}\right) + 0.5^2}$$

$$n = \frac{456{,}5}{(913)\left(\frac{0.025}{4}\right) + 0.25}$$

$$n = \frac{456{,}5}{(913)(0.00625) + 0.25}$$

$$n = \frac{456,5}{5,96}$$

$$n = 77$$

3.3 MÉTODOS, TÉCNICAS E INSTRUMENTOS DE LA INVESTIGACIÓN

3.3.1 MÉTODOS

MÉTODO DEDUCTIVO

El trabajo masivo con cumplimiento eficaz, nos permite a los usuarios tener una mayor calidad en atención, al empleado facilidad para manipular información de mercadería, así brindar un buen servicio. Para que todo esto allá se llego a concluir la necesidad de un Sistema de Ventas, con la finalidad de brindar un mejor rendimiento a la Empresa y a quienes se sirven de ella.

MÉTODO INDUCTIVO

Este Software se implementara para mejorar el proceso de eficiencia dentro de la empresa mejorando la calidad de atención y rapidez a los usuarios y así de esta manera obtener resultados prósperos a futuros, cumpliendo con todas las expectativas de la misma.

MÉTODO CIENTIFICO

Mediante el proceso de investigación de Ferrecentro Oñate se obtuvo resultados propicios que dieron lugar a dar una nueva propuesta que ayudara a dar facilidades eficientes en el manejo rapidez en los procesos de la empresa.

De esta manera tendríamos un sistema confiable con soluciones inmediatas y precisas sobre cada pedido o solicitud por los usuarios.

En el proceso de investigación realizado se obtuvo diferentes resultados que se dieron a conocer a través de una observación precisa de datos recopilados que nos llevaron a ver la verdadera problemática que hay en la empresa.

Con la información obtenida de la investigación se realizo una tabulación continua sobre el desarrollo y desempeño en el manejo de la empresa lo cual nos dio a conocer la problemática a través de una entrevista a los diferentes usuarios y así de esta manera poder proporcionar una solución que convenga para todos.

El propósito de este proceso investigativo se hizo con la finalidad de mejorar el manejo y desempeño que se ofrece a cada uno de los usuarios, dando facilidad de respuestas a cada uno de sus pedidos sin demoras y con soluciones inmediatas.

3.3.2 TÉCNICAS

ENTREVISTA

Al realizar las entrevistas a los Usuarios podremos analizar las necesidades y el manejo que se da en Ferrecentro Oñate S.A.

ENCUESTA

Las encuestas que realizaremos a los usuarios, nos permitirán establecer los inconvenientes que se encuentran en Ferrecentro Oñate S.A.

3.3.3 INSTRUMENTOS

Mediante un cuestionario de preguntas que realizaran de forma interna se podrá establecer cuáles son los problemas que presenta la empresa en área de ventas.

INSTRUMENTO (ENCUESTA)

Dirigida a: Los usuarios de la Ciudad de Babahoyo.

Objetivo: Identificar las necesidades de los usuarios, para mejorar el servicio que brinda Ferrecentro Oñate.

1) ¿En el área de Ventas de Ferrecentro Oñate hay personal que den excelentes información a los Usuarios?

$Sí$ No

2) ¿Cree usted que la Atención al público es de manera eficiente?

$Sí$ No

3) ¿En el área de ventas se guarda la información de manera manual?

$Sí$ No

4) ¿En el departamento de ventas existe material técnico disponible?

$Sí$ No

5) Le gustaría que existiera un sistema Informático que facilite al usuario hacer su solicitud de pedido directamente a la Empresa

$Sí$ No

6) Le gustaría que sea de calidad la atención en la empresa

$Sí$ No

7) Le parece importante que se dé facilidad de pago al momento de la compra

$Sí$ No

8) Le gustaría que existiera un sitio específico en internet alternativo a un buscador que le facilite encontrar accesorios y repuestos de la mercadería de venta

$Sí$ No

9) Considera que los sistemas y las comunicaciones actuales son más accesibles y fáciles que en años anteriores

$Sí$ No

10) Considera segura una transacción por internet

$Sí$ No

3.3.4 Aplicación de la Metodología

En esta aplicación de la encuesta se utilizó como instrumento de recopilación de datos en un cuestionario para realizar la encuesta respectiva.

También se utilizó información bibliográfica y apoyándose en la dirección del tutor se elaboró la encuesta con sus respectivas preguntas, las mismas que se realizaron con un conteo cualitativo y cuantitativo.

La observación

Es una técnica que consiste en observar atentamente el fenómeno, hechos o casos, tomar información y registrarla para su posterior análisis en el proceso de la investigación.

Nos basamos en la observación directa y la indirecta:

Es directa.- Cuando el investigador trata directamente o personalmente el hecho o el problema para obtener los resultados.

Es indirecta.- Cuando el investigador adquiere el conocimiento del hecho o del fenómeno analizando a través de las observaciones anteriormente por otra persona.

3.3.5 Interpretación de los resultados de la aplicación de entrevista en la Ciudad de Babahoyo.

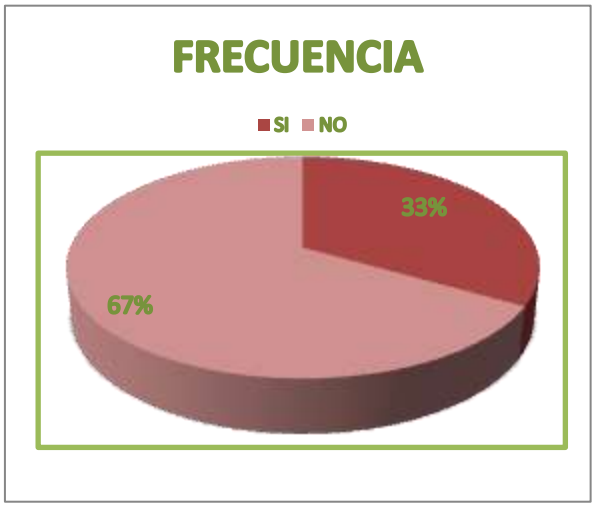
A la muestra de 913 personas se le realizo el siguiente análisis detallado a continuación con tablas y gráficos para una mejor representación y análisis

1) **¿En el área de Ventas de Ferrecentro Oñate hay personal que den excelentes información a los Usuarios?**

CUADRO 1

	Frecuencia	Porcentaje
SI	30	33%
NO	60	67%
TOTAL	90	100%

GRÁFICO



Con el resultado se obtuvo un porcentaje mínimo de usuarios que opinaron a favor del servicio ofrecido.

2) ¿Cree usted que la Atención al público es de manera eficiente?

CUADRO 2 **GRÁFICO 2**

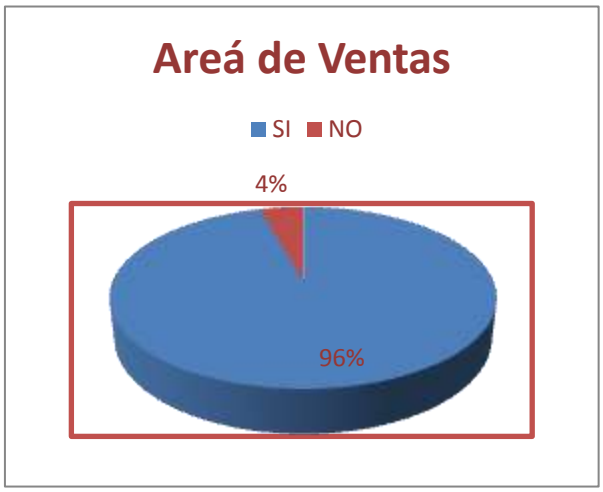
	Frecuencia	Porcentaje
SI	35	8%
NO	55	92%
TOTAL	90	100%

El 92**%** designan que la atención no se lleva de una manera adecuada, quizás porque el departamento de ventas no cuenta con un sistema que le facilite realizar esto y es por eso que los técnicos no tienen una coordinación para brindar el servicio de manera eficiente.

3) ¿En el área de ventas se guarda la información de manera manual?

CUADRO 3 **GRÁFICO 3**

	Frecuencia	Porcentaje
SI	70	4%
NO	20	96%
TOTAL	90	100%



En el análisis que se realizó, se concluyó en que el grupo menor cree que no se guarda la información de forma manual, esto puede ser por apoyar al personal de área de

ventas. Pero con respecto al 96% de los encuestados nos dimos cuenta que dicho departamento aún no cuenta con la suficiente tecnología para almacenar sus registros de una manera actualizada.

4) ¿En el departamento de ventas existe material técnico disponible?

CUADRO 4

	Frecuencia	Porcentaje
SI	80	80%
NO	20	20%
TOTAL	100	100%

GRÁFICO 4

Con respecto a los resultados obtenidos, observamos que el 80% de los encuestados reconocen que si tienen materiales disponibles y esto nos permite decir que el departamento de información si cuenta con material técnico disponible por lo cual se descarta que este sea un motivo por lo que el servicio técnicos no es eficiente.

5) Le gustaría que existiera un sistema Informático que facilite al usuario hacer su solicitud de pedido directamente a la Empresa.

CUADRO 5

	Frecuencia	Porcentaje
SI	90	90%
NO	10	10%
TOTAL	100	100%

GRAFICO 5

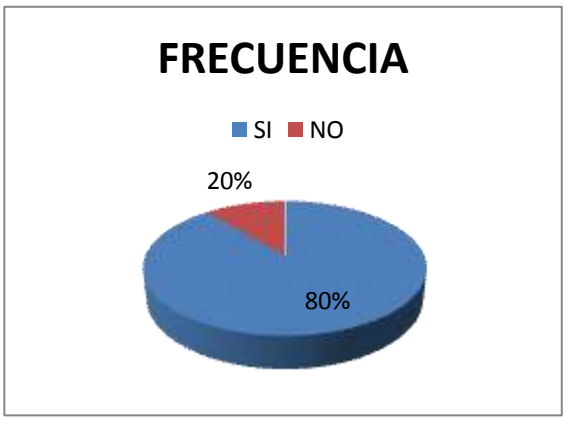
Según los encuestados se considera como una alternativa que existiese un sistema Informático, donde se le facilite al usuario hacer su solicitud de pedidos directamente al departamento de ventas, pero el otro grupo menor, considera que no es necesario, quizás porque ellos no necesita un contacto directo o están satisfechos con la eficiencia de los técnicos.

6) Le gustaría que sea de calidad la atención en la Empresa.

CUADRO 6

	Frecuencia	Porcentaje
SI	80	80%
NO	20	20%
TOTAL	100	100%

GRAFICO 6



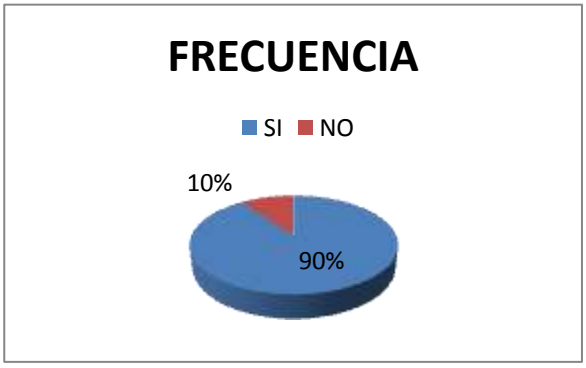
Con respecto a los encuestados la mayor parte, considera como una alternativa que si le gustaría tener una buena calidad en la atención a sus requerimientos, pero el otro grupo menor, considera que no es necesario, quizás porque están satisfechos con la atención.

7) **Le parece importante que se dé facilidad de pago al momento de la compra.**

CUADRO 7

	Frecuencia	Porcentaje
SI	90	90%
NO	10	10%
TOTAL	100	100%

GRAFICO 7



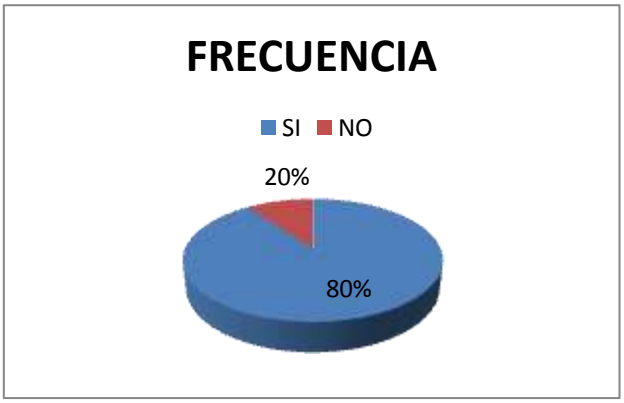
Se considera como una alternativa importante que se muestren opciones de perfiles de usuarios, donde se le facilite al usuario tener comodidades de pago.

8) Le gustaría que existiera un sitio específico en internet alternativo a un buscador que le facilite encontrar accesorios y repuestos de la mercadería de venta.

CUADRO 8

	Frecuencia	Porcentaje
SI	80	80%
NO	20	20%
TOTAL	100	100%

GRAFICO 8



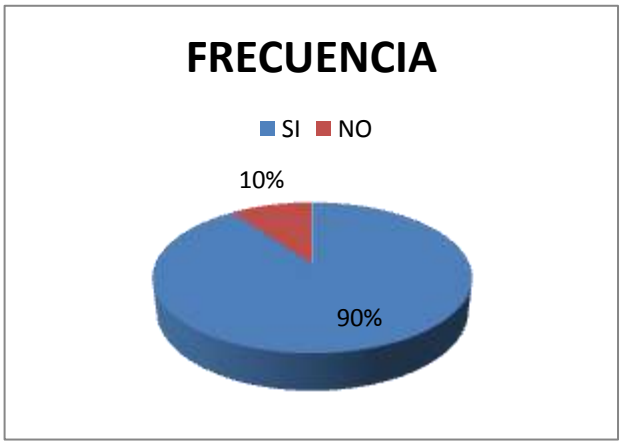
La mayor parte, considera como una alternativa importante que se existiera un sitio especifico en internet alternativo que le facilite con facilidad la búsqueda de accesorios ante de acercarse a la empresa, pero el otro grupo menor, considera que no es necesario.

9) Considera que los sistemas y las comunicaciones actuales son más accesibles y fáciles que en años anteriores.

CUADRO 9

	Frecuencia	Porcentaje
SI	90	90%
NO	10	10%
TOTAL	100	100%

GRAFICO 9



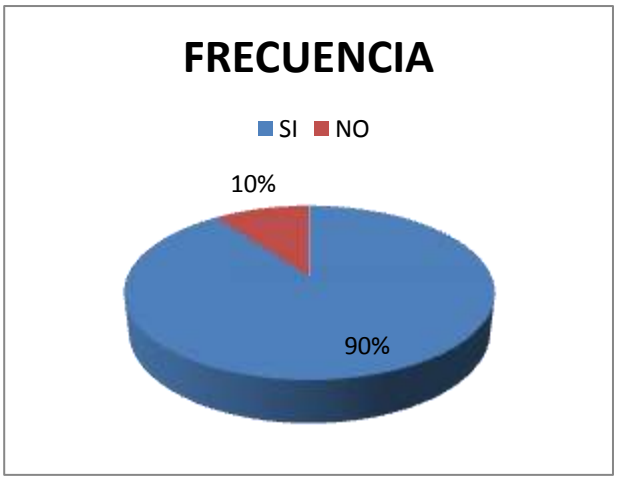
La mayor parte, considera **que los sistemas y las comunicaciones actuales son más accesibles y fáciles que en años anteriores para la empresa**, pero el otro grupo menor, considera que no es necesario.

10) Considera segura una transacción por internet.

CUADRO 10

	Frecuencia	Porcentaje
SI	90	90%
NO	10	10%
TOTAL	100	100%

GRAFICO 10



La mayor parte, considera que si es segura una búsqueda de Empleo por internet pero el otro grupo menor, considera que no es necesario.

3.4 CONCLUSIONES Y RECOMENDACIONES

3.4.1 Conclusiones

La presente investigación se ha dedicado al estudio técnico en el manejo de información de la Empresa Ferrecentro Oñate S.A. en base a solicitudes y respuestas por pedidos de los diferentes usuarios, se ha utilizado diferentes métodos que han hecho posible conocer la problemática del sistema de cual nos servimos.

Del análisis de los aspectos teóricos encontrados y de las respectivas entrevistas y de las tabulaciones realizadas se llegó a conocer la profundidad del problema en el sistema que maneja la Empresa.

Mediante esta investigación se dio a conocer qué tipo de servicio es el que ofrece la institución a sus usuarios.

Con la utilización de herramientas y métodos se llego a saber que tan eficiente es el manejo de datos y si cuenta con el material técnico suficiente para brindar un buen servicio.

Se ha determinado mediante esta investigación que puede haber un mayor equilibrio si se dieran a conocer las medidas de seguridad posible en el manejo y eficacias hacia todo pedido y consulta por el usuario, claro que también contando con que el servicio sea mucho más comprometido y de resultados inmediatos.

3.4.2 Recomendaciones

Extender los estudios expuestos sobre lo que quiere el usuario en base a tiempos de respuestas mediante una investigación o auditoria del trabajo empleado anteriormente hasta la actualidad.

Trabajar en mejorar el modelo dinámico utilizado para determinar la variación del servicio que ofrece la Empresa.

Analizar con mayor detenimiento, y buscar la razón de, él porque el servicio que ofrece la institución no es tan acogido por sus usuarios.

Dar a conocer a los usuarios a través de anuncios, que propósito tiene y da la empresa a sus clientes sobre los productos que ofrecen.

Mejorar el proceso de atención con soportes técnicos que permitan la rapidez de envíos de información entre sus usuarios.

CAPITULO IV

4. MARCO PROPOSITIVO

4.1 TITULO

Sistema Informático para mejorar las ventas en Ferrecentro Oñate S.A. de la ciudad de Babahoyo en la Provincia de los Ríos.

4.2 Desarrollo de Propuesta

Mediante una investigación a fondo y comprensiva del estudio practico y dedicado hemos llegado de manera directa a gestionar un sistema informático que ayude a mejorar el servicio de ventas en Ferrecentro Oñate S.A. para que de soluciones en atención, producción y amplitud al manejo preciso con alta calidad de desempeño de soluciones inmediatas.

Con este sistema daremos al usuario soluciones a sus pedidos de forma precisa sin molestia alguna registrando cual tipo de información que de alguna manera más adelante sea solicitada.

Cumpliendo con todos los requerimientos de la empresa dejamos a conocer que el sistema que se implantara será de mayor utilidad y avance para la empresa.

4.2.1 DISEÑO DEL SISTEMA

4.2.1.1 REQUERIMIENTOS DEL HARDWARE

- Procesador Intel(R) Pentium ® 4 CPU 3.06 GHZ.
- Memoria RAM 4 GB
- Disco Duro 160 GB. O 5OO GB
- Monitor.
- Mouse.
- Teclado.
- Impresora.
- CDS Write 52 x 32 x 52.
- Scanner Bend

4.2.1.2 REQUERIMIENTOS DEL SOFTWARE

- Microsoft Sql Server 2005

- Visual Studio 2008
- Windows 7
- Microsoft Office 2010

4.2.1.3 PROCESO DE INSTALACIÓN

Para la instalación de nuestro sistema debemos seguir los siguientes pasos:

- Ejecutar el Archivo .exe del Sistema.
- Configurar los Elementos Necesarios de Instalación..
- Configurar la base de datos en el servidor local y listo.
- Abrimos la Aplicación del Sistema y a Trabajar.

4.2.1.4 SEGURIDADES

Este sistema tiene dos perfiles de usuario: que son Administrador, Técnico Usuario. Cada uno de ellos cumple una función en el sistema, la cual se la define de la siguiente manera:

Administrador: es quien tenga el permiso de manipular la información que este contiene mediante la clave de acceso, en donde se lleva el registro del ingreso de los usuarios ingresados, las solicitudes que el Usuario envía, la asignación para cada técnico de las diferentes áreas de trabajos que ingresan los usuarios.

Técnico: es único, porque la información que contiene solo le compete al técnico asignado, es decir el administrador asigna un caso a técnicos diferentes y ellos mediante su clave acceden al sistema para verificar el caso y luego llenan un formulario de acuerdo a la solicitud asignada, el cual es enviado al administrador para que el verifique que solicitud fue terminada y cual esta pendientes.

4.2.1.5 Alcances de Software

El software implementado presentara:

- Diseño Partidario para los Usuarios
- Fácil manejo.
- Control de manejo de datos.
- Control de Base de datos.

