UNIVERSIDAD LASALLISTA BENAVENTE

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

Con estudios incorporados a la
Universidad Nacional Autónoma de México
CLAVE: 8793-16

"SISTEMA DE CONTROL DE MERCANCÍA INCAUTADA Y DISTRIBUIDA"

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

JUAN PEDRO SAN ELÍAS GASCA.

ASESOR: **ING. ALEJANDRO GUZMÁN ZAZUETA.**

Celaya, Gto. **Febrero de 2009**.

A DIOS.

Por darme la vida de la que ahora gozo además de la salud y la capacidad de poder salir adelante cuando algún problema se ha presentado en mi vida y por enviarme a las personas que me rodean y me hacen ser una mejor persona.

MIS PADRES.

Eusebio San Elías Gasca y Avelina Gasca Ávila. A ustedes que sin su apoyo nunca hubiera logrado esto, por enseñarme lo más importante que es vivir la vida con todas sus consecuencias, gracias por todos los sacrificios que hicieron y siguen haciendo para que pueda salir adelante y ser quien hoy en día soy, estoy muy orgulloso de ser su hijo; muchísimas gracias y no olviden que los quiero mucho.

MI ESPOSA E HIJA.

Didi y Camila por haberme apoyado desde que están conmigo para salir adelante y lograr mis metas ya que ustedes son mi motor, y de esta manera poder tener un futuro mejor y también por haber aguantado muchas veces el no convivir con ustedes, muchas gracias por su apoyo.

MIS HERMANAS.

Primero que nada a tí Olivia y tu esposo Jorge Luis, ya que tanto ustedes como mis padres siempre me apoyaron tanto en consejos los cuales no los olvido, así como económicamente en lo referente a la educación y en lo familiar muchas gracias a ustedes y a sus hijos Jorge e Itzel.

Roxana, creo que tú fuiste la que más carencias tuviste para que yo lograra esta meta, pero a veces, aunque lo creas eso era lo que me motivaba a seguir adelante con mis objetivos, gracias chanita.

FAMILIARES.

Gracias a todos por su apoyo incondicional y por creer en que era capaz de lograr esto, por los consejos que me brindaban todos y cada uno de ustedes.

MIS PROFESORES.

Que sin lugar a dudas son una pieza fundamental en la que soy como persona ya que desde el primer maestro que tuve hasta la fecha, espero haber tomado lo mejor de cada uno de ustedes tanto en lo académico como en lo personal. Muchas gracias por brindarme su conocimiento.

MIS AMIGOS.

A todos ustedes que en las buenas y en las malas siempre estuvieron conmigo, al momento de compartir conocimientos sin pedir nada a cambio y simplemente por ser parte de mi vida, pero en especial a ti Rodolfo Macías Arellano† por tu consejos tan sinceros, aunque ya no estás con nosotros siempre serás la persona más alegre y ocurrente que conocí, a todos ustedes gracias por su amistad brindada.

ÍNDICE

INTRODUCCIÓN.

El estudio y aplicación que a continuación se muestra consiste en el desarrollo de un sistema el cual está enfocado en lograr un control específico de las mercancías y vehículos que sean incautados por el Servicio de Administración Tributaria (SAT); debido a que estas no cuentas con los permisos o pagos obligatorios para ser introducidos en el país de una manera legal; cuando se realiza el decomiso todo lo incautado pasa a resguardo del SAT en alguno de los recintos fiscales que tiene en su poder.

Debido a que el SAT ubicado en Celaya, Gto., no cuenta con un sistema para ubicar fácilmente la mercancía en su recinto ó si ha sido enviada a algún otro lugar, donada o destruida, este sistema facilitará a los usuarios el poder localizarla rápidamente e introducir toda la información referente a la mercancía o vehículos introduciendo sus características así como las de sus propietarios.

También contará con un área de registro donde los trabajadores del SAT que requieran de usar este sistema sean registrados de acuerdo al nivel asignado los cuales son Administrador y Usuarios, donde los Administradores tendrán acceso a registrar a los Usuarios, Propietarios, Mercancías y sus Destinos, además tendrán el derecho de realizar consultas sobre los propietarios, mercancías y fecha de decomiso, a todas estas consultas los trabajadores con nivel de usuario también tendrán acceso.

El capítulo primero permitirá conocer a fondo lo que es la estructura interna de SAT federal, así como el Área de Auditoría Fiscal de Celaya y los servicios que este departamento ofrece así como su ubicación, también planteará la problemática con que cuenta al momento de registrar mercancías o vehículos, además de su destino, planteará algunos objetivos motivo por lo cual se realizará el sistema y metodología que se utilizara para lograr el objetivo deseado.