4.2.1.6 Delimitaciones y limitaciones del Software

- Realizar Búsqueda de Productos
- Realizar Modificaciones en Facturaciones
- Realizar Consultas y Eliminaciones de Productos

Permitirá un Control de:

- Control de Ingresos
- Control de Usuarios
- Control de Ventas
- Control de Modificación
- Control de Eliminación

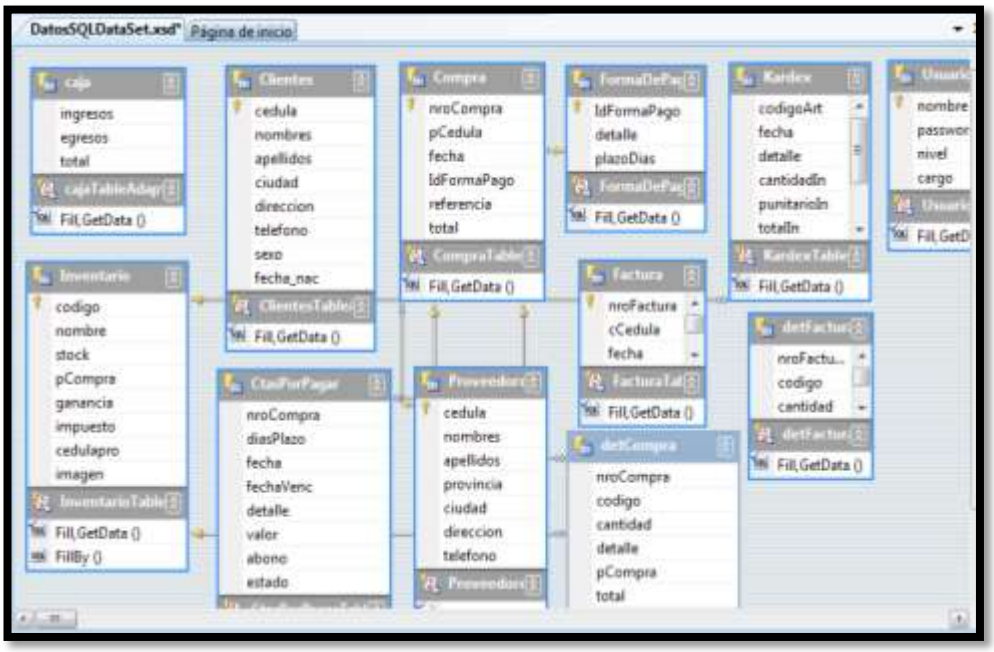
4.2.1.7 Descripción de los módulos.

Las funciones principales del sistema son:

- ✓ Ingreso al sistema por medio de login y password.
- ✓ Pantalla principal del sistema.
- ✓ Pantalla de mantenimiento de artículos.
- ✓ Pantalla de mantenimiento de usuario.
- ✓ Pantalla de compra.
- ✓ Pantalla de venta.
- ✓ Cancelar pedidos.
- ✓ Venta de artículos.
- ✓ Inventarios.
- ✓ Consulta de artículos.
- ✓ Consulta de solicitante.
- ✓ Consulta de pedidos.
- ✓ Reportes de artículos.
- ✓ Reporte de solicitante.
- ✓ Reporte de pedido.
- ✓ Salida del sistema.

4.2.2 DISEÑO DE INTERFAZ

Base de datos del sistema relacionada



1.-La primera ventana que aparece cuando arranca el sistema de facturación de Ferrecentro Oñate, es el de acceso al mismo, en donde se encuentran dos botones como son:

Permite ingresar al sistema al dar clic sobre el botón.

No Permite ingresar al sistema al dar clic sobre el botón.

2.- Después de haber permitido el ingreso al sistema de facturación, aparece una ventana de autorización al mismo como se muestra a continuación:

En **LOGIN:** se debe ingresar el nombre de usuario que por defecto será **WILSON** y en **PASSWORD:** se debe de ingresar la contraseña que por defecto será **3434**

Aparecerá una ventana de bienvenida al sistema como esta:

3.- Después de dar la autorización al sistema aparece la ventana principal del programa el cual contiene los menús de cada proceso:

Al dar clic en la opción MENU; aparece una ventana activa en donde se encuentra todos los formularios de acceso a procesos como se muestra a continuación:

En el siguiente menú como es: REGISTRAR, aparece las siguientes opciones como es:

Al dar clic en la opción Clientes, aparece la siguiente ventana con información del cliente, en la cual se registra los datos personales como se muestra a continuación:

VENTANA DE FORMULARIO DE CLIENTES

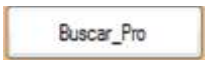
BOTONES DE MANTENIMIENTO

	Permite ingresar datos de un nuevo cliente.
	Permite guardar información del cliente, para ello debe ingresar los datos que el sistema solicite como: cedula, nombres, apellidos, etc..
	Permite actualizar información del cliente, para que esta opción sea accesible se debe primero buscar al cliente mediante el botón **BUSCAR_Cli.**
	Permite eliminar al cliente con todos los datos personales, para que esta opción sea accesible se debe primero buscar al cliente mediante el botón **BUSCAR_ Cli.**
	Permite cancelar la operación que se esté procesando ya sea guardar, modificar y eliminar.
	Permite cerrar la ventana activa.
	Permite buscar los datos del cliente.
	Son aquellos que permite movilizarse a los registros que se encuentran almacenados en la base de datos ya sea: al primer registro, siguiente registro, al registro anterior y al último registro de la tabla Clientes.

Al dar clic en la opción Proveedores, aparece la siguiente ventana con información del proveedor, en la cual se registra los datos personales como se muestra a continuación:

VENTANA DE FORMULARIO DE PROVEEDORES

BOTONES DE MANTENIMIENTO

	Permite ingresar datos de un nuevo proveedor.
	Permite guardar información del proveedor, para ello debe ingresar los datos que el sistema solicite como: cedula,
	Permite actualizar información del proveedor, para que esta opción sea accesible se debe primero buscar al proveedor
	Permite eliminar al proveedor con todos los datos personales, para que esta opción sea accesible se debe primero buscar al
	Permite cancelar la operación que se esté procesando ya sea guardar, modificar y eliminar.
	Permite cerrar la ventana activa.
	Permite buscar los datos del proveedor.
	Son aquellos que permite movilizarse a los registros que se encuentran almacenados en la base de datos ya sea: al primer

Al dar clic en la opción Usuarios, aparece la siguiente ventana con información del usuario, en la cual se registra los datos personales como se muestra a continuación:

VENTANA DE INGRESO DE USUARIOS

BOTONES DE MANTENIMIENTO

	Permite ingresar datos de un nuevo usuario.
	Permite guardar información del usuario, para ello debe ingresar los datos que el sistema solicite como: nombre, password, nivel y
	Permite actualizar información del usuario, para que esta opción sea accesible se debe primero buscar al usuario mediante el
	Permite eliminar al usuario con todos los datos ingresados, para que esta opción sea accesible se debe primero buscar al usuario
	Permite cancelar la operación que se esté procesando ya sea guardar, modificar y eliminar.
	Permite buscar los datos del usuario en la BD.

	Permite cerrar la ventana activa.

Al dar clic en la opción Inventarios, aparece la siguiente ventana con información de los productos ya registrados y en la cual se registra la mercadería como se muestra a continuación:

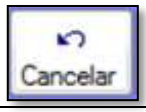
VENTANA DE INGRESO DE MERCADERÍAS

BOTONES DE MANTENIMIENTO

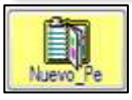
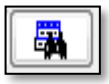
	Permite ingresar datos de un nuevo producto, pero para eso se debe de ingresar un código como por ejemplo: COD0001, después de ingresar el código asignado al producto, se debe dar clic al botón que se encuentra al lado del ingreso del código, es aquel que permite verificar si el código ya está ingresado a la base de datos, caso contrario permite registrar el producto.

	Permite actualizar información del producto, para que esta opción sea accesible se debe primero buscar al producto mediante el botón **BUSCAR.**
	Permite guardar información del producto, para ello debe ingresar los datos que el sistema solicite como: código, nombre, stock, cedula del proveedor y el precio de compra, automáticamente el sistema calcula la ganancia y el impuesto del producto por registrar.
	Permite eliminar al usuario con todos los datos ingresados, para que esta opción sea accesible se debe primero buscar al usuario mediante el botón **BUSCAR**
	Permite buscar el producto sea para modificar los datos o eliminar dicho producto.
	Permite cerrar la ventana activa

VENTANA DE FORMA DE PAGO

	Permite ingresar datos de una forma de pago.
	Permite guardar información de la nueva forma de pago, para ello debe ingresar los datos que el sistema solicite como: Id
	Permite actualizar información de las formas de pago, para que esta opción sea accesible se debe primero buscar la Id Forma
	Permite cancelar la operación que se esté procesando ya sea guardar y modificar.
	Permite buscar la identificación de la forma de pago, ya sea para modificarla o eliminarla.
	Permite cerrar la ventana activa.

VENTANA DE FORMULARIO DE VENTAS

	Permite ingresar una nueva venta y generar una nueva factura enumerada como por ejemplo: **Factura Nº 000005**
	Permite guardar información de la venta a la base de datos.
	Permite cancelar la operación que se esté procesando ya sea guardar, modificar y eliminar.
	Permite visualizar la factura de ventas en el momento de que se han ingresado los datos.
	Permite cerrar la ventana activa.
	Permite activar la venta de productos en la que se seleccionara el producto para la venta**.**
	Permite modificar el pedido de la venta de productos. Debe seleccionar con el puntero el producto a modificar.
	Permite eliminar el pedido de la venta de productos seleccionada. Debe seleccionar con el puntero el producto a
	Permite buscar los datos del cliente en el caso de que este registrado en la base de datos del sistema.

FORMULARIO DE (ACTUALIZACIÓN) LISTA DE PRODUCTOS DE FACTURACIÓN.

El formulario Factura se enlaza con este formulario para seleccionar los productos requeridos en el pedido y se van almacenando en el Datagrid de la Factura. A continuación se detalla ciertos botones que especifican sus funciones:

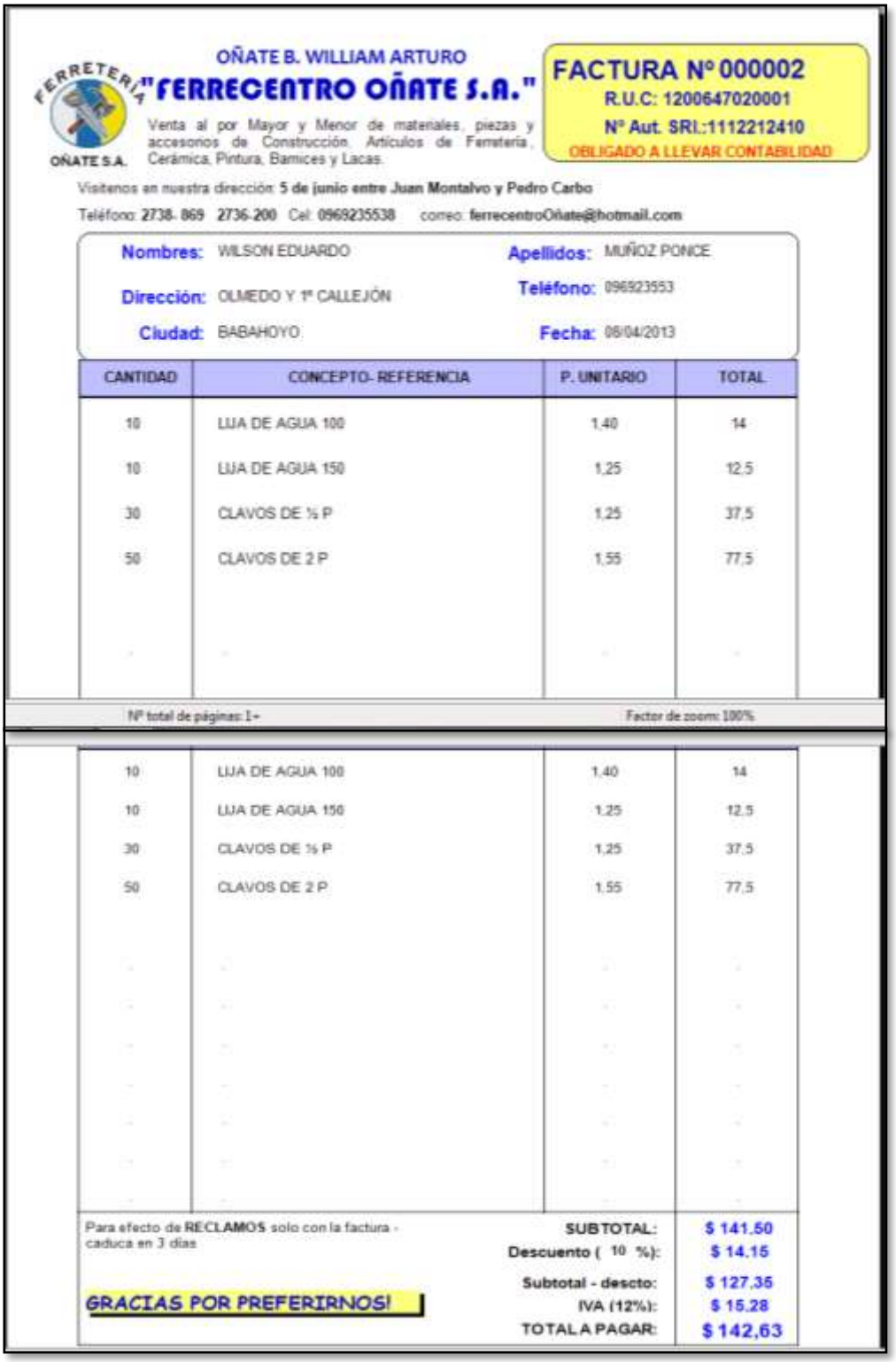
BOTONES DE MANTENIMIENTO

	Permite buscar en el inventario el producto requerido para la venta, en la cual se debe ingresar el código
	Permite buscar en el inventario el producto requerido para la venta, en la cual se debe ingresar el nombre
	Permite enviar al formulario de Factura, el producto o artículo requerido para la venta.
	Permite cancelar el proceso de venta en los artículos.
	Permite salir del formulario activo
	Permite actualizar el stock del artículo en existencia y a la vez la cantidad de pedidos del producto a

Una vez realizado los pedidos para la venta se mostrara un formulario de la siguiente manera y en la cual se debe seguir ciertos pasos para el pago e impresión de la factura.

Una vez definido la venta de los productos se debe guardar la información en la base de datos y se desplaza al formulario de pago como se muestra en la figura.

Después de haber cancelado el total a pagar se imprime la factura en la cual se muestra de la siguiente manera:



OÑATE B. WILLIAM ARTURO

"FERRECENTRO OÑATE S.A."

Venta al por Mayor y Menor de materiales, piezas y accesorios de Construcción. Artículos de Ferretería, Cerámica, Pintura, Barnices y Lacas.

FACTURA Nº 000002
R.U.C: 1200647020001
Nº Aut. SRI.:1112212410
OBLIGADO A LLEVAR CONTABILIDAD

Visítenos en nuestra dirección: **5 de junio entre Juan Montalvo y Pedro Carbo**

Teléfono: **2738- 869 2736-200** Cel: **0969235538** correo: **ferrecentroOñate@hotmail.com**

Nombres:	WILSON EDUARDO	**Apellidos:**	MUÑOZ PONCE
Dirección:	OLMEDO Y 1º CALLEJÓN	**Teléfono:**	096923553
Ciudad:	BABAHOYO	**Fecha:**	06/04/2013

CANTIDAD	CONCEPTO- REFERENCIA	P. UNITARIO	TOTAL
10	LIJA DE AGUA 100	1,40	14
10	LIJA DE AGUA 150	1,25	12,5
30	CLAVOS DE ½ P	1,25	37,5
50	CLAVOS DE 2 P	1,55	77,5

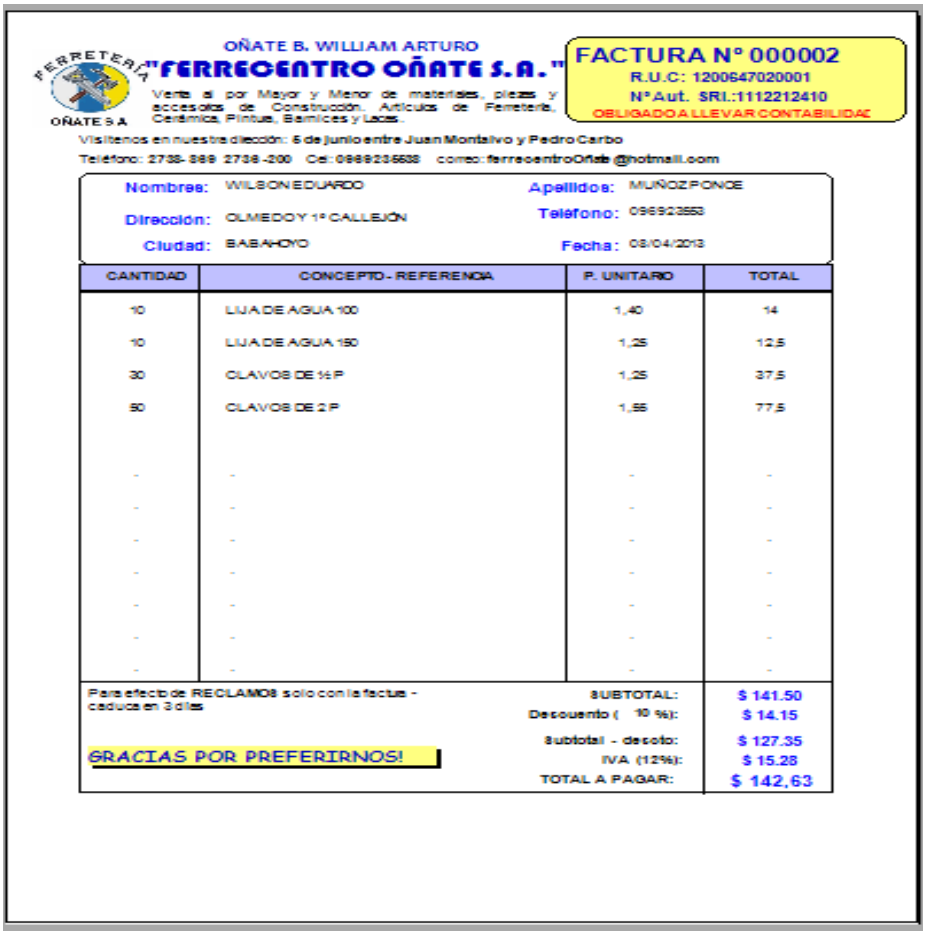
Nº total de páginas: 1+ Factor de zoom: 100%

10	LIJA DE AGUA 100	1,40	14
10	LIJA DE AGUA 150	1,25	12,5
30	CLAVOS DE ½ P	1,25	37,5
50	CLAVOS DE 2 P	1,55	77,5

Para efecto de **RECLAMOS** solo con la factura - caduca en 3 días

GRACIAS POR PREFERIRNOS!

SUBTOTAL:	$ 141,50
Descuento (10 %):	$ 14,15
Subtotal - descto:	$ 127,35
IVA (12%):	$ 15,28
TOTAL A PAGAR:	$ 142,63



OÑATE B. WILLIAM ARTURO

"FERRECENTRO OÑATE S.A."

Venta al por Mayor y Menor de materiales, piezas y accesorios de Construcción. Artículos de Ferretería, Cerámica, Pinturas, Barnices y Lacas.

FACTURA N° 000002
R.U.C: 1200647020001
N° Aut. SRI.:1112212410
OBLIGADO A LLEVAR CONTABILIDAD

Visítenos en nuestra dirección: 5 de junio entre Juan Montalvo y Pedro Carbo
Teléfono: 2738- 889 2738 -200 Cel: 0989235633 correo: ferrecentroOñate @hotmail.com

Nombres: WILSON EDUARDO
Apellidos: MUÑOZ PONCE
Dirección: OLMEDO Y 1° CALLEJÓN
Teléfono: 096923553
Ciudad: BABAHOYO
Fecha: 08/04/2013

CANTIDAD	CONCEPTO - REFERENCIA	P. UNITARIO	TOTAL
10	LIJA DE AGUA 100	1,40	14
10	LIJA DE AGUA 150	1,25	12,5
30	CLAVOS DE 1½ P	1,25	37,5
50	CLAVOS DE 2 P	1,55	77,5

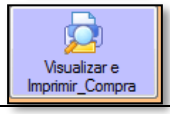
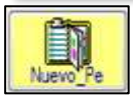
Para efecto de RECLAMOS solo con la factura - caduca en 3 días

GRACIAS POR PREFERIRNOS!

SUBTOTAL:	$ 141.50
Descuento (10 %):	$ 14.15
Subtotal - descto:	$ 127.35
IVA (12%):	$ 15.28
TOTAL A PAGAR:	$ 142,63

VENTANA DE FORMULARIO DE COMPRAS

BOTONES DE MANTENIMIENTOS.

	Permite ingresar una nueva compra de productos y generar una nueva Compra enumerada como por ejemplo: **Compra Nº**
	Permite guardar información de la compra a la base de datos.
	Permite cancelar la operación que se esté procesando ya sea guardar, modificar y eliminar.
	Permite visualizar e imprimir la compra de los productos ingresados en el inventario.
	Permite de salir de la ventana de compra activa.
	Permite activar la compra de productos en la que se seleccionara el producto para la compra**.**
	Permite modificar el pedido de la compra de productos. Debe seleccionar con el puntero el producto a modificar.
	Permite eliminar el pedido de la compra de productos seleccionada. Debe seleccionar con el puntero el producto a
	Permite buscar los datos del proveedor en el caso de que este registrado en la base de datos del sistema.

FORMULARIO DE (ACTUALIZACIÓN) LISTA DE PRODUCTOS DE COMPRA.

El formulario Compra se enlaza con este formulario para incrementar el stock de los diferentes productos en la cual también se actualiza el precio de compra en el caso de que se haya incrementado en el costo. A continuación se detalla ciertas funciones de los botones:

BOTONES DE MANTENIMIENTO

	Permite buscar en el inventario el producto requerido para la compra, en la cual se debe
	Permite buscar en el inventario el producto requerido para la compra, en la cual se debe
	Permite enviar al formulario de Compra, el producto o artículo requerido para la compra.
	Permite cancelar el proceso de compra en los artículos.
	Permite salir del formulario activo
	Permite actualizar el stock del artículo en existencia y también se puede actualizar el precio

VENTANA DE FORMULARIO DE CUENTAS POR PAGAR

BOTONES DE MANTENIMIENTO:

	Permite buscar la cuenta por pagar al proveedor por medio del número de la compra al sistema.
	Permite abonar el saldo de la cuenta por pagar a su proveedor y guardar la información en la base de datos del sistema.
	Permite imprimir un reporte de la deuda con el proveedor.
	Permite cerrar la ventana activa.