El segundo capítulo tiene la finalidad de dar a conocer todas las generalidades y aplicaciones de SQL, así como una breve explicación de lo que son las bases de datos y su funcionalidad, además conoceremos algunos de los comandos más utilizados por el analizador de consultas de SQL y los componentes del administrador corporativo; también una breve introducción a lo que es visual studio.net, como su estructura y algunas características generales.

El tercero tratará los temas del entorno del visual.net, como todos los apartados del cuadro de herramientas describiendo específicamente algunos de los más importantes como el gridview que es esencial al momento de realizar consultas y muchas otras aplicaciones.

El último capítulo mostrará el diseño y desarrollo del sistema donde el diseño se elaboró derivado de las fallas que actualmente se tiene en la forma de capturar la información, además aquí se planteará la forma y funcionamiento que deberá presentar el sistema al momento de su terminación, teniendo una breve descripción cada una de las páginas del sistema.

CAPÍTULO I
ANTECEDENTES

1.1 Historia

El 15 de diciembre de 1995 se publicó la Ley del Servicio de Administración Tributaria, en la cual se creó el nuevo órgano descentralizado como máxima autoridad fiscal.

El Servicio de Administración Tributaria (SAT)[1] es una sección descentralizada de la Secretaría de Hacienda y Crédito Público (SHCP) que tiene la responsabilidad de aplicar la ley fiscal y aduanera, con el propósito de que las personas físicas[2] y morales[3] contribuyan proporcional y equitativamente al gasto público[4], de calificar a los contribuyentes para que cumplan con las disposiciones tributarias y aduaneras, de facilitar y estimular el cumplimiento voluntario, generar y proporcionar la información necesaria para el diseño y la evaluación de la política tributaria.

Misión

Recaudar las contribuciones federales y controlar la entrada y salida de mercancías del territorio nacional, garantizando la correcta aplicación de la legislación y promoviendo el cumplimiento voluntario y oportuno.

Visión

Ser una institución eficaz y orientada al contribuyente, con procesos integrados, formada por un equipo honesto, profesional y comprometido, al servicio de los mexicanos.

---

[1] **SAT:** por sus siglas, Servicio de Administración Tributaria.
[2] **Persona física:** es un individuo con capacidad para contraer obligaciones y ejercer derechos.
[3] **Persona moral:** es una grupo de personas que se unen con un fin determinado, ejem. una sociedad mercantil, una asociación civil.
[4] **Gasto público:** Es el conjunto de erogaciones que realiza el Gobierno Federal, estatal y municipal incluidos los Poderes Legislativo y Judicial y el sector paraestatal en sus respectivos niveles, en el ejercicio de sus funciones.

1.2 Estructura Orgánica Operativa del SAT.

Esta estructura muestras los diferentes departamentos en los que se divide el SAT a nivel federal; siendo en uno de los subdepartamentos de la Administración General de Auditoría Fiscal Federal, como se verá en el próximo tema, el lugar donde se realizará el sistema.

Servicio de Administración Tributaria.

- Administración General de Aduanas.
- Administración General de Asistencia al Contribuyente.
- Administración General de Auditoría Fiscal Federal.[5]
- Administración General de Comunicaciones y Tecnologías de Información.
- Administración General de Evaluación.
- Administración General de Grandes Contribuyentes.
- Administración General de Innovación y Calidad.
- Administración General de Recaudación.
- Administración General Jurídica.
- Asesoria “1”.
- Departamento de Apoyo Técnico.
- Departamento de Control de Gestión.
- Departamento de la Jefatura del SAT.
- Representante del SAT del tratado de libre comercio de América del Norte.
- Secretaría Particular de la Jefatura del SAT.

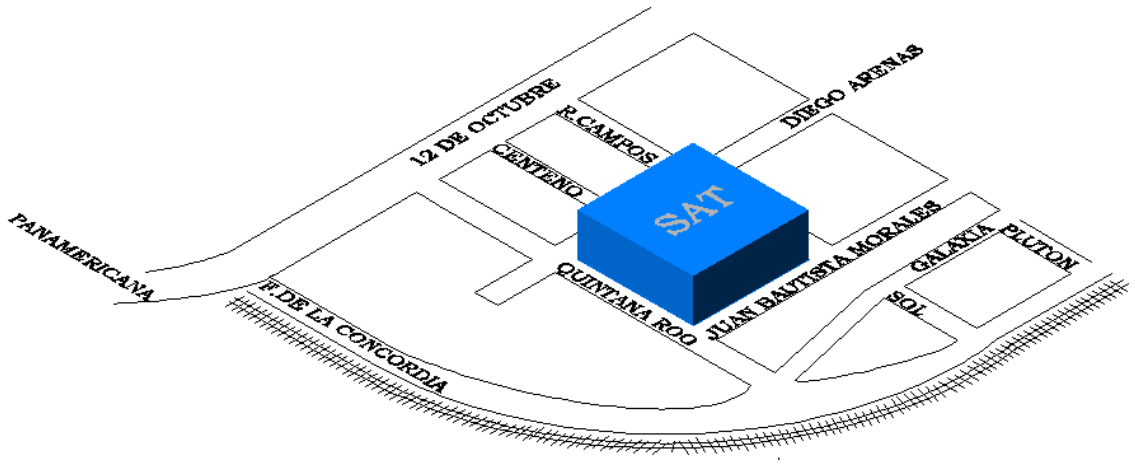
[5] Este Departamento se divide en las diferentes Administraciones Locales de Auditoria a través del país, siendo la Local de Celaya, Gto., donde se elaborara el sistema.

- Servicios Generales de la Jefatura del SAT.
- Titular del Órgano Interno de Control Dependencia Jerárquica y Funcional de la Secretaría de la Función Pública.
- Unidad de Plan Estratégico y Mejora Continua.

1.3 Ubicación del SAT.

Las oficinas del SAT de Celaya, Gto., se encuentran ubicadas en la calle Juan Bautista Morales No. 200, en la Colonia Zona de Oro I, siendo este su domicilio oficial, además cuenta con otros 2 accesos en las calles de Diego Arenas y R. Campos, véase figura1.1

Figura 1.1: Ubicación del SAT de Celaya.

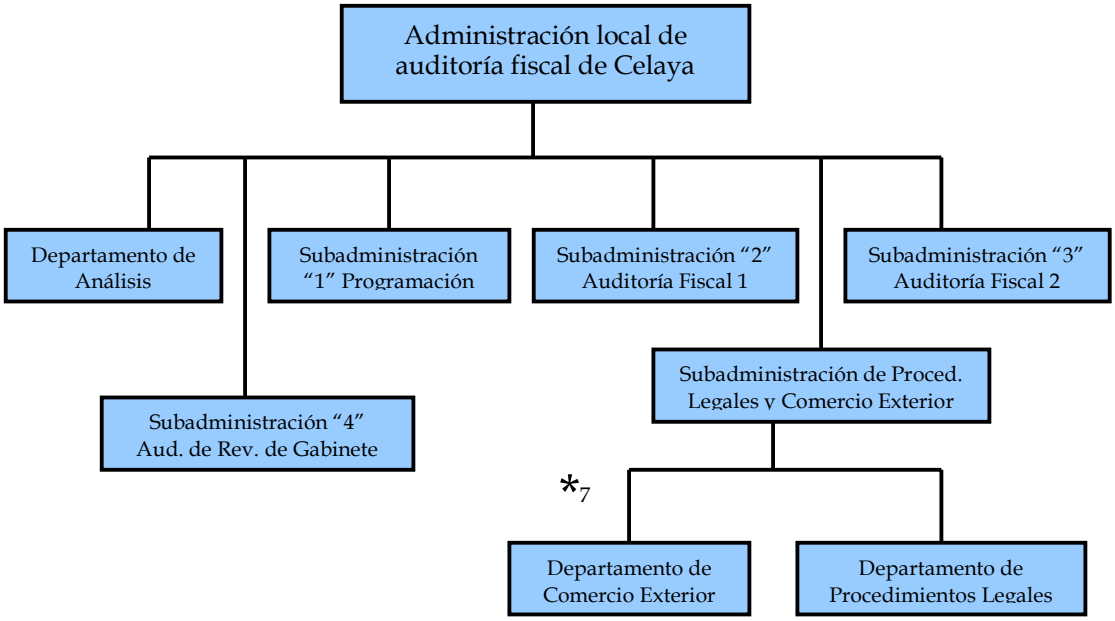


Fuente: Imagen creada por el Autor.

1.4 Organigrama de la ALAF de Celaya.

El organigrama siguiente muestra la estructura de la Administración Local de Auditoría Fiscal de Celaya (ALAF) [6] y sus diferentes departamentos; el departamento de Comercio Exterior es el lugar donde se implementará este sistema, véase figura 1.2.