Al hacer clic en el botón imprimir se obtiene el siguiente reporte que indica los datos que se tiene de la deuda con el proveedor como son los siguientes:

FORMULARIO DE DEVOLUCIONES EN VENTAS

BOTONES DE MANTENIMIENTO.

	Permite buscar el código del producto para actualizar el stock en el inventario
	Permite buscar el nombre del producto para actualizar el stock en el inventario
	Permite modificar (actualizar) el stock de acuerdo a la devolución del producto
	Permite salir de la ventana activa.

FORMULARIO DE DEVOLUCIONES EN COMPRAS

BOTONES DE MANTENIMIENTO

	Permite buscar el código del producto para actualizar el stock en el inventario, al disminuir el producto.
	Permite buscar el nombre del producto para actualizar el stock en el inventario, al disminuir el producto
	Permite guardar a la tabla de inventarios la cantidad de producto devuelto al Proveedor, actualizando el stock

	Permite salir de la ventana activa.

FORMULARIO DE ARQUEO DE CAJAS

VENTAS ANTERIORES: Permite consultar las ventas que se han generado desde una fecha determinad hasta otra fecha determinada por el usuario.

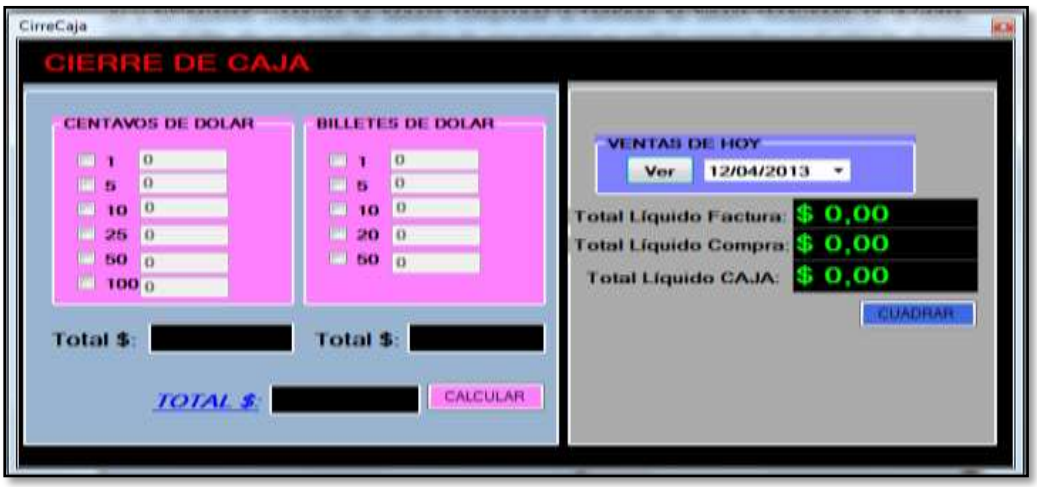
VENTAS DE HOY: Permite consultar solo las ventas que se han realizado en la presente fecha que se encuentre el sistema.

El botón imprimir permite visualizar la información del total de ventas, compras y cuentas por pagar, como se muestra en la siguiente ventana:

FORMULARIO DE CIERRE DE CAJA

En este formulario de cierre de caja solo se deberá ingresar el valor que se obtiene de la venta del día, que serán valores en centavos de dólar como en billetes.

Se deberá especificar la cantidad de dinero que se encuentra en la venta del día como por ejemplo: la cantidad de 1 , 5, 10, 25, 50 centavos de dólar, así como los billetes de 1, 5, 10, 20 y 50 dólares. Después se deberá comprobar la cantidad de dinero recaudado en la venta del día al dar clic en el botón cuadrar de esa manera se podrá comprobar si el valor de dinero que se encuentra en la caja cuadra con el valor de dinero que se ha realizado por la venta del día de acuerdo el sistema.



4.2.3 Código de programación

FORMULARIO SPLASH

```
Public Class Splash

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Timer1.Start()
        Label1.Visible = True
        Label2.Visible = True
        Label3.Visible = True
        ProgressBar1.Visible = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        ProgressBar1.Value = ProgressBar1.Value + 1
        Label1.Text = ProgressBar1.Value
        If ProgressBar1.Value = 100 Then

            Timer1.Enabled = False
            Me.Hide()
            contraseña.Show()
        End If

    End Sub

    Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Label1.Visible = False
        Label2.Visible = False
        Label3.Visible = False
        ProgressBar1.Visible = False

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        Me.Close()
    End Sub
```

```
    Private Sub lblPlatform_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles lblPlatform.Click

    End Sub
End Class
```

FORMULARIO CONTRASEÑA

```
Option Strict On
Option Explicit On

Imports Microsoft.VisualBasic
Imports System
Imports System.Windows.Forms
Imports System.Drawing

' Importaciones para el acceso a datos
Imports System.Data
' Si se va a trabajar con una base de SQL Server
Imports System.Data.SqlClient
' Si se va a trabajar con una base de Access
'Imports System.Data.OleDb
Public Class contraseña
    Inherits System.Windows.Forms.Form
    Private veces As Integer = 0
    Private Const NumeroIntentos As Integer = 3



    Private Sub Form4_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Usuario' Puede moverla o quitarla según sea
necesario.
        Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Dim a, b As String

        a = usu.Text
        b = con.Text
        If comprobarUsuario(Me.usu.Text, Me.con.Text) Then
            MsgBox("Bienvenido(a)  " & a, MsgBoxStyle.Information,
"Seguridad")
            Form1.Show()
            Me.Hide()
        Else
            ' Permitir varios intentos
            veces = veces + 1
            If veces < NumeroIntentos Then
                MsgBox("Usuario o Contraseña Incorrecta")
                MsgBox("Quedan " & (NumeroIntentos - veces) & "
intentos")
                usu.Text = ""
                con.Text = ""
                usu.Focus()
```

```
                Exit Sub
            End If
            MsgBox("Lo sentimos No puede Ingresar al SISTEMA",
MsgBoxStyle.Critical, "Seguridad")
            Me.DialogResult = DialogResult.No
        End If
        Hide()
    End Sub

    Private Sub UsuarioBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.UsuarioBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

    End Sub
    ''*******************CONEXIÓN A LA BASE DE
DATOS*****************************************
    Private cadenaCnn As String = "Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True"


''*****************************************************************
**********************

    ' Función para comprobar si el acceso es correcto
    Private Function comprobarUsuario( _
                ByVal nombre As String, _
                ByVal clave As String) As Boolean
        ' Conectar a la base de datos
        Dim cnn As SqlConnection = Nothing
        '
        Try
            ' Conectar a la base de datos de SQL Server
            ' (la cadena debe estar inicializada previamente)
            cnn = New SqlConnection(cadenaCnn)
            cnn.Open()
            ' Definir la cadena que vamos a usar para comprobar
            ' si el usuario y el password son correctos.
            ' Utilizo parámetros para evitar inyección de código.
            Dim sel As New System.Text.StringBuilder
            ' Usando COUNT(*) nos devuelve el total que coincide
            ' con lo indicado en el WHERE,
            ' por tanto, si la clave y el usuario son correctos,
            ' devolverá 1, sino, devolverá 0
            sel.Append("SELECT COUNT(*) FROM Usuario ")
            sel.Append("WHERE Nombre = @nombre AND password =
@password")
            ' Definir el comando que vamos a ejecutar
            Dim cmd As New SqlCommand(sel.ToString, cnn)
            ' Creamos los parámetros
            cmd.Parameters.Add("@nombre", SqlDbType.NVarChar, 50)
            cmd.Parameters.Add("@password", SqlDbType.NVarChar, 40)
            '
            ' Asignamos los valores recibidos como parámetro
            cmd.Parameters("@nombre").Value = nombre
            cmd.Parameters("@password").Value = clave
            '
            ' Ejecutamos la consulta
```

```
            ' ExecuteScalar devuelve la primera columna de la primera
fila
            ' por tanto, devolverá el número de coincidencias
halladas,
            ' que si es 1, quiere decir que el usuario y el password
son correctos.
            Dim t As Integer = CInt(cmd.ExecuteScalar())
            ' Cerramos la conexión
            cnn.Close()
            '
            ' Si el valor devuelto es cero
            ' es que no es correcto.
            If t = 0 Then
                Return False
            End If

        Catch ex As Exception
            MessageBox.Show("ERROR al conectar a la base de datos: " &
vbCrLf & _
                            ex.Message, "Comprobar usuario",
MessageBoxButtons.OK, _
                            MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1)
            Return False
        Finally
            If Not cnn Is Nothing Then 'AndAlso cnn.State <>
ConnectionState.Closed Then
                cnn.Dispose()
            End If
        End Try
        '
        ' Si llega aquí es que todo ha ido bien
        Return True
    End Function

    Private Sub Btncerrar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Btncerrar.Click
        Me.Close()
    End Sub

    Private Sub usu_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles usu.TextChanged
        Me.usu.CharacterCasing = CharacterCasing.Upper
    End Sub


End Class
```

FORMULARIO PRINCIPAL

```
Imports System.Data.SqlClient
Public Class Form1


    Private Sub TsFactura_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TsFactura.Click
        Factura.Show()
    End Sub
```

```
    Private Sub Formulario_Menu_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Formulario_Menu.Click
        FormMenu.Show()
    End Sub

    Private Sub ClientesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ClientesToolStripMenuItem.Click
        Clientes.Show()
    End Sub

    Private Sub ProveedoresToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ProveedoresToolStripMenuItem.Click
        Proveedores.Show()
    End Sub

    Private Sub UsuariosToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
UsuariosToolStripMenuItem.Click
        Usuario.Show()
    End Sub

    Private Sub CToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CToolStripMenuItem.Click
        ConsultClientes.Show()
    End Sub

    Private Sub ConsultasGeneralToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ConsultasGeneralToolStripMenuItem.Click
        ConsultClientes2.Show()
    End Sub

    Private Sub TsClientes_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TsClientes.Click
        Clientes.Show()
    End Sub

    Private Sub TsProveedores_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TsProveedores.Click
        Proveedores.Show()
    End Sub

    Private Sub TsCompras_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TsCompras.Click
        Compra.Show()
    End Sub

    Private Sub TsInventarios_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TsInventarios.Click
        Inventario.Show()
    End Sub

    Private Sub TsCuentasxPagar_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TsCuentasxPagar.Click
        CuentasPorPagar.Show()
    End Sub
```

```
    Private Sub TsReportes_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TsReportes.Click

    End Sub

    Private Sub TsUsuarios_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TsUsuarios.Click
        Usuario.Show()
    End Sub

    Private Sub TsRespaldos_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TsRespaldos.Click
        'Respaldo.Show()
    End Sub

    Private Sub ConsultasEspecíficasToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
ConsultasEspecíficasToolStripMenuItem.Click
        ConsultProveedores.Show()
    End Sub

    Private Sub ConsultasGeneralToolStripMenuItem1_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ConsultasGeneralToolStripMenuItem1.Click
        ConsultProveedores2.Show()
    End Sub

    Private Sub ReportesDeClientesToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ReportesDeClientesToolStripMenuItem.Click
        Dim rpt As New ReporteClientes2
        rpt.SetDataSource(DatosSQLDataSet)
        FormularioRe.CrystalReportViewer1.ReportSource = rpt
        FormularioRe.Show()
    End Sub

    Private Sub ClientesBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.ClientesBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.caja' Puede moverla o quitarla según sea necesario.
        Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Kardex' Puede moverla o quitarla según sea necesario.
        Me.KardexTableAdapter.Fill(Me.DatosSQLDataSet.Kardex)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Usuario' Puede moverla o quitarla según sea
necesario.
        Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Proveedores' Puede moverla o quitarla según sea
necesario.
```

```
Me.ProveedoresTableAdapter.Fill(Me.DatosSQLDataSet.Proveedores)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Clientes' Puede moverla o quitarla según sea
necesario.
        Me.ClientesTableAdapter.Fill(Me.DatosSQLDataSet.Clientes)

        Me.valor.Text = CStr(0)
    End Sub

    Private Sub ReportesDeProveedoresToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
ReportesDeProveedoresToolStripMenuItem.Click
        Dim rpt1 As New ReporteProveedor
        rpt1.SetDataSource(DatosSQLDataSet)
        FormRPROVEEDOR.CrystalReportPRO.ReportSource = rpt1
        FormRPROVEEDOR.Show()
    End Sub

    Private Sub ReportesDeUsuariosToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
ReportesDeUsuariosToolStripMenuItem.Click
        Dim rpt2 As New ReporteUsuarios
        rpt2.SetDataSource(DatosSQLDataSet)
        FormRUsuario.CrystalReportUsuario.ReportSource = rpt2
        FormRUsuario.Show()
    End Sub

    Private Sub ReportToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ReportToolStripMenuItem.Click

    End Sub

    Private Sub ReporteDeKARDEXToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ReporteDeKARDEXToolStripMenuItem.Click
        Dim rpt2 As New ReporteKardex

        rpt2.SetDataSource(Me.DatosSQLDataSet)
        FormularioKardex.CrystalReportKardex.ReportSource = rpt2
        FormularioKardex.Show()
    End Sub

    Private Sub DeToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
DeToolStripMenuItem.Click
        Me.valor.Text = CStr(1)
        DevoluciónVentas.Show()
        DevoluciónVentas.Label3.Visible = True
        DevoluciónVentas.Button2.Enabled = False
        DevoluciónVentas.calStock.Enabled = False
    End Sub

    Private Sub DevolucionesEnCOMPRAToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
DevolucionesEnCOMPRAToolStripMenuItem.Click
        Me.valor.Text = CStr(2)
        DevoluciónVentas.Show()
        DevoluciónVentas.Label5.Visible = True
```

```
        DevoluciónVentas.Button2.Enabled = False
        DevoluciónVentas.calStock.Enabled = False



    End Sub

    Private Sub ToolStripButton1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ToolStripButton1.Click
        System.Diagnostics.Process.Start("Calc.exe")
    End Sub

    Private Sub ToolStripButton2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ToolStripButton2.Click
        calendario.Show()
    End Sub

    Private Sub CAToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CAToolStripMenuItem.Click
        System.Diagnostics.Process.Start("Calc.exe")
    End Sub

    Private Sub CalendarioToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CalendarioToolStripMenuItem.Click
        calendario.Show()
    End Sub

    Private Sub ArqueoDeCajaToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ArqueoDeCajaToolStripMenuItem.Click
        CirreCaja.Show()
    End Sub

    Private Sub SaldosDeCajaToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SaldosDeCajaToolStripMenuItem.Click
        CAJA.Show()
    End Sub

    Private Sub ToolStripMenuItem2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ToolStripMenuItem2.Click
        Factura.Show()
    End Sub

    Private Sub ToolStripMenuItem3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ToolStripMenuItem3.Click
        Compra.Show()
    End Sub

    Private Sub InventariosToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
InventariosToolStripMenuItem.Click
        Inventario.Show()
    End Sub
```

```
    Private Sub FormaPagoToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
FormaPagoToolStripMenuItem.Click
        FormaDePago.Show()
    End Sub

    Private Sub ToolStripMenuItem5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ToolStripMenuItem5.Click
        CuentasPorPagar.Show()
    End Sub


    Private Sub ConsultasEspecificasToolStripMenuItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles
ConsultasEspecificasToolStripMenuItem.Click
        ConsultProducto.Show()
    End Sub



    Private Sub ReporteDeCAJAToolStripMenuItem_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ReporteDeCAJAToolStripMenuItem.Click
        Dim rpt2 As New Report_CAJA
        rpt2.SetDataSource(Me.DatosSQLDataSet)
        FormularioRe.CrystalReportViewer1.ReportSource = rpt2
        FormularioRe.Show()
    End Sub

    Private Sub CopiaDeSeguridadToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
CopiaDeSeguridadToolStripMenuItem.Click
        Backup.Show()
    End Sub

    Private Sub RestarurarBDToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RestarurarBDToolStripMenuItem.Click
        RestoreBD.Show()
    End Sub

    Private Sub ToolStripButton4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ToolStripButton4.Click
        Process.Start("C:\Users\Rita\Documents\FERRECENTRO OÑATE
2013\MANUAL DE USUARIO.pdf")

    End Sub
End Class
```

FORMULARIO USUARIO.

```
Imports System.Data
Imports System.Data.SqlClient


Public Class Usuario
    Dim CONN As SqlConnection
    Dim COMANDO As SqlDataAdapter
```

```
    Dim DS As New DataSet
    Dim dw As New DataView
    Private Sub UsuarioBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.UsuarioBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

    End Sub

    Private Sub Usuario_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Usuario' Puede moverla o quitarla según sea
necesario.
        Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)

        ''CADENA DE CONEXIÓN DE LA BASE DE DATOS***********
        'CONN = New SqlConnection("Data Source=COMANDATO-
VAIO\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        CONN.Open()
        COMANDO = New SqlDataAdapter("select * from Usuario", CONN)
        COMANDO.Fill(DS, "Usuario")
        Me.UsuarioDataGridView.DataSource = DS.Tables("Usuario")
        llamado.Text = 0
        Try
            BtnNuevo.Enabled = True
            BtnGuardar.Enabled = False
            BtnCancelar.Enabled = False
            BtnBuscar.Enabled = False
            CombNIVEL.Items.Add("1")
            CombNIVEL.Items.Add("2")
            Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnNuevo.Click
        Try
            Me.NombreTextBox.Text = ""
            Me.PasswordTextBox.Text = ""
            Me.CombNIVEL.Text = ""
            Me.CombCARGO.Text = ""
            BtnNuevo.Enabled = False
            BtnGuardar.Enabled = False
            BtnModificar.Enabled = False
            BtnCancelar.Enabled = True
            BtnBuscar.Enabled = True
            Me.NombreTextBox.Enabled = True
            Me.PasswordTextBox.Enabled = True
            Me.CombNIVEL.Enabled = True
            Me.CombCARGO.Enabled = True
            Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
            Me.NombreTextBox.Focus()
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
```

```
        End Try
    End Sub

    Private Sub BtnGuardar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnGuardar.Click
        If Me.llamado.Text = 0 Then

            ''CADENA DE CONEXIÓN DE LA BASE DE DATOS*****
            'CONN = New SqlConnection("Data Source=COMANDATO-
VAIO\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")

            CONN.Open()
            COMANDO = New SqlDataAdapter("insertar_Usuario", CONN)
            COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("contar_i", SqlDbType.Int)).Direction =
ParameterDirection.Output
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombre", SqlDbType.NVarChar)).Value = NombreTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("password", SqlDbType.NVarChar)).Value =
PasswordTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nivel", SqlDbType.Int)).Value = CombNIVEL.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cargo", SqlDbType.NVarChar)).Value = CombCARGO.Text
            COMANDO.SelectCommand.ExecuteNonQuery()

            Me.N°_Usuario.Text = "N° de Usuarios Registrados:" &
COMANDO.SelectCommand.Parameters(0).Value
            MsgBox("USUARIO registrado correctamente")
            Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)
            Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
            Me.UsuarioDataGridView.DataSource =
Me.UsuarioBindingSource
            BtnModificar.Enabled = True
            BtnEliminar.Enabled = True
            BtnNuevo.Enabled = True

        Else
            If llamado.Text = 1 Then
                'CADENA DE CONEXIÓN DE LA BSE DE DATOS
                'CONN = New SqlConnection("Data Source=COMANDATO-
VAIO\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
                CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")

                CONN.Open()
                COMANDO = New SqlDataAdapter("Modificar_USUARIO",
CONN)
                COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombre", SqlDbType.NVarChar)).Value = NombreTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("password", SqlDbType.NVarChar)).Value =
PasswordTextBox.Text
```

```
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nivel", SqlDbType.Int)).Value = CombNIVEL.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cargo", SqlDbType.NVarChar)).Value = CombCARGO.Text
                COMANDO.SelectCommand.ExecuteNonQuery()
                Me.UsuarioDataGridView.Update()
                Me.UsuarioDataGridView.Refresh()
                MsgBox("Los Datos fueron Modificados con exito")

Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)
                Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
                Me.UsuarioDataGridView.DataSource =
Me.UsuarioBindingSource
                Me.llamado.Text = 0
                BtnGuardar.Enabled = False
                Me.NombreTextBox.Enabled = False
                Me.PasswordTextBox.Enabled = False
                Me.CombNIVEL.Enabled = False
                Me.CombCARGO.Enabled = False
                Me.NombreTextBox.Text = ""
                Me.PasswordTextBox.Text = ""
                Me.CombNIVEL.Text = ""
                Me.CombCARGO.Text = ""
            End If
        End If
    End Sub

    Private Sub BtnBuscar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnBuscar.Click

        If (NombreTextBox.Text = "") Then
            MsgBox("Por favor ingrese USUARIO")
            NombreTextBox.Focus()
        Else
            Me.UsuarioDataGridView.DataSource = Nothing
            dw.Table = DS.Tables("Usuario")
            dw.RowFilter = "nombre='" & NombreTextBox.Text & "'"

            If dw.Count > 0 Then
                Me.UsuarioDataGridView.DataSource = dw
                Me.UsuarioDataGridView.Refresh()
                MsgBox("El USUARIO ya existe")
                BtnNuevo.Enabled = True
                NombreTextBox.Text = ""
                NombreTextBox.Focus()
            Else
                MsgBox("USUARIO NO Registrado ")
                PasswordTextBox.Focus()
                BtnGuardar.Enabled = True
            End If
        End If
    End Sub

    Private Sub BtnCancelar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnCancelar.Click
        Try
            NombreTextBox.Text = ""
            PasswordTextBox.Text = ""
            CombNIVEL.Text = ""
            CombCARGO.Text = ""
```

```
            NombreTextBox.Focus()
            BtnNuevo.Enabled = True
            BtnGuardar.Enabled = False
            BtnModificar.Enabled = False
            BtnEliminar.Enabled = False
            BtnCancelar.Enabled = False
            Me.NombreTextBox.Enabled = False
            Me.PasswordTextBox.Enabled = False
            Me.CombNIVEL.Enabled = False
            Me.CombCARGO.Enabled = False
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
    End Sub

    Private Sub BtnModificar_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnModificar.Click
        ''CADENA DE CONEXIÓN DE LA BASE DE DATOS
        'CONN = New SqlConnection("Data Source=COMANDATO-
VAIO\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")

        CONN.Open()
        COMANDO = New SqlDataAdapter("Modifi_Usua", CONN)
        COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombre", SqlDbType.NVarChar)).Value = NombreTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("password", SqlDbType.NVarChar)).Value =
PasswordTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New SqlParameter("nivel",
SqlDbType.Int)).Value = CombNIVEL.Text
        COMANDO.SelectCommand.Parameters.Add(New SqlParameter("cargo",
SqlDbType.NVarChar)).Value = CombCARGO.Text
        COMANDO.SelectCommand.ExecuteNonQuery()
        MsgBox("Los Datos fueron Modificados con exito")

        ''''Se actualiza la tabla de clientes
        Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
        Me.UsuarioDataGridView.DataSource = Me.UsuarioBindingSource
        Me.UsuarioDataGridView.Update()
        NombreTextBox.Focus()

    End Sub

    Private Sub BtnEliminar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnEliminar.Click

        Dim resp As Object
        resp = MsgBox("¿Realmente Desea Eliminar al USUARIO?",
vbYesNo, "Ferrecentro Oñate")
        If resp = vbYes Then
            ''CADENA DE CONEXIÓN DE LA BASE DE DATOS******
            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")

            CONN.Open()
```

```
            COMANDO = New SqlDataAdapter("eliminar_USUARIO", CONN)
            COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombre", SqlDbType.NVarChar)).Value = NombreTextBox.Text
            COMANDO.SelectCommand.ExecuteNonQuery()
            MsgBox("El Usuario fue eliminado")
            NombreTextBox.Focus()
            Me.UsuarioTableAdapter.Fill(Me.DatosSQLDataSet.Usuario)
            Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
            Me.UsuarioDataGridView.DataSource =
Me.UsuarioBindingSource
            Me.NombreTextBox.Focus()
        End If
    End Sub

    Private Sub BtnSalir_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnSalir.Click
        Me.NombreTextBox.Text = InputBox("Ingrese el Nombre de
USUARIO: ", "Ferreteria Oñate S.A.")
        If (NombreTextBox.Text = "") Then
            MsgBox("Por favor ingrese Nombre de Usuario")
            NombreTextBox.Focus()
        Else
            Me.UsuarioDataGridView.DataSource = Nothing
            dw.Table = DS.Tables("Usuario")
            dw.RowFilter = "nombre='" & NombreTextBox.Text & "'"

            If dw.Count > 0 Then
                Me.UsuarioDataGridView.DataSource = dw
                Me.UsuarioDataGridView.Refresh()
                Me.NombreTextBox.Text = Me.UsuarioDataGridView.Item(0,
0).Value
                Me.PasswordTextBox.Text =
Me.UsuarioDataGridView.Item(1, 0).Value
                Me.CombNIVEL.Text = Me.UsuarioDataGridView.Item(2,
0).Value
                Me.CombCARGO.Text = Me.UsuarioDataGridView.Item(3,
0).Value
                BtnModificar.Enabled = True
            Else
                MsgBox("USUARIO NO Registrado ")
                BtnGuardar.Enabled = True
            End If
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Me.Close()
    End Sub

    Private Sub NombreTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NombreTextBox.TextChanged
        Me.NombreTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub
```

```
    Private Sub CombNIVEL_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CombNIVEL.SelectedIndexChanged
        Select Case CombNIVEL.Text

            Case "1"
                CombCARGO.Items.Clear()
                CombCARGO.Items.Add("GERENTE(A)")
                CombCARGO.Items.Add("SUBGERENTE(A)")
                CombCARGO.Items.Add("ADMINISTRADOR(A)")
            Case "2"
                CombCARGO.Items.Clear()
                CombCARGO.Items.Add("EMPLEADO(A)")
                CombCARGO.Items.Add("CAJERO(A)")
        End Select

    End Sub


End Class
```

FORMULARIO CLIENTES

```
Imports System.Data
Imports System.Data.SqlClient
Public Class Clientes
    Dim CONN As SqlConnection
    Dim COMANDO As SqlDataAdapter
    Dim DS As New DataSet
    Dim dw As New DataView

    Private Sub ClientesBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.ClientesBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Clientes' Puede moverla o quitarla según sea
necesario.
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
        ''********************CONEXIÒN A LA BASE DE
DATOS*************************************

        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")

        CONN.Open()
        COMANDO = New SqlDataAdapter("select * from Clientes", CONN)
        COMANDO.Fill(DS, "Clientes")
        Me.ClientesDataGridView.DataSource = DS.Tables("Clientes")
        Try
            btnNuevo.Enabled = True
            btnGuardar.Enabled = False
            cancelar.Enabled = False
```

```
            BtnBusca.Enabled = False

            Me.NombresTextBox.Enabled = False
            Me.ApellidosTextBox.Enabled = False
            Me.Combciudad.Enabled = False
            Me.DireccionTextBox.Enabled = False
            Me.TelefonoTextBox.Enabled = False
            Me.Combsexo.Enabled = False
            Me.Fecha_nac.Enabled = False
            Me.ClientesTableAdapter.Fill(Me.DatosSQLDataSet.Clientes)
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
    End Sub
    Private Sub btnNuevo_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnNuevo.Click
        Try
            CedulaTextBox.BackColor = Color.Yellow
            CedulaTextBox.ForeColor = Color.Red
            Me.CedulaTextBox.Text = ""
            Me.NombresTextBox.Text = ""
            Me.ApellidosTextBox.Text = ""
            Me.Combciudad.Text = ""
            Me.DireccionTextBox.Text = ""
            Me.TelefonoTextBox.Text = ""
            Me.Combsexo.Text = ""
            Me.Fecha_nac.Value = System.DateTime.Today

            btnGuardar.Enabled = False
            cancelar.Enabled = True
            BtnBusca.Enabled = True
            Me.NombresTextBox.Enabled = True
            Me.ApellidosTextBox.Enabled = True
            Me.Combciudad.Enabled = True
            Me.DireccionTextBox.Enabled = True
            Me.TelefonoTextBox.Enabled = True
            Me.Combsexo.Enabled = True
            Me.Fecha_nac.Enabled = True
            Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
            Me.CedulaTextBox.Focus()
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try


    End Sub

    Private Sub btnGuardar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnGuardar.Click

        Me.ClientesDataGridView.DataSource = Nothing
        dw.Table = DS.Tables("Clientes")
        dw.RowFilter = "cedula='" & CedulaTextBox.Text & "'"

        If dw.Count > 0 Then
            Me.ClientesDataGridView.DataSource = dw
            Me.ClientesDataGridView.Refresh()
            MsgBox("el cliente ya existe")
            CedulaTextBox.Text = ""
            CedulaTextBox.Focus()
```

```
        Else
            NombresTextBox.Focus()
''***VERIFICA SI LA CEDULA ES CORRECTA*********
            Dim MD As New negocio
            Dim cedula12 As String
            cedula12 = CedulaTextBox.Text
            ''***VERIFICA SI LA CEDULA ES CORRECTA***
            If MD.VerificaCedula(cedula12) = False Then
                btnEliminar.Enabled = False
                BtnModificar.Enabled = False
                CedulaTextBox.Text = ""
                CedulaTextBox.Focus()
            Else
                '***CONEXIÓN A LA BASE DE DATOS*******

                CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
                CONN.Open()
                COMANDO = New SqlDataAdapter("insertar", CONN)
                COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cedula", SqlDbType.NVarChar)).Value = CedulaTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombres", SqlDbType.NVarChar)).Value =
NombresTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("apellidos", SqlDbType.NVarChar)).Value =
ApellidosTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("ciudad", SqlDbType.NVarChar)).Value = Combciudad.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("direccion", SqlDbType.NVarChar)).Value =
DireccionTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("telefono", SqlDbType.NVarChar)).Value =
TelefonoTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("sexo", SqlDbType.NVarChar)).Value = Combsexo.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("fecha_nac", SqlDbType.DateTime)).Value = Fecha_nac.Value
                COMANDO.SelectCommand.ExecuteNonQuery()
                MsgBox("Cliente registrado correctamente")

Me.ClientesTableAdapter.Fill(Me.DatosSQLDataSet.Clientes)
                Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
                Me.ClientesDataGridView.DataSource =
Me.ClientesBindingSource
                BtnModificar.Enabled = True
                btnEliminar.Enabled = True
                btnNuevo.Enabled = True
            End If
        End If
    End Sub

    Private Sub btnEliminar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnEliminar.Click
        'Mensaje si queremos eliminar el campo
        If MessageBox.Show("¿Esta seguro de Eliminar el Cliente?",
"Responda", _
```

```
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) =
Windows.Forms.DialogResult.Yes Then
            'creamos cadena de conexion
            '****CONEXIÓN A LA BASE DE DATOS*********
            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            CONN.Open()
            'realizamos la operacion SQL
            Dim ELIMINAR As New SqlClient.SqlCommand()
            ELIMINAR.CommandType = System.Data.CommandType.Text
            ELIMINAR.CommandText = "Delete From Clientes Where cedula
= '" & CedulaTextBox.Text & "'"
            ELIMINAR.Connection = CONN
            'comprobamos si se elimino los datos
            Try
                If ((ELIMINAR.ExecuteNonQuery <> 0)) Then
                    MsgBox("Cliente Eliminado")

Me.ClientesTableAdapter.Fill(Me.DatosSQLDataSet.Clientes)
                    Me.ClientesDataGridView.DataSource =
Me.ClientesBindingSource
                End If
            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try
        End If
    End Sub

    Private Sub BtnModificar_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnModificar.Click
        '***CONEXIÓN A LA BASE DE DATOS************

        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        CONN.Open()
        COMANDO = New SqlDataAdapter("Modificar1", CONN)
        COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cedula", SqlDbType.NVarChar)).Value = CedulaTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombres", SqlDbType.NVarChar)).Value =
NombresTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("apellidos", SqlDbType.NVarChar)).Value =
ApellidosTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("ciudad", SqlDbType.NVarChar)).Value = Combciudad.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("direccion", SqlDbType.NVarChar)).Value =
DireccionTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("telefono", SqlDbType.NVarChar)).Value =
TelefonoTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New SqlParameter("sexo",
SqlDbType.NVarChar)).Value = Combsexo.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("fecha_nac", SqlDbType.DateTime)).Value = Fecha_nac.Value
        COMANDO.SelectCommand.ExecuteNonQuery()
        MsgBox("Los Datos fueron Modificados con exito")
```

```
        ''''Se actualiza la tabla de clientes
        Me.ClientesTableAdapter.Fill(Me.DatosSQLDataSet.Clientes)
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
        Me.ClientesDataGridView.DataSource = Me.ClientesBindingSource
        Me.ClientesDataGridView.Update()
        CedulaTextBox.Focus()


    End Sub

    Private Sub primero_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles primero.Click
        Me.ClientesBindingSource.MoveFirst()

    End Sub

    Private Sub anterior_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles anterior.Click
        Me.ClientesBindingSource.MovePrevious()
    End Sub

    Private Sub siguiente_Click_1(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles siguiente.Click
        Me.ClientesBindingSource.MoveNext()
    End Sub

    Private Sub ultimo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ultimo.Click
        Me.ClientesBindingSource.MoveLast()
    End Sub

    Private Sub cerrar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cerrar.Click
        Me.Close()
    End Sub

    Private Sub cancelar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cancelar.Click
        Try
            CedulaTextBox.Text = ""
            NombresTextBox.Text = ""
            ApellidosTextBox.Text = ""
            Combciudad.Text = ""
            DireccionTextBox.Text = ""
            TelefonoTextBox.Text = ""
            Combsexo.Text = ""
            CedulaTextBox.Focus()
            btnNuevo.Enabled = True
            btnGuardar.Enabled = False
            BtnModificar.Enabled = False
            btnEliminar.Enabled = False
            cancelar.Enabled = False
            Me.NombresTextBox.Enabled = False
            Me.ApellidosTextBox.Enabled = False
            Me.Combciudad.Enabled = False
            Me.DireccionTextBox.Enabled = False
            Me.TelefonoTextBox.Enabled = False
            Me.Combsexo.Enabled = False
            Me.Fecha_nac.Enabled = False
```

```
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
    End Sub

    Private Sub CedulaTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CedulaTextBox.TextChanged
        Me.CedulaTextBox.MaxLength = 10
    End Sub

    Private Sub BtnBusca_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnBusca.Click

        If (CedulaTextBox.Text = "") Then
            MsgBox("Por favor ingrese N° de cedula")
            CedulaTextBox.Focus()
        Else
            Me.ClientesDataGridView.DataSource = Nothing
            dw.Table = DS.Tables("Clientes")
            dw.RowFilter = "cedula='" & CedulaTextBox.Text & "'"

            If dw.Count > 0 Then
                Me.ClientesDataGridView.DataSource = dw
                Me.ClientesDataGridView.Refresh()
                MsgBox("el cliente ya existe")

                CedulaTextBox.Text = ""
                CedulaTextBox.Focus()
            Else
                MsgBox("Cliente NO Registrado ")
                NombresTextBox.Focus()
                btnGuardar.Enabled = True
            End If
        End If
    End Sub

    Private Sub NombresTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NombresTextBox.TextChanged
        Me.NombresTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Me.ClientesDataGridView.DataSource = Nothing
        dw.Table = DS.Tables("Clientes")
        dw.RowFilter = "cedula='" & CedulaTextBox.Text & "'"
        If dw.Count > 0 Then
            Me.ClientesDataGridView.DataSource = dw
            Me.ClientesDataGridView.Refresh()
            Me.CedulaTextBox.Text = Me.ClientesDataGridView.Item(0,
0).Value
            Me.NombresTextBox.Text = Me.ClientesDataGridView.Item(1,
0).Value
            Me.ApellidosTextBox.Text = Me.ClientesDataGridView.Item(2,
0).Value
            Me.Combciudad.Text = Me.ClientesDataGridView.Item(3,
0).Value
```

```
            Me.DireccionTextBox.Text = Me.ClientesDataGridView.Item(4,
0).Value
            Me.TelefonoTextBox.Text = Me.ClientesDataGridView.Item(5,
0).Value
            Me.Combsexo.Text = Me.ClientesDataGridView.Item(6,
0).Value
            Me.Fecha_nac.Text = Me.ClientesDataGridView.Item(7,
0).Value
        Else
            MsgBox("El Cliente No Existe")
            CedulaTextBox.Text = ""
            CedulaTextBox.Focus()
        End If
    End Sub

    Private Sub ApellidosTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ApellidosTextBox.TextChanged
        Me.ApellidosTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub DireccionTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
DireccionTextBox.TextChanged
        Me.DireccionTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub TelefonoTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TelefonoTextBox.TextChanged
        Me.TelefonoTextBox.MaxLength = 10
    End Sub


    Private Sub CedulaLabel_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)

    End Sub
End Class.
```