Figura 1.2: Organigrama de la Administración Local de Auditoría Fiscal de Celaya.



Fuente: Diagrama creado por el Autor.

[6] Administración Local de Auditoria Fiscal.
[7] Departamento donde se realizara el sistema.

1.5 Actividades y Servicios de la ALAF Celaya.

Dentro del departamento de Comercio Exterior se tienen diferentes actividades y servicios [8] como las de realizar visitas domiciliarias, Auditorías, inspecciones, actos de vigilancia, verificaciones y demás actos que establezcan las disposiciones fiscales y aduaneras, para comprobar el cumplimiento de los todos contribuyentes, incluyendo los que realicen entradas y salidas de mercancías y medios de transporte del país, siendo algunas de las más importantes las siguientes:

- Solicitar de los contribuyentes, responsables solidarios o terceros, datos, informes o documentos, para planear y programar actos de fiscalización.

- Ordenar y realizar la verificación de mercancías de comercio exterior en transporte, de vehículos de procedencia extranjera en tránsito, de aeronaves y embarcaciones, practicar actos de vigilancia para asegurar el cumplimiento de las disposiciones legales para la entrada al país y la salida del mismo de mercancías y medios de transporte.

- Inspeccionar y vigilar los lugares fiscales y fiscalizados, en este último caso, vigilar el cumplimiento de las obligaciones derivadas de la autorización permitida para prestar los servicios del manejo, almacenaje y la custodia de las mercancías de comercio exterior, habilitar instalaciones como espacios fiscales para uso de la

[8] Decreto por el que se expide el reglamento interior del Servicio de Administración Tributaria y se modifica el reglamento interior de la secretaria de hacienda y crédito publico.

autoridad fiscal y aduanera, así como declarar el abandono de las mercancías que se encuentren en los patios y recintos fiscales bajo su responsabilidad.

- Ordenar y practicar visitas domiciliarias a los contribuyentes, para verificar el cumplimiento de las obligaciones fiscales relacionadas con la expedición de comprobantes fiscales, así como solicitar la exhibición de los comprobantes que amparen la legal posesión o propiedad de los bienes y mercancías que vendan.

- Realizar la clausura preventiva de los establecimientos de los contribuyentes por no expedir o no entregar comprobantes de sus actividades; que los comprobantes no reúnan requisitos fiscales o que los datos asentados en el comprobante correspondan a persona distinta a la que adquiere la mercancía.

- Ordenar y realizar la custodia, persecución, embargo precautorio de mercancías de comercio exterior y medios de transporte, como lo establece la Ley Aduanera, llevar acabo el aseguramiento del embargo y la entrega de las mercancías embargadas, antes de la conclusión del procedimiento de que se trate, previa resolución y aceptación de la garantía del interés fiscal, declarar que las mercancías, vehículos, embarcaciones o aeronaves pasan a propiedad del Fisco Federal, liberar las garantías permitidas respecto de la posible omisión del pago de contribuciones en mercancías sujetas a precios estimados, notificar a las autoridades del país de procedencia la localización de vehículos o

aeronaves robados u objeto de disposición ilícita, así como decidir su devolución y del cobro de los gastos que se hayan autorizado.

- Vigilar la destrucción, donación o traslado de mercancías según sea el veredicto de la resolución.

En el tercer punto está basada la investigación para desarrollar el sistema y de esta forma erradicar la problemática que a continuación se describe.

1.6 Problema a Resolver.

Actualmente cualquier empresa tiene la necesidad de tener el total control de los productos o mercancías que se adquieran, por tal motivo es esencial que esta organización cuente con un sistema de control de mercancía incautada y distribuida, para lograr esto se desarrollara el SACM[9] que como lo dice su nombre permitirá un absoluto control de estas mercancías.

En la actualidad me encuentro laborando dentro de esta organización federal, la cual tiene ingresos considerables de mercancía, que no se encuentran en venta, si no que son destruidas, donadas o entregadas a otro organismo federal.

[9] Sistema Absoluto de Control de Mercancías.