FORMULARIO PROVEEDORES

```
Imports System.Data
Imports System.Data.SqlClient
Public Class Proveedores
    Dim CONN As SqlConnection
    Dim COMANDO As SqlDataAdapter
    Dim DS As New DataSet
    Dim dw As New DataView

    Private Sub ProveedoresBindingNavigatorSaveItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.ProveedoresBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

    End Sub
```

```
    Private Sub Proveedores_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Clientes' Puede moverla o quitarla según sea
necesario.
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
        ''*******CADENA DE CONEXIÓN A LA BASE DE
DATOS******************
        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")


        Try
            CONN.Open()
            COMANDO = New SqlDataAdapter("select * from Proveedores",
CONN)
            COMANDO.Fill(DS, "Proveedores")
            Me.ProveedoresDataGridView.DataSource =
DS.Tables("Proveedores")
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
        Try

            btnNuevo.Enabled = True
            btnGuardar.Enabled = False
           cancelar.Enabled = False
            BtnBuscar.Enabled = False

            Me.NombresTextBox.Enabled = False
            Me.ApellidosTextBox.Enabled = False
            Me.Combprovincia.Enabled = False
            Me.Combciudad.Enabled = False
            Me.DireccionTextBox.Enabled = False
            Me.TelefonoTextBox.Enabled = False

Me.ProveedoresTableAdapter.Fill(Me.DatosSQLDataSet.Proveedores)
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try

    End Sub

    Private Sub btnNuevo_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnNuevo.Click
        Try
            CedulaTextBox.BackColor = Color.Yellow
            CedulaTextBox.ForeColor = Color.Red
            Me.CedulaTextBox.Text = ""
            Me.NombresTextBox.Text = ""
            Me.ApellidosTextBox.Text = ""
            Me.Combprovincia.Text = ""
            Me.Combciudad.Text = ""
            Me.DireccionTextBox.Text = ""
            Me.TelefonoTextBox.Text = ""
            btnGuardar.Enabled = False
            cancelar.Enabled = True
            BtnBuscar.Enabled = True
            Me.NombresTextBox.Enabled = True
            Me.ApellidosTextBox.Enabled = True
```

```
            Me.Combprovincia.Enabled = True
            Me.Combciudad.Enabled = True
            Me.DireccionTextBox.Enabled = True
            Me.TelefonoTextBox.Enabled = True
            Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
            Me.CedulaTextBox.Focus()
        Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
    End Sub

    Private Sub btnGuardar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnGuardar.Click

        Me.ProveedoresDataGridView.DataSource = Nothing
        dw.Table = DS.Tables("Proveedores")
        dw.RowFilter = "cedula='" & CedulaTextBox.Text & "'"

        If dw.Count > 0 Then
            Me.ProveedoresDataGridView.DataSource = dw
            Me.ProveedoresDataGridView.Refresh()
            MsgBox("el Proveedor ya existe")
            CedulaTextBox.Text = ""
            CedulaTextBox.Focus()
        Else

            NombresTextBox.Focus()
            'VERIFICA EL NÚMERO DE CEDULA**********
            Dim MD As New negocio
            Dim cedula12 As String
            cedula12 = CedulaTextBox.Text
            ''***VERIFICA SI LA CEDULA ES CORRECTA***
            If MD.VerificaCedula(cedula12) = False Then

                btnEliminar.Enabled = False
                BtnModificar.Enabled = False
                CedulaTextBox.Text = ""
                CedulaTextBox.Focus()
            Else
                ''***************CADENA DE CONEXIÓN*********
                CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
                CONN.Open()
                COMANDO = New SqlDataAdapter("insertar provedo", CONN)
                COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cedula", SqlDbType.NVarChar)).Value = CedulaTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombres", SqlDbType.NVarChar)).Value =
NombresTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("apellidos", SqlDbType.NVarChar)).Value =
ApellidosTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("provincia", SqlDbType.NVarChar)).Value =
Combprovincia.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("ciudad", SqlDbType.NVarChar)).Value = Combciudad.Text
```

```
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("direccion", SqlDbType.NVarChar)).Value =
DireccionTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("telefono", SqlDbType.NVarChar)).Value =
TelefonoTextBox.Text
                COMANDO.SelectCommand.ExecuteNonQuery()
                MsgBox("Proveedor registrado correctamente")

Me.ProveedoresTableAdapter.Fill(Me.DatosSQLDataSet.Proveedores)
                Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
                Me.ProveedoresDataGridView.DataSource =
Me.ProveedoresBindingSource
                BtnModificar.Enabled = True
                btnEliminar.Enabled = True
                btnNuevo.Enabled = True
            End If
        End If
    End Sub

    Private Sub cerrar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cerrar.Click
        Me.Close()
    End Sub

    Private Sub cancelar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cancelar.Click
        Try
            CedulaTextBox.Text = ""
            NombresTextBox.Text = ""
            ApellidosTextBox.Text = ""
            Combprovincia.Text = ""
            Combciudad.Text = ""
            DireccionTextBox.Text = ""
            TelefonoTextBox.Text = ""

            CedulaTextBox.Focus()
            btnNuevo.Enabled = True
            btnGuardar.Enabled = False
            BtnModificar.Enabled = False
            btnEliminar.Enabled = False
            cancelar.Enabled = False
            Me.NombresTextBox.Enabled = False
            Me.ApellidosTextBox.Enabled = False
            Me.Combprovincia.Enabled = False
            Me.Combciudad.Enabled = False
            Me.DireccionTextBox.Enabled = False
            Me.TelefonoTextBox.Enabled = False
            Catch err As Exception
            MsgBox("ERROR:" & err.Message, MsgBoxStyle.Critical)
        End Try
    End Sub

    Private Sub BtnBuscar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnBuscar.Click
        Try
            If (CedulaTextBox.Text = "") Then
                MsgBox("Por favor ingrese N° de cedula")
                CedulaTextBox.Focus()
            Else
```

```
                'Crear el Adaptador
                Dim COMANDO As New SqlDataAdapter("buscar_proveedor1'"
& CedulaTextBox.Text & "'", CONN)
                'crear conjunto de datos
                Dim ds As New DataSet
                'Utilizar el DataAdapter para llenar el Dataset con
una tabla
                COMANDO.Fill(ds, "Proveedores")
                ProveedoresDataGridView.DataSource =
ds.Tables("Proveedores")
                If CedulaTextBox.Text =
ProveedoresDataGridView.Item(0, 0).Value Then
                    MsgBox("el proveedor ya existe")
                    CedulaTextBox.Text = ""
                    CedulaTextBox.Focus()
                Else
                    MsgBox("Proveedor NO Registrado ")
                    NombresTextBox.Focus()
                    btnGuardar.Enabled = True
                End If
            End If
        Catch ex As SqlException
            MessageBox.Show(ex.Message)
        End Try
    End Sub

    Private Sub btnEliminar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnEliminar.Click
        'Mensaje si queremos eliminar el campo
        If MessageBox.Show("¿Esta seguro de Eliminar el Proveedor?",
"Responda", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) =
Windows.Forms.DialogResult.Yes Then
            'creamos cadena de conexion
            ''''****CADENA DE CONEXIÓN A LA BASE DE DATOS
            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            CONN.Open()
            'realizamos la operacion SQL
            Dim ELIMINAR As New SqlClient.SqlCommand()
            ELIMINAR.CommandType = System.Data.CommandType.Text
            ELIMINAR.CommandText = "Delete From Proveedores Where
cedula = '" & CedulaTextBox.Text & "'"
            ELIMINAR.Connection = CONN
            'comprobamos si se elimino los datos
            Try
                If ((ELIMINAR.ExecuteNonQuery <> 0)) Then
                    MsgBox("Cliente Eliminado")

Me.ProveedoresTableAdapter.Fill(Me.DatosSQLDataSet.Proveedores)
                    Me.ProveedoresDataGridView.DataSource =
Me.ProveedoresBindingSource
                End If
            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try
        End If
    End Sub
```

```
    Private Sub BtnModificar_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnModificar.Click
        '***CADENA DE CONEXIÓN A LA BASE DE DATOS******
        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        CONN.Open()
        COMANDO = New SqlDataAdapter("Modificar_pro", CONN)
        COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cedula", SqlDbType.NVarChar)).Value = CedulaTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombres", SqlDbType.NVarChar)).Value =
NombresTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("apellidos", SqlDbType.NVarChar)).Value =
ApellidosTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("provincia", SqlDbType.NVarChar)).Value =
Combprovincia.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("ciudad", SqlDbType.NVarChar)).Value = Combciudad.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("direccion", SqlDbType.NVarChar)).Value =
DireccionTextBox.Text
        COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("telefono", SqlDbType.NVarChar)).Value =
TelefonoTextBox.Text
       COMANDO.SelectCommand.ExecuteNonQuery()
        MsgBox("Los Datos fueron Modificados con exito")

Me.ProveedoresTableAdapter.Fill(Me.DatosSQLDataSet.Proveedores)
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
        Me.ProveedoresDataGridView.DataSource =
Me.ProveedoresBindingSource
        Me.ProveedoresDataGridView.Update()
        CedulaTextBox.Focus()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
        If Inventario.llama.Text = 1 Then

            Inventario.CedulaproTextBox.Text = CedulaTextBox.Text
            Inventario.nombrepro.Text = NombresTextBox.Text
            Me.Hide()
        Else
            If Compra.llama.Text = 2 Then
                Compra.CedulaTextBox.Text = CedulaTextBox.Text
                Compra.NombresTextBox.Text = NombresTextBox.Text
                Compra.ApellidosTextBox.Text = ApellidosTextBox.Text
                Compra.CiudadTextBox.Text = Combciudad.Text
                Compra.DireccionTextBox.Text = DireccionTextBox.Text
                Compra.TelefonoTextBox.Text = TelefonoTextBox.Text
            End If
        End If
    End Sub
```

```
    Private Sub BtnBuscarp_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnBuscarp.Click
        Me.ProveedoresDataGridView.DataSource = Nothing
        dw.Table = DS.Tables("Proveedores")
        dw.RowFilter = "cedula='" & CedulaTextBox.Text & "'"

        If dw.Count > 0 Then
            Me.ProveedoresDataGridView.DataSource = dw
            Me.ProveedoresDataGridView.Refresh()
            Me.CedulaTextBox.Text = Me.ProveedoresDataGridView.Item(0,
0).Value
            Me.NombresTextBox.Text =
Me.ProveedoresDataGridView.Item(1, 0).Value
            Me.ApellidosTextBox.Text =
Me.ProveedoresDataGridView.Item(2, 0).Value
            Me.Combprovincia.Text = Me.ProveedoresDataGridView.Item(3,
0).Value
            Me.Combciudad.Text = Me.ProveedoresDataGridView.Item(4,
0).Value
            Me.DireccionTextBox.Text =
Me.ProveedoresDataGridView.Item(5, 0).Value
            Me.TelefonoTextBox.Text =
Me.ProveedoresDataGridView.Item(6, 0).Value
        Else
            MsgBox("El Proveedor no esta Registrado")
            CedulaTextBox.Text = ""
            CedulaTextBox.Focus()
        End If
    End Sub

    Private Sub TelefonoTextBox_KeyPress(ByVal sender As Object, ByVal
e As System.Windows.Forms.KeyPressEventArgs) Handles
TelefonoTextBox.KeyPress
        If Char.IsDigit(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsControl(e.KeyChar) Then
            e.Handled = False
        Else
            e.Handled = True
        End If

    End Sub

    Private Sub TelefonoTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TelefonoTextBox.TextChanged
        Me.TelefonoTextBox.MaxLength = 10
    End Sub

    Private Sub NombresTextBox_KeyPress(ByVal sender As Object, ByVal
e As System.Windows.Forms.KeyPressEventArgs) Handles
NombresTextBox.KeyPress
        If Char.IsLetter(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsControl(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsSeparator(e.KeyChar) Then
            e.Handled = False
        Else
            e.Handled = True
```

```
        End If

    End Sub

    Private Sub NombresTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NombresTextBox.TextChanged
        Me.NombresTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub ApellidosTextBox_KeyPress(ByVal sender As Object,
ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles
ApellidosTextBox.KeyPress
        If Char.IsLetter(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsControl(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsSeparator(e.KeyChar) Then
            e.Handled = False
        Else
            e.Handled = True
        End If

    End Sub

    Private Sub ApellidosTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ApellidosTextBox.TextChanged
        Me.ApellidosTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub DireccionTextBox_KeyPress(ByVal sender As Object,
ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles
DireccionTextBox.KeyPress
        If Char.IsLetter(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsControl(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsSeparator(e.KeyChar) Then
            e.Handled = False
        Else
            e.Handled = True
        End If

    End Sub

    Private Sub DireccionTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
DireccionTextBox.TextChanged
        Me.DireccionTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button4.Click
        Me.ProveedoresBindingSource.MoveFirst()
    End Sub

    Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        Me.ProveedoresBindingSource.MovePrevious()
```

```
    End Sub

    Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        Me.ProveedoresBindingSource.MoveNext()
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button3.Click
        Me.ProveedoresBindingSource.MoveLast()
    End Sub

    Private Sub CedulaTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CedulaTextBox.TextChanged
        Me.CedulaTextBox.MaxLength = 10
    End Sub

    Private Sub Combprovincia_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Combprovincia.SelectedIndexChanged
        Select Case Combprovincia.SelectedIndex

            Case "0"
                Combciudad.Items.Clear()
                Combciudad.Items.Add("BABAHOYO")
                Combciudad.Items.Add("BABA")
                Combciudad.Items.Add("MONTALVO")
                Combciudad.Items.Add("QUEVEDO")
                Combciudad.Items.Add("VENTANAS")
                Combciudad.Items.Add("VINCES")
                Combciudad.Items.Add("BUENA FÉ")
                Combciudad.Items.Add("VALENCIA")
                Combciudad.Items.Add("MOCACHE")

            Case "1"
                Combciudad.Items.Clear()
                Combciudad.Items.Add("GUAYAQUIL")
                Combciudad.Items.Add("DURAN")
                Combciudad.Items.Add("MILAGRO")
                Combciudad.Items.Add("YAGUACHI")
                Combciudad.Items.Add("DAULE")
                Combciudad.Items.Add("PLAYAS")
                Combciudad.Items.Add("JUJAN")
                Combciudad.Items.Add("BALZAR")
                Combciudad.Items.Add("SIMÓN BOLÍVAR")
                Combciudad.Items.Add("NOBOL")
                Combciudad.Items.Add("LA LIBERTAD")
        End Select
    End Sub


    Private Sub Combciudad_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Combciudad.SelectedIndexChanged

    End Sub
End Class
```

FORMULARIO FACTURA

```
Option Explicit On
Option Strict On

Imports System.Data
Imports System.Data.SqlClient

Public Class Factura
    '''***CADENA DE CONEXIÓN A LA BASE DE DATOS******
    Public CONN As New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
    Dim COMANDO As SqlDataAdapter
    Dim DS As New DataSet
    Dim dw As New DataView
    Dim NUM As Integer = 0
    Dim filas As Integer = 0
    Dim e As Integer = 6

    Dim w As Integer = 0
    Dim x As Integer = 0
    Dim y As Integer = 0
    Dim z As Integer = 0

    Private WithEvents bs As New BindingSource
    Private bEdit As Boolean

    Sub CargarDatos()
        Dim daCategories As New SqlDataAdapter("Select * From
detFactura", CONN)
        DS.Clear()
        daCategories.Fill(DS, "detFactura")
        Me.DetFacturaDataGridView.DataSource = DS
        Me.DetFacturaDataGridView.DataMember = "detFactura"
    End Sub
    Private Sub ClientesBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.ClientesBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
    End Sub


    Private Sub Factura_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.caja' Puede moverla o quitarla según sea necesario.
        Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Factura' Puede moverla o quitarla según sea
necesario.
        Me.FacturaTableAdapter.Fill(Me.DatosSQLDataSet.Factura)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Inventario' Puede moverla o quitarla según sea
necesario.
        Me.InventarioTableAdapter.Fill(Me.DatosSQLDataSet.Inventario)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.detFactura' Puede moverla o quitarla según sea
necesario.
```

```
        Me.DetFacturaTableAdapter.Fill(Me.DatosSQLDataSet.detFactura)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Clientes' Puede moverla o quitarla según sea
necesario.
        Me.ClientesTableAdapter.Fill(Me.DatosSQLDataSet.Clientes)
        Me.lleva.Text = CStr(0)
        Me.filita.Text = CStr(0)
        limpiar_valores()

        ''***CADENA DE CONEXIÓN DE LA BASE ED DATOS*******
        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        COMANDO = New SqlDataAdapter("select * from Clientes", CONN)
        DS = New DataSet
        COMANDO.Fill(DS, "Clientes")
        Me.ClientesDataGridView.DataSource = DS.Tables("Clientes")


        'CONN = New SqlConnection("Data Source=ZULEYKA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        'CONN = New SqlConnection("Data Source=COMANDATO-
VAIO\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        COMANDO = New SqlDataAdapter("select * from Inventario", CONN)
        DS = New DataSet
        COMANDO.Fill(DS, "Inventario")
        'Me.DetFacturaDataGridView.DataSource = Nothing
        Me.DetFacturaDataGridView.Refresh()
        Me.valorFila.Text = CStr(0)

        Me.limpiar_cajas()
        Me.deshabilitar()
        Me.deshabilitar_botones()
        Me.deshabilitar_comandos()


        'Me.descuento.Text =
FormatCurrency(CStr(CDbl(Me.SUBTOTAL.Text) * CDbl(CDbl(ComboDESC.Text)
/ 100)), 2)
        'Me.subt_descuento.Text =
FormatCurrency(CStr(CDbl(Me.SUBTOTAL.Text) - CDbl(Me.descuento.Text)),
2)
        'Me.IVA.Text =
FormatCurrency(CStr(CDbl(Me.subt_descuento.Text) * 0.12), 2)
        'Me.TOTAL_P.Text =
FormatCurrency(CStr(CDbl(Me.subt_descuento.Text) + CDbl(Me.IVA.Text)),
2)
End Sub


    Private Sub Btn_NuevoItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnNuevoItem.Click

        Lista_ProductoFact.Show()
        Lista_ProductoFact.lbDespachar.Visible = True
        lleva.Text = CStr(1)
        Lista_ProductoFact.n°factura.Text = labelNroCompra.Text
        Lista_ProductoFact.CodigoTextBox.Text = ""
        Lista_ProductoFact.NombreTextBox.Text = ""
        Lista_ProductoFact.StockTextBox.Text = ""
        Lista_ProductoFact.costo.Text = ""
```

```
    End Sub

    Private Sub BtnModificarItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnModificarItem.Click
        If CDbl(filita.Text) >= 0 Then
            Lista_ProductoFact.Show()
            Lista_ProductoFact.lbmodificar.Visible = True
            Lista_ProductoFact.buscarpro()
            lleva.Text = CStr(2)
            Lista_ProductoFact.Button1.Enabled = True
        Else
            MsgBox("Debe seleccionar el pedido en la Grilla")
        End If
    End Sub

    Private Sub DetFacturaDataGridView_RowHeaderMouseClick(ByVal
sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellMouseEventArgs) Handles
DetFacturaDataGridView.RowHeaderMouseClick
        If e.RowIndex >= 0 Then
            filita.Text = CStr(e.RowIndex)
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnEliminarItem.Click
        If CDbl(filita.Text) >= 0 Then
            Lista_ProductoFact.Show()
            Lista_ProductoFact.lbeliminar.Visible = True
            Lista_ProductoFact.buscarpro()
            lleva.Text = CStr(3)
            Lista_ProductoFact.StockTextBox1.Visible = True
            Lista_ProductoFact.StockTextBox1.Text =
Lista_ProductoFact.StockTextBox.Text
            Lista_ProductoFact.Button1.Enabled = True
            Lista_ProductoFact.Button2.Enabled = False

        Else
            MsgBox("Debe seleccionar el pedido en la Grilla")
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnNUEVO.Click

        limpiar_valores()
        Me.DetFacturaTableAdapter.Fill(Me.DatosSQLDataSet.detFactura)
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
        Me.BtnBUSCARcli.Enabled = True
        Me.habilitartexto()
        Me.habilitar_botones()
        Me.Labelfecha.Text = CStr(Date.Today)
        Me.DetFacturaBindingSource.MoveLast()
        'NUM = 1
        NUM = CInt(CDbl(NroFacturaTextBox.Text) + 1)
        labelNroCompra.Text = "00000" & CStr(CInt(NUM))
        'labelNroCompra.Text = CStr(NUM)
        BtnGUARDAR.Enabled = True
        limpiar_cajas()
```

```
    End Sub

    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnBUSCARcli.Click
        Busq_Cliente.Show()
    End Sub
    Public Sub deshabilitar()
        Me.CedulaTextBox.Enabled = False
        Me.NombresTextBox.Enabled = False
        Me.ApellidosTextBox.Enabled = False
        Me.DireccionTextBox.Enabled = False
        Me.CiudadTextBox.Enabled = False
        Me.TelefonoTextBox.Enabled = False
    End Sub
    Public Sub habilitartexto()
        Me.CedulaTextBox.Enabled = True
        Me.NombresTextBox.Enabled = True
        Me.ApellidosTextBox.Enabled = True
        Me.DireccionTextBox.Enabled = True
        Me.CiudadTextBox.Enabled = True
        Me.TelefonoTextBox.Enabled = True
    End Sub
    Public Sub deshabilitar_botones()
        Me.BtnNuevoItem.Enabled = False
        Me.BtnModificarItem.Enabled = False
        Me.BtnEliminarItem.Enabled = False
    End Sub
    Public Sub habilitar_botones()
        Me.BtnNuevoItem.Enabled = True
        Me.BtnModificarItem.Enabled = True
        Me.BtnEliminarItem.Enabled = True
    End Sub
    Public Sub limpiar_cajas()
        Me.CedulaTextBox.Text = ""
        Me.NombresTextBox.Text = ""
        Me.ApellidosTextBox.Text = ""
        Me.CiudadTextBox.Text = ""
        Me.DireccionTextBox.Text = ""
        Me.TelefonoTextBox.Text = ""
        Me.ComboDESC.Text = CStr(0)
    End Sub
    Public Sub deshabilitar_comandos()
        Me.BtnGUARDAR.Enabled = False
        Me.BtnCANCELAR.Enabled = True
        Me.BtnIMPRIMIR.Enabled = True
        Me.BtnBUSCARcli.Enabled = False
    End Sub
    Public Sub limpiar_valores()
        Me.SUBTOTAL.Text = "$ 0.00"
        Me.descuento.Text = "$ 0.00"
        Me.subt_descuento.Text = "$ 0.00"
        Me.IVA.Text = "$ 0.00"
        Me.TOTAL_P.Text = "$ 0.00"
    End Sub
    Public Sub AGREGAR()
        Dim rdInforme As New Report_FACTURA2
        Form_FACTURA.CrystalReportViewer1.ReportSource = rdInforme
        rdInforme.DataDefinition.FormulaFields.Item(5).Text = "'" &
CStr(Me.DetFacturaDataGridView.Rows(0).Cells(2).Value) & "'"
```

```
        rdInforme.DataDefinition.FormulaFields.Item(6).Text = "'" &
CStr(Me.DetFacturaDataGridView.Rows(0).Cells(3).Value) & "'"
        rdInforme.DataDefinition.FormulaFields.Item(7).Text = "'" &
CStr(Me.DetFacturaDataGridView.Rows(0).Cells(4).Value) & "'"
        rdInforme.DataDefinition.FormulaFields.Item(8).Text = "'" &
CStr(Me.DetFacturaDataGridView.Rows(0).Cells(5).Value) & "'"
        Form_FACTURA.Show()
    End Sub

    Private Sub BtnEFECTIVO_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)
        Efectivo.Show()
        Efectivo.Pago_efe.Focus()
    End Sub

    Private Sub BtnGUARDAR_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnGUARDAR.Click

        If DetFacturaDataGridView.Rows.Count > 0 Then
            'CADENA DE CONEXIÓN DE LA BASE DE DATOS*****
            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            CONN.Open()
            Try
                For i As Integer = 0 To
DetFacturaDataGridView.Rows.Count - 1
                    COMANDO = New
SqlDataAdapter("insertar_detFactura", CONN)
                    COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@nroFactura", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(0).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@codigo", SqlDbType.NVarChar)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(1).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cantidad", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(2).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@detalle", SqlDbType.NVarChar)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(3).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@pUnitario", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(4).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(5).Value
                    COMANDO.SelectCommand.ExecuteNonQuery()

                    COMANDO = New SqlDataAdapter("ordenarnrofactu",
CONN)
                    COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure


                    COMANDO = New SqlDataAdapter("insertar_KARDEX1",
CONN)
                    COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
```

```
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@codigoArt", SqlDbType.NVarChar)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(1).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@fecha", SqlDbType.DateTime)).Value = Me.Labelfecha.Text
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@detalle", SqlDbType.NVarChar)).Value = "Venta S/F No."
& CStr(Me.DetFacturaDataGridView.Rows(i).Cells(0).Value)
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cantidadIn", SqlDbType.Int)).Value = 0
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@punitarioIn", SqlDbType.Decimal)).Value = 0
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@totalIn", SqlDbType.Decimal)).Value = 0
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cantidadSa", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(2).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@punitarioSa", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(4).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@totalSa", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(5).Value
                    COMANDO.SelectCommand.ExecuteNonQuery()
                Next
            Catch ex As Exception
                MessageBox.Show(ex.Message)
            End Try

            COMANDO = New SqlDataAdapter("insertar_Facturas", CONN)
            COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@nroFactura", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(0).Cells(0).Value
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cCedula", SqlDbType.NVarChar)).Value =
Me.CedulaTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@fecha", SqlDbType.DateTime)).Value = Me.Labelfecha.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@subtotal", SqlDbType.Decimal)).Value =
CStr(CDbl(Me.SUBTOTAL.Text))
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@iva", SqlDbType.Decimal)).Value =
CStr(CDbl(Me.IVA.Text))
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@descuento", SqlDbType.Decimal)).Value =
CStr(CDbl(Me.descuento.Text))
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value =
CStr(CDbl(Me.TOTAL_P.Text))
            COMANDO.SelectCommand.ExecuteNonQuery()



            COMANDO = New SqlDataAdapter("actualizar_caja", CONN)
            COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
```

```
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@ingresos", SqlDbType.Decimal)).Value =
Me.ingresocaja.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@egresos", SqlDbType.Decimal)).Value =
Me.egresocaja.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value = Me.totalcaja.Text
            COMANDO.SelectCommand.ExecuteNonQuery()

            ''''Se actualiza la tabla de caja
            Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
            Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
            Me.CajaDataGridView.DataSource = Me.CajaBindingSource
            Me.CajaDataGridView.Update()
            MsgBox("Factura registrada correctamente")
            Efectivo.Show()
        Else
            MessageBox.Show("No hay informacion para guardar")
        End If
    End Sub

    Public Function AbrirConexion() As Boolean
        Dim band As Boolean = False
        Try
            CONN.Open()
            band = True
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
        Return band
    End Function
    'cierra la conexion
    Public Function CerrarConexion() As Boolean
        Dim band As Boolean = False
        Try
            CONN.Close()
            band = True
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
        Return band
    End Function

    Public Function EjecutarTransaccion(ByVal ListaSentencias As
ArrayList) As Boolean
        Dim band As Boolean = False
        If AbrirConexion() Then
            Dim command As SqlCommand = CONN.CreateCommand()
            Dim transaction As SqlTransaction
            Dim strSentencia As Object
            Dim sentencia As String = ""
            transaction = CONN.BeginTransaction()
            command.Connection = CONN
            command.Transaction = transaction
            Try
                For Each strSentencia In ListaSentencias
                    sentencia = strSentencia.ToString()
                    command.CommandText = sentencia.ToString()
                    command.ExecuteNonQuery()
```

```
                Next
                transaction.Commit()
                band = True
            Catch ex As Exception
                MessageBox.Show(ex.Message)
                Try
                    transaction.Rollback()
                Catch ex2 As Exception
                    MessageBox.Show(ex2.Message)
                End Try
            Finally
                CerrarConexion()
            End Try
        End If
        Return band
    End Function

    Private Sub BtnSALIR_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnSALIR.Click
        Me.Close()
    End Sub

    Private Sub CedulaTextBox_Validating(ByVal sender As Object, ByVal
e As System.ComponentModel.CancelEventArgs) Handles
CedulaTextBox.Validating
        Dim MD As New negocio
        Dim cedula12 As String
        cedula12 = CedulaTextBox.Text
        ''***VERIFICA SI LA CEDULA ES CORRECTA***
        If MD.VerificaCedula(cedula12) = False Then
            'Codigo en caso de que la cedula no pase la validacion
            'MsgBox("cedula incorrecta")
            CedulaTextBox.Focus()
        Else
            NombresTextBox.Focus()
        End If
    End Sub

    Private Sub BtnCANCELAR_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnCANCELAR.Click
        Me.limpiar_cajas()
        Me.limpiar_valores()
        Me.Labelfecha.Text = ""
    End Sub


    Private Sub Label13_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)

    End Sub

    Private Sub Button1_Click_2(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnIMPRIMIR.Click
        Dim j As Integer = 4
        Dim rdInforme As New Report_FACTURA2

        rdInforme.DataDefinition.FormulaFields.Item(0).Text = "'" &
NombresTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(60).Text = "'" &
ApellidosTextBox.Text & "'"
```

```
        rdInforme.DataDefinition.FormulaFields.Item(1).Text = "'" &
DireccionTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(2).Text = "'" &
CiudadTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(3).Text = "'" &
TelefonoTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(4).Text = "'" &
labelNroCompra.Text & "'"
        For i As Integer = 0 To (Me.DetFacturaDataGridView.Rows.Count)
- 1
            If i <= (Me.DetFacturaDataGridView.Rows.Count) - 1 Then
                w = j + 1
                rdInforme.DataDefinition.FormulaFields.Item(w).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(2).Value) & "'"
                x = w + 1
                rdInforme.DataDefinition.FormulaFields.Item(x).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(3).Value) & "'"
                y = x + 1
                rdInforme.DataDefinition.FormulaFields.Item(y).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(4).Value) & "'"
                z = y + 1
                rdInforme.DataDefinition.FormulaFields.Item(z).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(5).Value) & "'"
                j = z
            End If
        Next i

        rdInforme.DataDefinition.FormulaFields.Item(57).Text = "'" &
SUBTOTAL.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(61).Text = "'" &
descuento.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(62).Text = "'" &
subt_descuento.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(58).Text = "'" &
IVA.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(59).Text = "'" &
TOTAL_P.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(63).Text = "'" &
ComboDESC.Text & "'"


        Form_FACTURA.CrystalReportViewer1.ReportSource = rdInforme
        Form_FACTURA.Show()


    End Sub

    Private Sub ComboDESC_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboDESC.Click
        Me.descuento.Text = FormatCurrency(CStr(CDbl(Me.SUBTOTAL.Text)
* CDbl(CDbl(ComboDESC.Text) / 100)), 2)
        Me.subt_descuento.Text =
FormatCurrency(CStr(CDbl(Me.SUBTOTAL.Text) - CDbl(Me.descuento.Text)),
2)
        Me.IVA.Text = FormatCurrency(CStr(CDbl(Me.subt_descuento.Text)
* 0.12), 2)
        Me.TOTAL_P.Text =
FormatCurrency(CStr(CDbl(Me.subt_descuento.Text) + CDbl(Me.IVA.Text)),
2)
    End Sub
```

```
End Class
```

FORMULARIO COMPRA

```
Option Explicit On
Option Strict On

Imports System.Data
Imports System.Data.SqlClient
Public Class Compra
    '''***CADENA DE CONEXIÓN A LA BASE DE DATOS*******************
    Public CONN As New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
    'Public CONN As New SqlConnection("Data Source=COMANDATO-
VAIO\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")

    Dim COMANDO As SqlDataAdapter
    Dim DS As New DataSet
    Dim dw As New DataView
    Dim NUM As Integer = 0
    Dim filas As Integer = 0
    Private WithEvents bs As New BindingSource
    Private bEdit As Boolean


    Dim w As Integer = 0
    Dim x As Integer = 0
    Dim y As Integer = 0
    Dim z As Integer = 0
    Sub CargarDatos()
        Dim daCategories As New SqlDataAdapter("Select * From
detCompra", CONN)
        DS.Clear()
        daCategories.Fill(DS, "detCompra")
        Me.DetCompraDataGridView.DataSource = DS
        Me.DetCompraDataGridView.DataMember = "detCompra"
    End Sub

    Private Sub BtnBUSCARcli_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnBUSCARcli.Click
        Busq_Proveedores.llama.Text = CStr(1)
        Busq_Proveedores.Show()
    End Sub

    Private Sub Compra_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.caja' Puede moverla o quitarla según sea necesario.
        Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Compra' Puede moverla o quitarla según sea necesario.
        Me.CompraTableAdapter.Fill(Me.DatosSQLDataSet.Compra)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.FormaDePago' Puede moverla o quitarla según sea
necesario.

Me.FormaDePagoTableAdapter.Fill(Me.DatosSQLDataSet.FormaDePago)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.detCompra' Puede moverla o quitarla según sea
necesario.
        Me.DetCompraTableAdapter.Fill(Me.DatosSQLDataSet.detCompra)
```

```
        limpiar_valores()
        COMANDO = New SqlDataAdapter("select * from FormaDePago",
CONN)
        DS = New DataSet
        COMANDO.Fill(DS, "FormaDePago")
        Me.FormaDePagoDataGridView.DataSource =
DS.Tables("FormaDePago")
        llama.Text = CStr(10)
        Me.lleva.Text = CStr(0)
        Me.filita.Text = CStr(0)

        deshabilitar_comandos()
        deshabilitar_botones()
        deshabilitar()
        limpiar_valores()

        ComboBox1.Text = ""

        'DetFacturaDataGridView.RowsDefaultCellStyle.BackColor =
Color.White
        'DetFacturaDataGridView.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter

    End Sub

    Private Sub BtnNuevoItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnNuevoItem.Click
        Lista_Producto2.Show()
        lleva.Text = CStr(1)
        Lista_Producto2.lbingreso.Visible = True
        Lista_Producto2.nroCompra.Text = labelNroCompra.Text
        Lista_Producto2.CodigoTextBox.Text = ""
        Lista_Producto2.NombreTextBox.Text = ""
        Lista_Producto2.StockTextBox.Text = ""
        Lista_Producto2.costo.Text = ""

    End Sub

    Private Sub DetFacturaDataGridView_RowHeaderMouseClick(ByVal
sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellMouseEventArgs)
        If e.RowIndex >= 0 Then
            filita.Text = CStr(e.RowIndex)
        End If
    End Sub

    Private Sub BtnModificarItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnModificarItem.Click
        If CDbl(filita.Text) >= 0 Then
            Lista_Producto2.Show()
            Lista_Producto2.lbmodificar.Visible = True
            Lista_Producto2.buscarpro2()
            Lista_Producto2.Button1.Text = "Modificar"
            Lista_Producto2.Button1.Enabled = False
            Lista_Producto2.GroupBox1.Enabled = False
            lleva.Text = CStr(2)
            Lista_Producto2.Button1.Enabled = True
        Else
            MsgBox("Debe seleccionar con el puntero el pedido en la
Grilla a Modificar")
```

```
        End If
    End Sub

    Private Sub BtnEliminarItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnEliminarItem.Click
        If CDbl(filita.Text) >= 0 Then
            Lista_Producto2.Show()
            Lista_Producto2.lbeliminar.Visible = True
            Lista_Producto2.buscarpro2()
            Lista_Producto2.StockTextBox1.Visible = True
            Lista_Producto2.StockTextBox1.Text =
Lista_Producto2.StockTextBox.Text
            Lista_Producto2.costo1.Text = Lista_Producto2.costo.Text
            Lista_Producto2.costo1.Visible = False
            Lista_Producto2.StockTextBox1.Enabled = False
            Lista_Producto2.Button1.Text = "Eliminar"
            Lista_Producto2.GroupBox1.Enabled = False
            Lista_Producto2.RdStock.Enabled = False
            lleva.Text = CStr(3)
            Lista_Producto2.Button1.Enabled = True
        Else
            MsgBox("Debe seleccionar el pedido en la Grilla")
        End If
    End Sub
    Public Sub limpiar_cajas()
        Me.CedulaTextBox.Text = ""
        Me.NombresTextBox.Text = ""
        Me.ApellidosTextBox.Text = ""
        Me.CiudadTextBox.Text = ""
        Me.DireccionTextBox.Text = ""
        Me.TelefonoTextBox.Text = ""
        Me.referencia.Text = ""
        Me.ComboBox1.Text = ""
        Me.dias.Text = ""
        Me.fecha_venc.Text = ""
        Me.SUBTOTAL.Text = ""
        Me.IVA.Text = ""
        Me.TOTAL_P.Text = ""
        Me.labelNroCompra.Text = "00000"

    End Sub

    Private Sub BtnNUEVO_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnNUEVO.Click
        limpiar_valores()
        Me.SUBTOTAL.Focus()
        ''''Se actualiza la tabla detCompra
        Me.DetCompraTableAdapter.Fill(Me.DatosSQLDataSet.detCompra)
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

        Me.BtnBUSCARcli.Enabled = True
        Me.Labelfecha.Text = CStr(Date.Today)
        Me.CompraBindingSource.MoveLast()
        habilitartexto()

        ''NUM = 1
        NUM = CInt(CDbl(NroCompraTextBox1.Text) + 1)
        labelNroCompra.Text = "00000" & CStr(CInt(NUM))
        ''labelNroCompra.Text = CStr(NUM)
        Me.Fecha_nac.Text = Me.Labelfecha.Text
```

```
        BtnGUARDAR.Enabled = True
    End Sub

    Private Sub DetCompraBindingNavigatorSaveItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.DetCompraBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
    End Sub

    Private Sub BtnGUARDAR_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnGUARDAR.Click
        If DetFacturaDataGridView.Rows.Count > 0 Then
            ''**CADENA DE CONEXIÓN A LA BASE DE DATOS********
            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            CONN.Open()
            Try
                COMANDO = New SqlDataAdapter("insertar_Compra", CONN)
                COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@nroCompra", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(0).Cells(0).Value
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@pCedula", SqlDbType.NVarChar)).Value =
Me.CedulaTextBox.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@fecha", SqlDbType.DateTime)).Value = Me.Labelfecha.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@IdFormaPago", SqlDbType.Int)).Value = Me.idforma.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@referencia", SqlDbType.NVarChar)).Value =
Me.referencia.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value = Me.TOTAL_P.Text
                COMANDO.SelectCommand.ExecuteNonQuery()


                COMANDO = New SqlDataAdapter("insertar_ctasporpagar",
CONN)
                COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@nroCompra", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(0).Cells(0).Value
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@diasPlazo", SqlDbType.Int)).Value = Me.dias.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@fecha", SqlDbType.DateTime)).Value = Me.Labelfecha.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@fechaVenc", SqlDbType.DateTime)).Value =
Me.fecha_venc.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@detalle", SqlDbType.NVarChar)).Value =
Me.referencia.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@valor", SqlDbType.Decimal)).Value = Me.TOTAL_P.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@abono", SqlDbType.Decimal)).Value = 0
```

```
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@estado", SqlDbType.Decimal)).Value = Me.TOTAL_P.Text
                COMANDO.SelectCommand.ExecuteNonQuery()


            Catch ex As Exception
                MessageBox.Show(ex.Message)
            End Try


            Try
                For i As Integer = 0 To
DetFacturaDataGridView.Rows.Count - 1
                    COMANDO = New
SqlDataAdapter("insertar_detCompra1", CONN)
                    COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@nroCompra", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(0).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@codigo", SqlDbType.NVarChar)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(1).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cantidad", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(2).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@detalle", SqlDbType.NVarChar)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(3).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@pCompra", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(4).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(5).Value
                    COMANDO.SelectCommand.ExecuteNonQuery()



                    COMANDO = New SqlDataAdapter("insertar_KARDEX1",
CONN)
                    COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@codigoArt", SqlDbType.NVarChar)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(1).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@fecha", SqlDbType.DateTime)).Value = Me.Labelfecha.Text
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@detalle", SqlDbType.NVarChar)).Value = "Compra S/C No."
& CStr(Me.DetFacturaDataGridView.Rows(i).Cells(0).Value)
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cantidadIn", SqlDbType.Int)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(2).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@punitarioIn", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(4).Value
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@totalIn", SqlDbType.Decimal)).Value =
Me.DetFacturaDataGridView.Rows(i).Cells(5).Value
```

```
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@cantidadSa", SqlDbType.Int)).Value = 0
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@punitarioSa", SqlDbType.Decimal)).Value = 0
                    COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@totalSa", SqlDbType.Decimal)).Value = 0
                    COMANDO.SelectCommand.ExecuteNonQuery()
                Next
            Catch ex As Exception
                MessageBox.Show(ex.Message)
            End Try

            If ComboBox1.Text = "CONTADO" Then
                COMANDO = New SqlDataAdapter("actualizar_caja2", CONN)
                COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@ingresos", SqlDbType.Decimal)).Value =
Me.ingresocaja.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@egresos", SqlDbType.Decimal)).Value =
Me.egresocaja.Text
                COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value = Me.totalcaja.Text
                COMANDO.SelectCommand.ExecuteNonQuery()

                ''''Se actualiza la tabla de caja
                Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
                Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
                Me.CajaDataGridView.DataSource = Me.CajaBindingSource
                Me.CajaDataGridView.Update()

            End If
            MsgBox("Compra registrada correctamente")
            deshabilitar_comandos()
            deshabilitar_botones()
            limpiar_cajas()
            Me.DetFacturaDataGridView.DataSource = Nothing
            Me.DetFacturaDataGridView.Refresh()
            Me.DetFacturaDataGridView.Rows.Clear()
            Me.DetFacturaDataGridView.ClearSelection()
        Else
            MessageBox.Show("No hay informacion para guardar")
        End If
    End Sub
    Public Function AbrirConexion() As Boolean
        Dim band As Boolean = False
        Try
            CONN.Open()
            band = True
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
        Return band
    End Function
    'cierra la conexion
    Public Function CerrarConexion() As Boolean
        Dim band As Boolean = False
        Try
            CONN.Close()
```

```
            band = True
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
        Return band
    End Function

    Public Function EjecutarTransaccion(ByVal ListaSentencias As
ArrayList) As Boolean
        Dim band As Boolean = False
        If AbrirConexion() Then
            Dim command As SqlCommand = CONN.CreateCommand()
            Dim transaction As SqlTransaction
            Dim strSentencia As Object
            Dim sentencia As String = ""
            transaction = CONN.BeginTransaction()
            command.Connection = CONN
            command.Transaction = transaction
            Try
                For Each strSentencia In ListaSentencias
                    sentencia = strSentencia.ToString()
                    command.CommandText = sentencia.ToString()
                    command.ExecuteNonQuery()
                Next
                transaction.Commit()
                band = True
            Catch ex As Exception
                MessageBox.Show(ex.Message)
                Try
                    transaction.Rollback()
                Catch ex2 As Exception
                    MessageBox.Show(ex2.Message)
                End Try
            Finally
                CerrarConexion()
            End Try
        End If
        Return band
    End Function
    Public Sub deshabilitar_comandos()
        Me.BtnGUARDAR.Enabled = False
        Me.BtnCANCELAR.Enabled = True
        Me.BtnIMPRIMIR.Enabled = True
        Me.BtnBUSCARcli.Enabled = False

    End Sub
    Public Sub deshabilitar_botones()
        Me.BtnNuevoItem.Enabled = False
        Me.BtnModificarItem.Enabled = False
        Me.BtnEliminarItem.Enabled = False

    End Sub
    Public Sub habilitar_botones()
        Me.BtnNuevoItem.Enabled = True
        Me.BtnModificarItem.Enabled = True
        Me.BtnEliminarItem.Enabled = True

    End Sub
    Public Sub deshabilitar()
        Me.CedulaTextBox.Enabled = False
```

```
        Me.NombresTextBox.Enabled = False
        Me.ApellidosTextBox.Enabled = False
        Me.DireccionTextBox.Enabled = False
        Me.CiudadTextBox.Enabled = False
        Me.TelefonoTextBox.Enabled = False
        Me.ComboBox1.Enabled = False
        Me.dias.Enabled = False
        Me.fecha_venc.Enabled = False
        Me.referencia.Enabled = False
    End Sub
    Public Sub habilitartexto()
        Me.CedulaTextBox.Enabled = True
        Me.NombresTextBox.Enabled = True
        Me.ApellidosTextBox.Enabled = True
        Me.DireccionTextBox.Enabled = True
        Me.CiudadTextBox.Enabled = True
        Me.TelefonoTextBox.Enabled = True
        Me.ComboBox1.Enabled = True
        Me.dias.Enabled = True
        Me.fecha_venc.Enabled = True
        Me.referencia.Enabled = True
    End Sub

    Private Sub BtnSALIR_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnSALIR.Click
        Me.Close()
    End Sub


    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        Me.FormaDePagoDataGridView.DataSource = Nothing
        dw.Table = DS.Tables("FormaDePago")
        dw.RowFilter = "detalle='" & ComboBox1.Text & "'"

        If dw.Count > 0 Then
            Me.FormaDePagoDataGridView.DataSource = dw
            Me.FormaDePagoDataGridView.Refresh()
            Me.idforma.Text = CStr(Me.FormaDePagoDataGridView.Item(0,
0).Value)
            Me.dias.Text = CStr(Me.FormaDePagoDataGridView.Item(2,
0).Value)
            Me.fecha_venc.Text =
CStr(Me.Fecha_nac.Value.AddDays(CDbl(dias.Text)))
            Me.referencia.Text = ""
            Me.referencia.Focus()
        End If

    End Sub



    Private Sub CedulaTextBox_Validating(ByVal sender As Object, ByVal
e As System.ComponentModel.CancelEventArgs) Handles
CedulaTextBox.Validating
        Dim MD As New negocio
        Dim cedula12 As String
        cedula12 = CedulaTextBox.Text
        ''***VERIFICA SI LA CEDULA ES CORRECTA***
```

```
        If MD.VerificaCedula(cedula12) = False Then
            CedulaTextBox.Focus()
        Else
            NombresTextBox.Focus()
        End If
    End Sub

    Private Sub BtnEFECTIVO_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)
        EfectivoC.Show()
        EfectivoC.Pago_efe.Focus()
    End Sub



    Public Sub limpiar_valores()
        Me.SUBTOTAL.Text = "$ 0.00"
        Me.IVA.Text = "$ 0.00"
        Me.TOTAL_P.Text = "$ 0.00"
    End Sub

    Private Sub BtnCANCELAR_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnCANCELAR.Click
        Me.limpiar_cajas()
        Me.limpiar_valores()
        Me.Labelfecha.Text = ""

    End Sub

    Private Sub referencia_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles referencia.TextChanged
        Me.referencia.CharacterCasing = CharacterCasing.Upper
    End Sub


    Private Sub BtnIMPRIMIR_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnIMPRIMIR.Click
        Dim j As Integer = 4
        Dim rdInforme As New Report_COMPRA2

        rdInforme.DataDefinition.FormulaFields.Item(0).Text = "'" &
NombresTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(60).Text = "'" &
ApellidosTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(1).Text = "'" &
DireccionTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(2).Text = "'" &
CiudadTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(3).Text = "'" &
TelefonoTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(4).Text = "'" &
labelNroCompra.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(61).Text = "'" &
ComboBox1.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(62).Text = "'" &
fecha_venc.Text & "'"
```

```
        For i As Integer = 0 To (Me.DetFacturaDataGridView.Rows.Count)
- 1
            If i <= (Me.DetFacturaDataGridView.Rows.Count) - 1 Then
                w = j + 1
                rdInforme.DataDefinition.FormulaFields.Item(w).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(2).Value) & "'"
                x = w + 1
                rdInforme.DataDefinition.FormulaFields.Item(x).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(3).Value) & "'"
                y = x + 1
                rdInforme.DataDefinition.FormulaFields.Item(y).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(4).Value) & "'"
                z = y + 1
                rdInforme.DataDefinition.FormulaFields.Item(z).Text =
"'" & CStr(Me.DetFacturaDataGridView.Rows(i).Cells(5).Value) & "'"
                j = z
            End If
        Next i

        rdInforme.DataDefinition.FormulaFields.Item(57).Text = "'" &
SUBTOTAL.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(58).Text = "'" &
IVA.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(59).Text = "'" &
TOTAL_P.Text & "'"
        Form_FACTURA.CrystalReportViewer1.ReportSource = rdInforme
        Form_FACTURA.Show()


    End Sub


    Private Sub DetFacturaDataGridView_RowHeaderMouseClick1(ByVal
sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellMouseEventArgs) Handles
DetFacturaDataGridView.RowHeaderMouseClick
        If e.RowIndex >= 0 Then
            filita.Text = CStr(e.RowIndex)
        End If
    End Sub
End Class
```

FORMULARIO CUENTAS POR PAGAR.

```
Imports System.Data
Imports System.Data.SqlClient
Public Class CuentasPorPagar
    Dim DA As SqlDataAdapter
    Dim dw As New DataView
    Dim CONN As SqlConnection
    Dim ds As New DataSet
```

```
    Private Sub CtasPorPagarBindingNavigatorSaveItem_Click(ByVal
sender As System.Object, ByVal e As System.EventArgs)
        Me.Validate()
        Me.CtasPorPagarBindingSource.EndEdit()
        Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)

    End Sub

    Private Sub CuentasPorPagar_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.caja' Puede moverla o quitarla según sea necesario.
        Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Proveedores' Puede moverla o quitarla según sea
necesario.

Me.ProveedoresTableAdapter.Fill(Me.DatosSQLDataSet.Proveedores)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.Compra' Puede moverla o quitarla según sea necesario.
        Me.CompraTableAdapter.Fill(Me.DatosSQLDataSet.Compra)
        'TODO: esta línea de código carga datos en la tabla
'DatosSQLDataSet.CtasPorPagar' Puede moverla o quitarla según sea
necesario.

Me.CtasPorPagarTableAdapter.Fill(Me.DatosSQLDataSet.CtasPorPagar)

        '***CADENA DE CONEXIÓN A LA BASE DE DATOS******
        CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        habilitar()
        caja()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        Try
            'Crear el Adaptador
            Me.NroCompraTextBox.Text = InputBox("Ingrese el N° de
Compra: ", "Ferreteria Oñate S.A.")

            If NroCompraTextBox.Text <> "" Then
                Dim da As New SqlDataAdapter("buscar_PAGO'" &
NroCompraTextBox.Text & "'", CONN)
                'crear conjunto de datos
                Dim ds As New DataSet
                'Utilizar el DataAdapter para llenar el Dataset con
una tabla
                da.Fill(ds, "CtasPorPagar")
                CtasPorPagarDataGridView.DataSource =
ds.Tables("CtasPorPagar")
                Me.CtasPorPagarDataGridView.Refresh()
                If (Me.CtasPorPagarDataGridView.Item(0, 0).Value) > 0
Then

                    Me.NroCompraTextBox.Text =
(Me.CtasPorPagarDataGridView.Item(0, 0).Value)
                    Me.DiasPlazoTextBox.Text =
(Me.CtasPorPagarDataGridView.Item(1, 0).Value)
```

```
                    Me.fecha_compra.Text =
(Me.CtasPorPagarDataGridView.Item(2, 0).Value)
                    Me.fecha_venc.Text =
(Me.CtasPorPagarDataGridView.Item(3, 0).Value)
                    Me.DetalleTextBox.Text =
(Me.CtasPorPagarDataGridView.Item(4, 0).Value)
                    Me.ValorTextBox.Text =
(Me.CtasPorPagarDataGridView.Item(5, 0).Value)
                    Me.AbonoTextBox.Text =
(Me.CtasPorPagarDataGridView.Item(6, 0).Value)
                    Me.EstadoTextBox.Text =
(Me.CtasPorPagarDataGridView.Item(7, 0).Value)
                    Me.prueba.Focus()
                Else
                    habilitar()
                    MsgBox("LA CUENTA POR PAGAR NO EXISTE")
                End If
            End If


            If NroCompraTextBox.Text <> "" Then
                Dim da As New SqlDataAdapter("buscar_PAGO_Compra'" &
NroCompraTextBox.Text & "'", CONN)
                'crear conjunto de datos
                Dim ds As New DataSet
                'Utilizar el DataAdapter para llenar el Dataset con
una tabla
                da.Fill(ds, "Compra")
                CompraDataGridView.DataSource = ds.Tables("Compra")
                Me.CompraDataGridView.Refresh()

                Me.PCedulaTextBox.Text =
(Me.CompraDataGridView.Item(1, 0).Value)

            End If
            If PCedulaTextBox.Text <> "" Then
                Dim da As New SqlDataAdapter("buscar_PAGO_Proveedor'"
& PCedulaTextBox.Text & "'", CONN)
                'crear conjunto de datos
                Dim ds As New DataSet
                'Utilizar el DataAdapter para llenar el Dataset con
una tabla
                da.Fill(ds, "Proveedores")
                ProveedoresDataGridView.DataSource =
ds.Tables("Proveedores")
                Me.ProveedoresDataGridView.Refresh()
                Me.NombresTextBox.Text =
(Me.ProveedoresDataGridView.Item(1, 0).Value)
                Me.ApellidosTextBox.Text =
(Me.ProveedoresDataGridView.Item(2, 0).Value)

            End If
        Catch ex As SqlException
            MessageBox.Show(ex.Message)
        End Try
    End Sub

    Private Sub habilitar()
        Me.NroCompraTextBox.Text = ""
        Me.DiasPlazoTextBox.Text = ""
```

```
        Me.fecha_venc.Text = ""
        Me.fecha_compra.Text = ""
        Me.DetalleTextBox.Text = ""
        Me.ValorTextBox.Text = ""
        Me.AbonoTextBox.Text = ""
        Me.EstadoTextBox.Text = ""
        Me.PCedulaTextBox.Text = ""
        Me.NombresTextBox.Text = ""
        Me.ApellidosTextBox.Text = ""
        Me.prueba.Text = ""
        Me.Button5.Enabled = False
        Me.Button4.Enabled = False
    End Sub
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button3.Click
        Me.Close()
    End Sub

    Public Sub caja()
        Me.ingresocaja.Text = CDbl(CajaDataGridView.Item(0, 0).Value)
        Me.egresocaja.Text = CDbl(CajaDataGridView.Item(1, 0).Value)
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button5.Click


        If Me.DiasPlazoTextBox.Text > 0 Then


            If CDbl(Me.prueba.Text) <= CDbl(Me.ValorTextBox.Text) Then
                Me.ValorTextBox.Text = CDbl(Me.ValorTextBox.Text) -
CDbl(Me.prueba.Text)
                'Me.AbonoTextBox.Text = CDbl(Me.AbonoTextBox.Text) +
(Me.CtasPorPagarDataGridView.Item(0, 5).Value)
                Me.AbonoTextBox.Text = CDbl(Me.AbonoTextBox.Text) +
CDbl(Me.prueba.Text)
                Me.EstadoTextBox.Text = EstadoTextBox.Text +
Me.AbonoTextBox.Text

                Me.egresocaja.Text = CDbl(Me.egresocaja.Text) +
CDbl(Me.prueba.Text)
                Me.totalcaja.Text = CDbl(Me.ingresocaja.Text) -
CDbl(Me.egresocaja.Text)

                '''''CADENA DE CONEXIÓN DE LA BASE DE DATOS****
                CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
                CONN.Open()
                DA = New SqlDataAdapter("ACT_CTASPAGAR", CONN)
                DA.SelectCommand.CommandType =
CommandType.StoredProcedure
                DA.SelectCommand.Parameters.Add(New
SqlParameter("nroCompra", SqlDbType.Int)).Value =
(Me.CtasPorPagarDataGridView.Item(0, 0).Value)
                DA.SelectCommand.Parameters.Add(New
SqlParameter("valor", SqlDbType.Decimal)).Value = Me.ValorTextBox.Text
                DA.SelectCommand.Parameters.Add(New
SqlParameter("abono", SqlDbType.Decimal)).Value = Me.AbonoTextBox.Text
```

```
                DA.SelectCommand.Parameters.Add(New
SqlParameter("estado", SqlDbType.Decimal)).Value =
Me.ValorTextBox.Text
                DA.SelectCommand.ExecuteNonQuery()



Me.CtasPorPagarTableAdapter.Fill(Me.DatosSQLDataSet.CtasPorPagar)
                Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
                Me.CtasPorPagarDataGridView.DataSource =
Me.CtasPorPagarBindingSource
                Me.CtasPorPagarDataGridView.Update()


                DA = New SqlDataAdapter("actualizar_caja2", CONN)
                DA.SelectCommand.CommandType =
CommandType.StoredProcedure

                DA.SelectCommand.Parameters.Add(New
SqlParameter("@ingresos", SqlDbType.Decimal)).Value =
Me.ingresocaja.Text
                DA.SelectCommand.Parameters.Add(New
SqlParameter("@egresos", SqlDbType.Decimal)).Value =
Me.egresocaja.Text
                DA.SelectCommand.Parameters.Add(New
SqlParameter("@total", SqlDbType.Decimal)).Value = Me.totalcaja.Text
                DA.SelectCommand.ExecuteNonQuery()

                ''''Se actualiza la tabla de caja
                Me.CajaTableAdapter.Fill(Me.DatosSQLDataSet.caja)
                Me.TableAdapterManager.UpdateAll(Me.DatosSQLDataSet)
                Me.CajaDataGridView.DataSource = Me.CajaBindingSource
                Me.CajaDataGridView.Update()
                Button5.Enabled = False
                MsgBox("El Saldo se Modifico con Éxito")
                deuda_inicial.Text = CDbl(CDbl(Me.AbonoTextBox.Text) +
CDbl(Me.ValorTextBox.Text))
                'habilitar()
            Else
                MsgBox("El abono sobrepasa el monto de Deuda ")

            End If
        Else

            MessageBox.Show("EL PROVEEDOR NO TIENE DEUDA", "Ferreteria
Oñate S.A.", MessageBoxButtons.OK, MessageBoxIcon.Error)
            habilitar()

        End If
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button4.Click
        Dim rdInforme As New Reporte_CtasPAGAR
        rdInforme.DataDefinition.FormulaFields.Item(0).Text = "'" &
PCedulaTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(1).Text = "'" &
NombresTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(2).Text = "'" &
ApellidosTextBox.Text & "'"
```

```
        rdInforme.DataDefinition.FormulaFields.Item(3).Text = "'" &
deuda_inicial.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(4).Text = "'" &
NroCompraTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(5).Text = "'" &
fecha_venc.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(6).Text = "'" &
prueba.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(7).Text = "'" &
AbonoTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(8).Text = "'" &
ValorTextBox.Text & "'"
        rdInforme.DataDefinition.FormulaFields.Item(10).Text = "'" &
fecha_compra.Text & "'"


        FormCtasPagar.CrystalReportViewer1.ReportSource = rdInforme
        FormCtasPagar.Show()
    End Sub


    Private Sub prueba_TextChanged_1(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles prueba.TextChanged
        Button5.Enabled = True
        Button4.Enabled = True
    End Sub
End Class.
```

FORMULARIO INVENTARIO.

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.IO

Public Class Inventario
    Dim dw As New DataView
    Dim CONN As SqlConnection
    Dim COMANDO As SqlDataAdapter
    'Dim ds As New DataSet


    ''CADENA DE CONEXIÓN DE LA BASE DE DATOS*******
    Public Cnx As New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
    Public ds As New DataSet
    Sub CargarDatos()
        Dim daCategories As New SqlDataAdapter("Select * From
Inventario order by codigo", Cnx)
        ds.Clear()
        daCategories.Fill(ds, "Inventario")
        Me.DataGridView1.DataSource = ds
        Me.DataGridView1.DataMember = "Inventario"
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        llama.Text = 0
        Call CargarDatos()
        INHABILITAR()
```

```
    End Sub


    Private Sub BtnInsertar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnInsertar.Click
        Try
            Cnx.Open()
            Dim arrFilename() As String = Split(Text, "\")
            Array.Reverse(arrFilename)

            Dim ms As New MemoryStream
            PictureBox1.Image.Save(ms, PictureBox1.Image.RawFormat)
            Dim arrImage() As Byte = ms.GetBuffer

            ''CADENA DE CONEXIÓN DE LA BASE DE DATOS*****

            CONN = New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            CONN.Open()
            COMANDO = New SqlDataAdapter("insertar_pro", CONN)
            COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("codigo", SqlDbType.NVarChar)).Value =
Me.CodigoTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("nombre", SqlDbType.NVarChar, 100)).Value =
Me.NombreTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("stock", SqlDbType.Int)).Value = Me.StockTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("pCompra", SqlDbType.Decimal, 10, 2)).Value =
Me.Pcompra1.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("ganancia", SqlDbType.Decimal, 10, 2)).Value =
Me.GananciaTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("impuesto", SqlDbType.Decimal, 10, 2)).Value =
Me.ImpuestoTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("cedulapro", SqlDbType.NVarChar)).Value =
CedulaproTextBox.Text
            COMANDO.SelectCommand.Parameters.Add(New
SqlParameter("imagen", SqlDbType.Image)).Value = arrImage
            COMANDO.SelectCommand.ExecuteNonQuery()
            COMANDO = New SqlDataAdapter("ordenar_inventarios", CONN)
            COMANDO.SelectCommand.CommandType =
CommandType.StoredProcedure

            MessageBox.Show("Se ha Guardo el registro correctamente",
"El Sistema informa", _
            MessageBoxButtons.OK, MessageBoxIcon.Information)

            BtnActualizar.Enabled = True
            BtnEliminar.Enabled = True
            Btn_BuscarCodigo.Enabled = False

            '
            Call CargarDatos()
            '
```

```
        Catch sqlExc As SqlException
            MessageBox.Show(sqlExc.ToString, "SQL Exception Error!", _
            MessageBoxButtons.OK, MessageBoxIcon.Error)
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            If ConnectionState.Open Then
                Cnx.Close()
            End If
        End Try
    End Sub


    Function ExtraerImagen(ByVal Foto As Integer) As Byte()
        Dim SqlSelect As String = "Select Imagen From Inventario Where
Codigo = " & Foto
        Dim Command As New SqlCommand(SqlSelect, Cnx)
        Cnx.Open()
        Dim MyPhoto() As Byte = CType(Command.ExecuteScalar(), Byte())
        Cnx.Close()
        Return MyPhoto
    End Function

    Private Sub DataGridView1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles DataGridView1.Click
        Try
            Me.CodigoTextBox.Text =
CStr(Me.DataGridView1.SelectedCells(0).Value)
            Me.NombreTextBox.Text =
CStr(Me.DataGridView1.SelectedCells(1).Value)
            Me.StockTextBox.Text =
CStr(Me.DataGridView1.SelectedCells(2).Value)
            Me.Pcompra.Text =
CStr(Me.DataGridView1.SelectedCells(3).Value)
            Me.GananciaTextBox.Text =
CStr(Me.DataGridView1.SelectedCells(4).Value)
            Me.ImpuestoTextBox.Text =
CStr(Me.DataGridView1.SelectedCells(5).Value)
            Me.CedulaproTextBox.Text =
CStr(Me.DataGridView1.SelectedCells(6).Value)
            'El MemoryStream nos permite crear un almacen de memoria
            Dim ms As New
MemoryStream(ExtraerImagen(CStr(Me.CodigoTextBox.Text)))
            Me.PictureBox1.Image = Image.FromStream(ms)
        Catch ex As Exception
        End Try
    End Sub

    Private Sub BtnNuevo_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnNuevo.Click
        INHABILITAR()
        Me.CodigoTextBox.Text = ""
        Me.NombreTextBox.Text = ""
        Me.StockTextBox.Text = ""
        Me.CedulaproTextBox.Text = ""
        Me.nombrepro.Text = ""
        Me.Pcompra.Text = "0.00"
        Me.GananciaTextBox.Text = "0.00"
        Me.ImpuestoTextBox.Text = "0.00"
        Btn_BuscarCodigo.Enabled = True
```

```
        Me.CodigoTextBox.Focus()
    End Sub

    Private Sub BtnExtraer_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnExtraer.Click
        Try
            Dim saveImage As New SaveFileDialog 'Este es el
SaveFileDialog
            Dim ruta As String = "" 'Para tener la ruta de la imagen

            saveImage.Title = "Guardar imagen como..." 'Título de la
ventana
            saveImage.Filter = "Imagen BMP (*.bmp)|*.bmp|Imagen JPG
(*.jpg)|*.jpg|Imagen PNG (*.png)|*.png" 'Los formatos en que se
guardará la imagen
            If saveImage.ShowDialog() = Windows.Forms.DialogResult.OK
Then
                'Recuperar la ruta de la imagen si no está vacía
                If Not String.IsNullOrEmpty(saveImage.FileName) Then
ruta = saveImage.FileName

                Dim myImg As Image 'Objeto Image para guardar la
imagen del Picture
                Dim extension As String = ruta.Substring(ruta.Length -
3, 3) 'Recuperar los ultimos 3 caracteres de la extensión

                myImg = PictureBox1.Image 'Guardar la imagen del
PictureBox en el objeto Image
                Select Case extension
                    Case "bmp"
                        myImg.Save(ruta, Imaging.ImageFormat.Bmp)
'Guardar en formato BMP
                    Case "jpg"
                        myImg.Save(ruta, Imaging.ImageFormat.Jpeg)
'Guardar en formato JPG
                    Case "png"
                        myImg.Save(ruta, Imaging.ImageFormat.Png)
'Guardar en formato PNG
                End Select
            End If
        Catch ex As Exception
            MsgBox("Ocurrió el siguiente error: " & ex.Message,
MsgBoxStyle.Critical, "Error!")
        End Try
    End Sub

    Private Sub BtnActualizar_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnActualizar.Click
        'CADENA DE CONEXIÓN DE LA BASE DE DATOS
        Dim conectar As New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
        conectar.Open()
        Dim ms As New MemoryStream
        PictureBox1.Image.Save(ms, PictureBox1.Image.RawFormat)
        Dim arrImage() As Byte = ms.GetBuffer

        'realizamos la operacion SQL
        Dim modificar As New SqlClient.SqlCommand()
        modificar.CommandType = System.Data.CommandType.Text
```

```
        modificar.CommandText = "Update Inventario Set Nombre = '" &
NombreTextBox.Text & "', Stock = '" & StockTextBox.Text & "',
imagen=@imagen WHERE codigo = '" & CodigoTextBox.Text & "'"

        modificar.Parameters.Add(New SqlParameter("@Imagen",
SqlDbType.Image)).Value = arrImage

        'reliazamos la conexion
        modificar.Connection = conectar
        Try
            If ((modificar.ExecuteNonQuery <> 0)) Then
                MsgBox("DATOS ACTUALIZADOS")
                CargarDatos()
            End If
            '  Operaciones.CargarInformacion()
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
    End Sub

    Private Sub BtnEliminar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnEliminar.Click
        'Mensaje si queremos eliminar el campo
        If MessageBox.Show("¿Esta seguro de Borrar los Datos?",
"Responda", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) =
Windows.Forms.DialogResult.Yes Then
            'creamos cadena de conexion
            ''''''cadena de conexion de la base de datos
            Dim conectar As New SqlConnection("Data Source=RITA-
PC\SQLEXPRESS;Initial Catalog=DatosSQL;Integrated Security=True")
            conectar.Open()
            'realizamos la operacion SQL
            Dim ELIMINAR As New SqlClient.SqlCommand()
            ELIMINAR.CommandType = System.Data.CommandType.Text
            ELIMINAR.CommandText = "Delete From Inventario Where
Codigo = '" & CodigoTextBox.Text & "'"
            ELIMINAR.Connection = conectar
            'comprobamos si se elimino los datos
            Try
                If ((ELIMINAR.ExecuteNonQuery <> 0)) Then
                    MsgBox("Datos Eliminados")
                    CargarDatos()
                End If
            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try
        End If
    End Sub

    Private Sub BtnExaminar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtnExaminar.Click
        PictureBox1.Visible = True
        With OpenFD
            .InitialDirectory = ""
            .FileName = "Todos los Archivos"
            .Filter = "Todos los
Archivos|*.*|JPEGs|*.jpg|GIFs|*.gif|Bitmaps|*.bmp"
            .FilterIndex = 2
        End With
```

```
        If OpenFD.ShowDialog() = Windows.Forms.DialogResult.OK Then
            With PictureBox1
                .Image = Image.FromFile(OpenFD.FileName)
                .SizeMode = PictureBoxSizeMode.CenterImage
                .BorderStyle = BorderStyle.Fixed3D
                '.................................
                Me.BtnInsertar.Enabled = True
            End With
        End If
    End Sub

    Private Sub BtnBUSCARcli_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BtnBUSCARcli.Click
        Busq_Proveedores.Show()
        Busq_Proveedores.llama.Text = 0
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtnCerrar.Click
        Me.Close()
    End Sub
    Private Sub INHABILITAR()
        BtnNuevo.Enabled = True
        BtnActualizar.Enabled = False
        BtnInsertar.Enabled = False
        BtnEliminar.Enabled = False
        BtnCerrar.Enabled = True
        BtnExaminar.Enabled = False
        BtnExtraer.Enabled = False
        Btn_BuscarCodigo.Enabled = False
        BtnBUSCARcli.Enabled = False

    End Sub

   Private Sub Btn_BuscarCodigo_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Btn_BuscarCodigo.Click

        If (CodigoTextBox.Text = "") Then
            MsgBox("Por favor ingrese codigo")
            CodigoTextBox.Focus()
        Else
            Me.DataGridView1.DataSource = Nothing
            dw.Table = ds.Tables("Inventario")
            dw.RowFilter = "codigo='" & CodigoTextBox.Text & "'"

            If dw.Count > 0 Then
                Me.DataGridView1.DataSource = dw
                Me.DataGridView1.Refresh()
                MsgBox("el producto ya existe")
                CodigoTextBox.Text = ""
                CodigoTextBox.Focus()
            Else
                MsgBox("Producto NO Registrado ")
                BtnInsertar.Enabled = True
                Me.NombreTextBox.Enabled = True
                Me.StockTextBox.Enabled = True
                Me.Pcompra.Enabled = True
                Me.GananciaTextBox.Enabled = True
                Me.ImpuestoTextBox.Enabled = True
                Me.CedulaproTextBox.Enabled = True
```

```
                NombreTextBox.Focus()
                Me.BtnBUSCARcli.Enabled = True
                Me.BtnExaminar.Enabled = True
                Me.BtnExtraer.Enabled = True
            End If
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        Me.CodigoTextBox.Text = InputBox("Ingrese el CODIGO DEL
PRODUCTO: ", "Ferreteria Oñate S.A.")
        If (Me.CodigoTextBox.Text = "") Then
            MsgBox("Por favor ingrese CODIGO DEL PRODUCTO ")
            Me.CodigoTextBox.Focus()
        Else
            Me.DataGridView1.DataSource = Nothing
            dw.Table = ds.Tables("Inventario")
            dw.RowFilter = "codigo='" & CodigoTextBox.Text & "'"
            If (dw.Count > 0) Then
                Me.DataGridView1.DataSource = dw
                Me.DataGridView1.Refresh()
                Me.CodigoTextBox.Text = (Me.DataGridView1.Item(0,
0).Value)
                Me.NombreTextBox.Text = (Me.DataGridView1.Item(1,
0).Value)
                Me.StockTextBox.Text = (Me.DataGridView1.Item(2,
0).Value)
                Me.Pcompra.Text = (Me.DataGridView1.Item(3, 0).Value)
                Me.GananciaTextBox.Text = (Me.DataGridView1.Item(4,
0).Value)
                Me.ImpuestoTextBox.Text = (Me.DataGridView1.Item(5,
0).Value)
                Me.CedulaproTextBox.Text = (Me.DataGridView1.Item(6,
0).Value)
                'Me.PictureBox1.SizeMode = (Me.DataGridView1.Item(7,
0).Value)
            Else
                MsgBox("Producto NO Registrado ")
            End If
        End If
    End Sub
    Private Sub Pcompra_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Pcompra.TextChanged
        If Pcompra.Text = "" Then
            MsgBox("INGRESE VALOR")
            Me.Pcompra.Focus()
            Me.GananciaTextBox.Text = "0.00"
            Me.ImpuestoTextBox.Text = "0.00"

        Else
            Me.Pcompra1.Text = Val(Me.Pcompra.Text) * 1
            Me.GananciaTextBox.Text = Val(Me.Pcompra.Text) *
Val(Me.StockTextBox.Text)
            Me.ImpuestoTextBox.Text = Val(Me.GananciaTextBox.Text) *
0.12
        End If
    End Sub
```

```
    Private Sub CodigoTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CodigoTextBox.TextChanged
        Me.CodigoTextBox.CharacterCasing = CharacterCasing.Upper
    End Sub

    Private Sub NombreTextBox_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NombreTextBox.TextChanged
        Me.NombreTextBox.CharacterCasing = CharacterCasing.Upper
        Me.NombreTextBox.MaxLength = 50
    End Sub

    Private Sub primero_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)

    End Sub

    Private Sub DataGridView1_CellContentClick(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
DataGridView1.CellContentClick

    End Sub

    Private Sub FillByToolStripButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Try

Me.InventarioTableAdapter.FillBy(Me.DatosSQLDataSet.Inventario)
        Catch ex As System.Exception
            System.Windows.Forms.MessageBox.Show(ex.Message)
        End Try

    End Sub
End Class
```

4.3 CONCLUSIONES Y RECOMENDACIONES PARA UNA EFICIENTE IMPLEMENTACIÓN DEL SOTFWARE

CONCLUSIONES

- ✓ La implementación de este Software proporcionara un mejor desempeño y rapidez en el manejo de de las Ventas.
- ✓ Mayor control y seguridad de datos.
- ✓ Mejor servicio de atención y resultados exitosos.
- ✓ Integración inmediata a soluciones por pedidos de Usuarios.
- ✓ Confiabilidad en el proceso de digitación y almacenamiento de datos.

RECOMENDACIONES

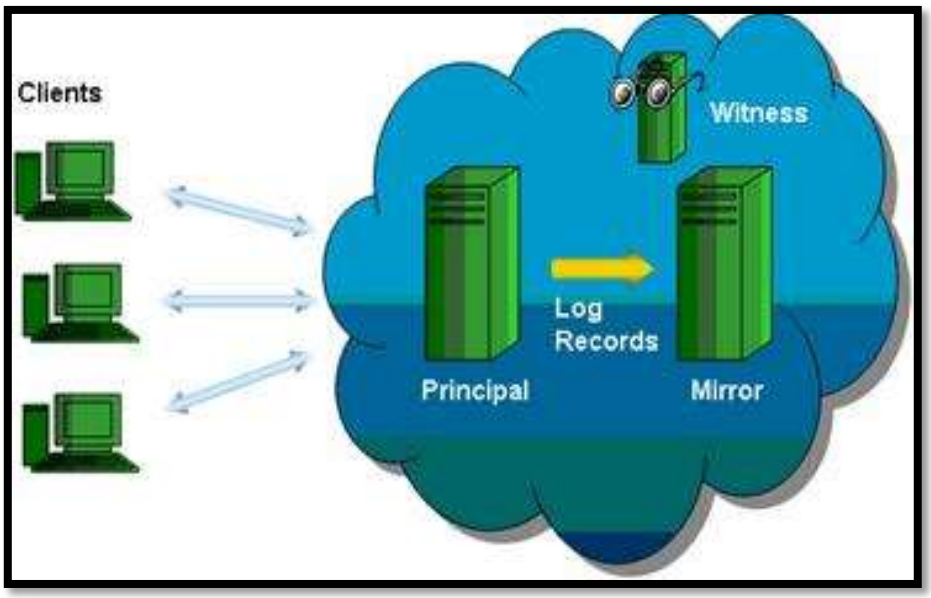
- ✓ Realizar respaldos de la base de datos, de tal manera de que siempre tenga los recursos necesarios para actualizaciones en caso de pérdida de información.
- ✓ Que el personal encargado de utilizar el software tenga una completa capacitación del sistema para un óptimo manejo de todas las aplicaciones.
- ✓ Con el desarrollo de este sistema mejorará el servicio técnico en el Área de Ventas con mayor calidad de atención al usuario.
- ✓ Mostrar solides y confianza con el sistema porque de esta manera podrá dar las soluciones necesarias en el caso requerido.

Anexo 1

La plataforma de datos SQL Server 2005

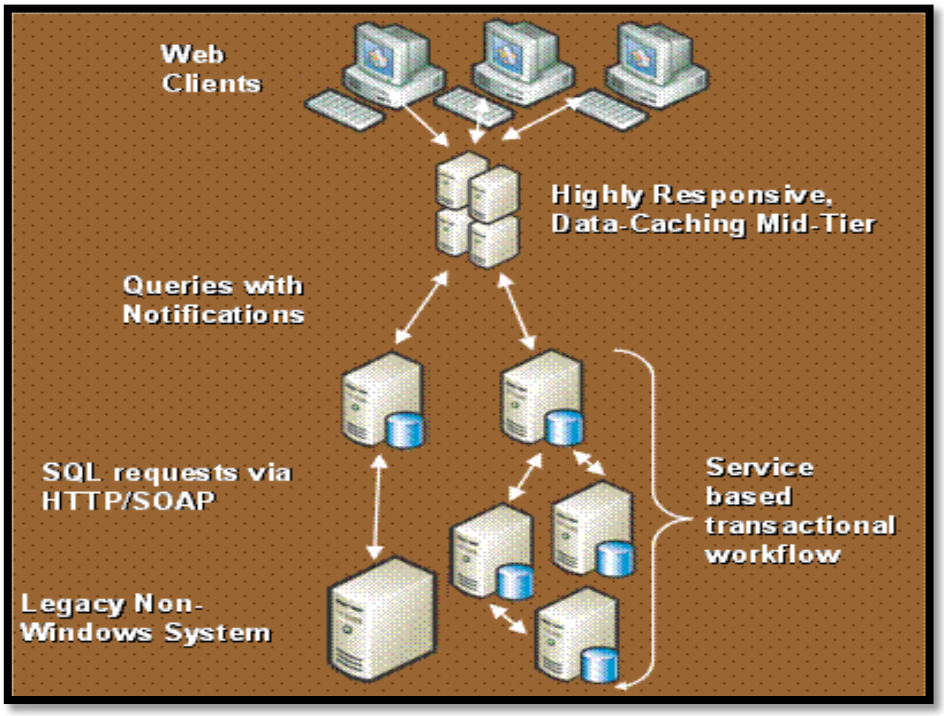
Anexo 2

Configuración básica de la creación de un reflejo de una base de datos



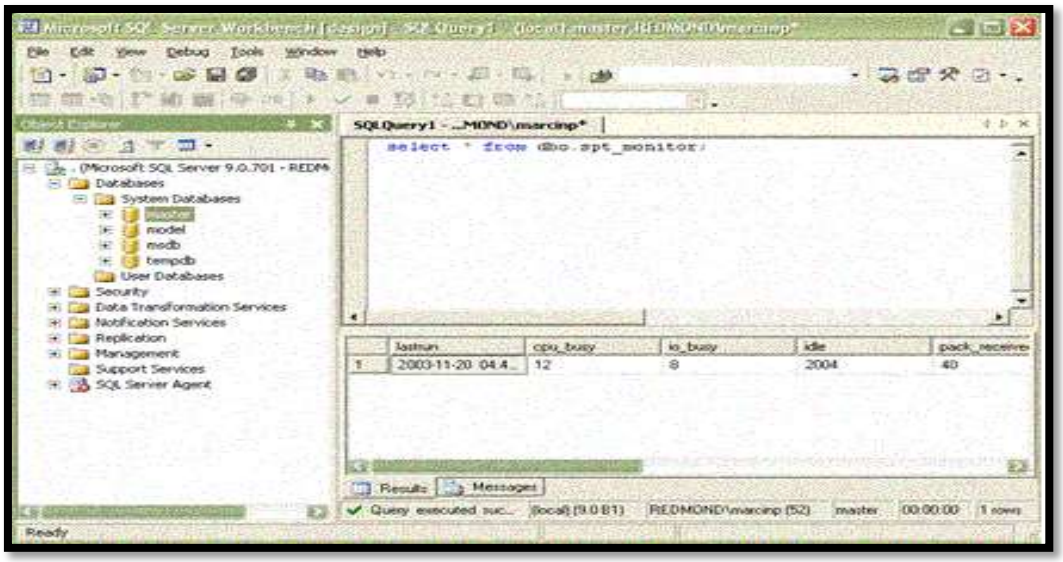
Anexo 3

Arquitectura de Service Broker



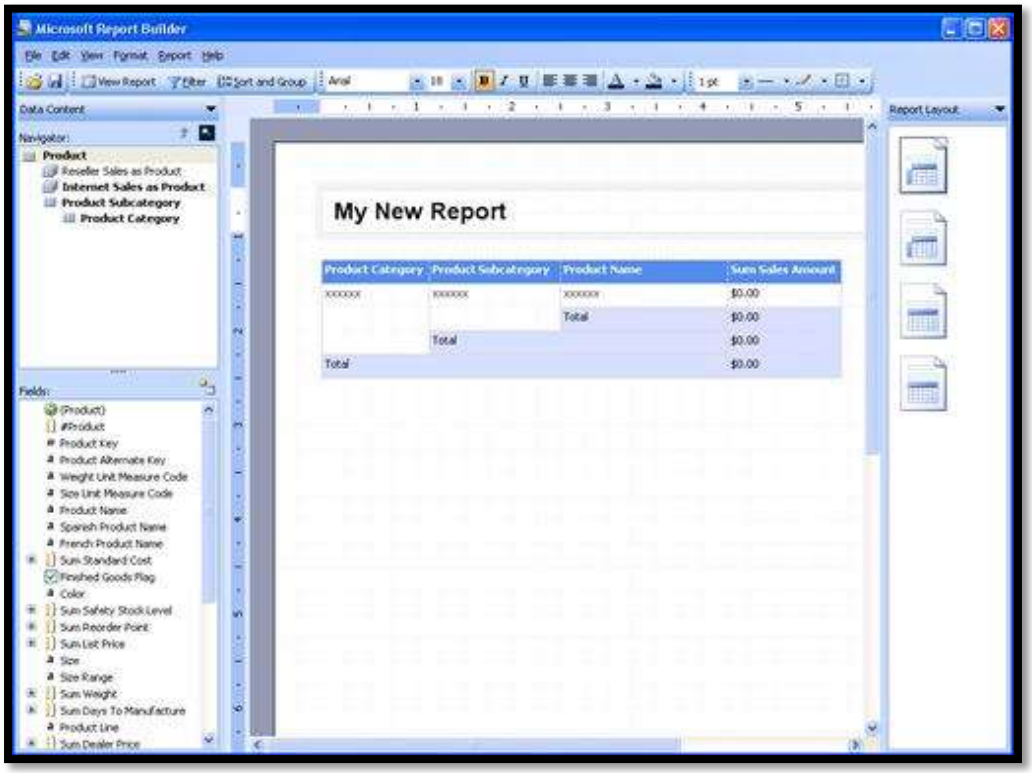
Anexo 4

El editor de consultas en SQL Server Express Manager (XM)



Anexo 5

Diseño de informes con Report Builder



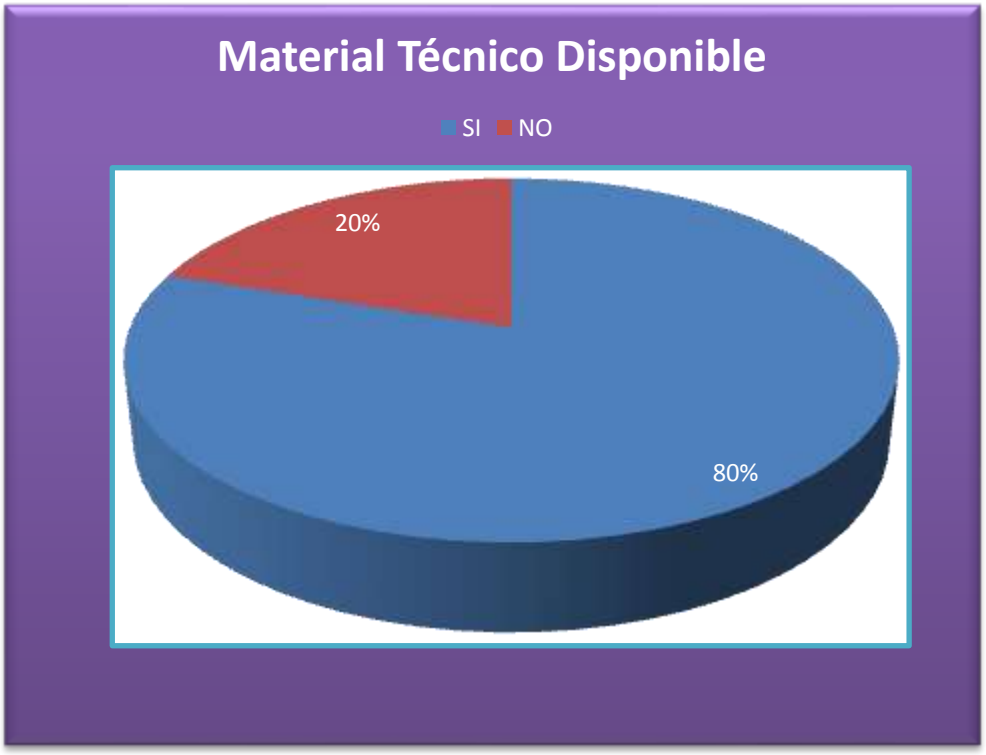
Anexo 6

Ciclo de vida de un formulario

Anexo7

Datos de Encuestados Gráficamente en la Ciudad de Babahoyo a Ferrecentro Oñate S.A.

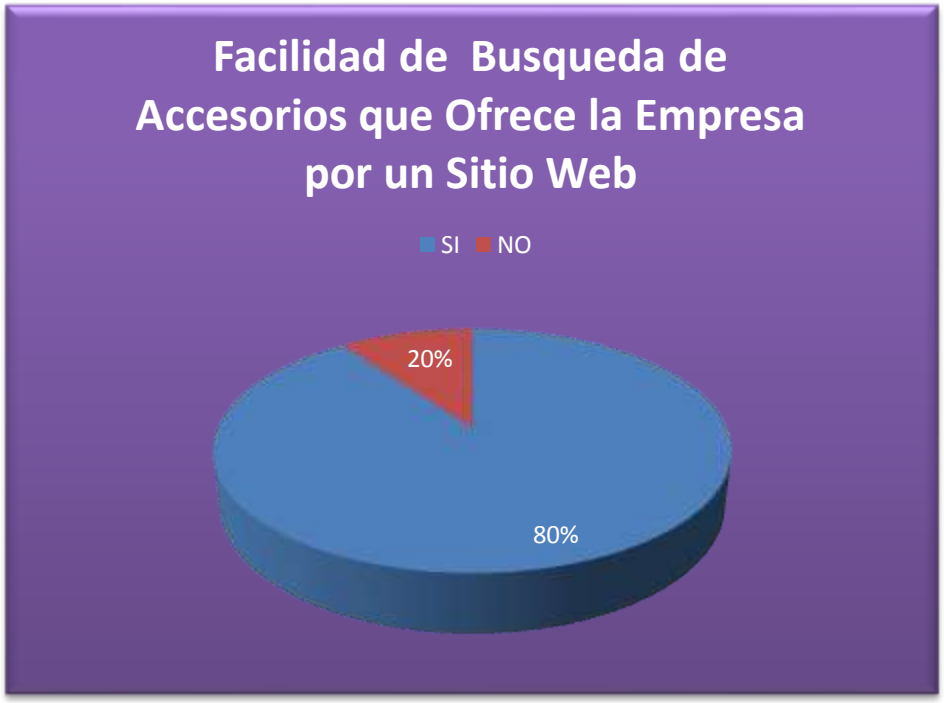

Administración de Información Manualmente
SI
NO
4%
96%


Material Técnico Disponible
SI
NO
20%
80%


Existencia de un Sistema Informático
SI
NO
10%
90%


Atención de Calidad en la Empresa
SI
NO
20%
80%


Facilidad de Pago en la Compra
SI
NO
10%
90%


Facilidad de Busqueda de Accesorios que Ofrece la Empresa por un Sitio Web
SI
NO
20%
80%


Sistemas y Comunicaciones mas Accesibles que años anteriores
SI
NO
10%
90%


Es Segura una Transacción
SI
NO
10%
90%

BIBLIOGRAFÍA

MYSQL

- ✓ Pau Dubois
- ✓ Ed. New Readers

UML Y PATRONES

- ✓ Graig Larman
- ✓ Ed. PRENTICE HALL

UML gota a gota

- ✓ Martin Fowler, Kendall Scott
- ✓ Ed PEARSON
- ✓ Notas Análisis y diseño de Bases de Datos, Diplomado en Bases de Datos
- ✓ M.C. Yolanda Moyao Martínez
- ✓ Primavera 2004
- ✓ Diplomado en Bases de Datos, Aplicaciones en Bases de Datos Locales.

MySQL- PHP.

- ✓ Dr. Ivo Humberto Pineda Torres.
- ✓ Febrero 2004

LINKOGRAFÍA

- ✓ Tutoriales PHP:
- ✓ www.programacion.com/php/
- ✓ http://es.tldp.org/Manuales-LuCAS/manual_PHP/
- ✓ www.webestilo.com/php
- ✓ Tutoriales MYSQL
- ✓ www.mysql-hispano.org
- ✓ www.mysql.com
- ✓ www.mysql.es
- ✓ http://www.alegsa.com.ar/Dic/multiplataforma.php
- ✓ http://www.monografias.com/trabajos/metoinves/metoinves.shtml

- ✓ http://www.mistareas.com.ve/Tipo-de-estudio-tipo-de-investigacion.htm
- ✓ http://www.monografias.com/trabajos15/invest-cientifica/invest-cientifica.shtml
- ✓ http://www.mistareas.com.ve/
- ✓ http://www.mistareas.com.ve/tipo-de-investigacion/que-es-la-investigacion-cualitativa.htm
- ✓ http://www.alegsa.com.ar/Dic/visual%20basic.php

5. Cronograma de Actividades.

Actividades	Mes1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6				Mes 7				Mes 8			
Búsqueda del problema	x	x																														
Recopilación de Información		x																														
Desarrollo del Problema			x																													
Presentación de la Propuesta				X																												
Aprobación del proyecto					x	x																										
Diseño de la Solución							x	x	x	x	x																					
Desarrollo del Sistema												x	x	x	x	x																
Prueba del Sistema																	x	x	x													
Elaboración del Primer Borrador del Informe Final																				x	x	x										
Corrección del Primer Borrador																							x	x								
Elaboración del Segundo Borrador del Informe Final																									x	x	x					
Corrección del Segundo Borrador																												x	x			
Elaboración del Informe Final																														x	x	x
Entrega de Informe Final																																