Durante el periodo de tiempo en que se decide el destino de éstas, se tienen en resguardo en bodega o recinto fiscal donde no se cuenta con un sistema, que permita la fácil descripción, ubicación, acceso y destino de ellas, ya que en estos momentos se realiza de forma manual o en simples tablas, de manera que cuando se realiza el embargo precautorio la forma en que se realiza el inventario, las mercancías se van anotando a mano en hojas cuadriculadas describiendo las características de cada una de estas, posteriormente se le asigna un espacio en la bodega o recinto fiscal al azar hasta que se tenga una resolución fiscal, una vez realizado esto, dicha información se captura en tablas de Excel, siendo esta la manera en que se conocen las descripciones, cantidades y ubicación de las mercancías incautadas, dichas acciones lo que producen son pérdidas considerables de tiempo al momento de tratar de ubicarlas e identificarlas por sus diferentes características, además algunas veces se traspapelan los documentos y se desconoce el paradero de la mercancía, generando graves problemas para los encargados del almacén.

1.7 Justificación.

El propósito de este proyecto consiste en desarrollar un sistema de información que permita al usuario lograr una mejor descripción y ubicación de las mercancías desde cualquier lugar, logrando así una gran disminución de tiempo al momento de ubicarlas, tanto en su descripción como en su localización dentro del almacén, además se lograría conocer su destino cuando se resuelva su situación legal, lo cual sería de gran utilidad y de esta forma se lograrían evitar problemas de localización, debido a que muchas de las mercancías son similares en

cuanto a marcas, modelos, colores, etc. y con este sistema se ubicarían e identificarían de manera eficaz.

1.8 Objetivos

1.8.1 Objetivo General.

Tener un acceso rápido, eficaz y confiable de la información que se solicite sobre las mercancías, las cuales se han obtenido a lo largo del tiempo y se obtendrán en un tiempo futuro, además de brindar mejoras para su captura en la base de datos que se creara, actualizándose la información en el sitio Web instantáneamente y de esta forma facilitar el trabajo a los usuarios.

1.8.2 Objetivos Específicos.

- Mostrar qué son las bases de datos dando a conocer sus generalidades y características y cómo utilizarlas.
- Dar a conocer las aplicaciones de Visual Studio para comprender su aplicación.
- Desarrollar el conocimiento adecuado del funcionamiento del sistema para elaborar un análisis y diseño de una base de datos.
- Diseñar una base de datos que contenga la información sobre las mercancías incautadas.
- Desarrollar el diseño de páginas Web, para poder lograr su implementación.
- Tener una lista de la mercancía incautada en el menor tiempo posible, la cual describa esta de la mejor manera.

- Tener de forma accesible la ubicación exacta de cada una de las mercancías dentro del almacén, para su rápido ubicación.
- Tener conocimiento de cual fue el destino de alguna mercancía después de que se determine su situación.

1.9 METODOLOGIA A UTILIZAR.

La metodología que se utilizará para lograr el sistema mencionado es conocida como, **modelo en cascada**, mantiene de una forma exacta las etapas de la creación del software, de manera que el inicio de cada etapa debe esperar la terminación de la anterior y de esta manera eliminar la mayor parte de errores posibles en cada fase.

Pasos de la metodología de modelo en cascada es:

- **Análisis de requerimientos.-** Se analizan las necesidades que los usuarios finales del software requieren para cumplir sus objetivos, en esta parte se deberá estar de acuerdo en lo que requiere el sistema.
- **Diseño del Sistema.-** Aquí se separa y organiza el sistema en elementos que puedan realizarse por partes, además contiene la descripción de la estructura relacional del sistema y las especificaciones de lo que debe hacer cada una de las partes, así como la forma en que se combinan unas con otras.
- **Diseño del Programa.-** Es la fase en donde se realizan los algoritmos para el cumplimiento de los requerimientos del usuario

así como el análisis necesario para saber que herramientas usar en la etapa de codificación.

- **Codificación.-** Es la fase de programación donde se implementa el código fuente, haciendo uso de prototipos, pruebas y ensayos para corregir errores.
- **Pruebas.-** Las partes ya programadas, se juntan para armar el sistema y se comprueba que funciona antes de ser utilizado.
- **Implantación.-** Se establecen los niveles software y hardware que componen el proyecto, es la parte con más duración y con más cambios en la elaboración de un proyecto.
- **Mantenimiento.-** Es donde se realizan cambios al sistema, además de que se corrigen averías y errores cometidos por los usuarios.

Ventajas

- Se tiene todo bien organizado y no se mezclan las fases.
- Es ideal para proyectos que son rígidos, y además e especifiquen muy bien los requerimientos y se conoce muy bien las herramientas a utilizar.

EL siguiente diagrama muestra el funcionamiento del modelo de cascada del ciclo de vida de un software, véase figura 1.3.

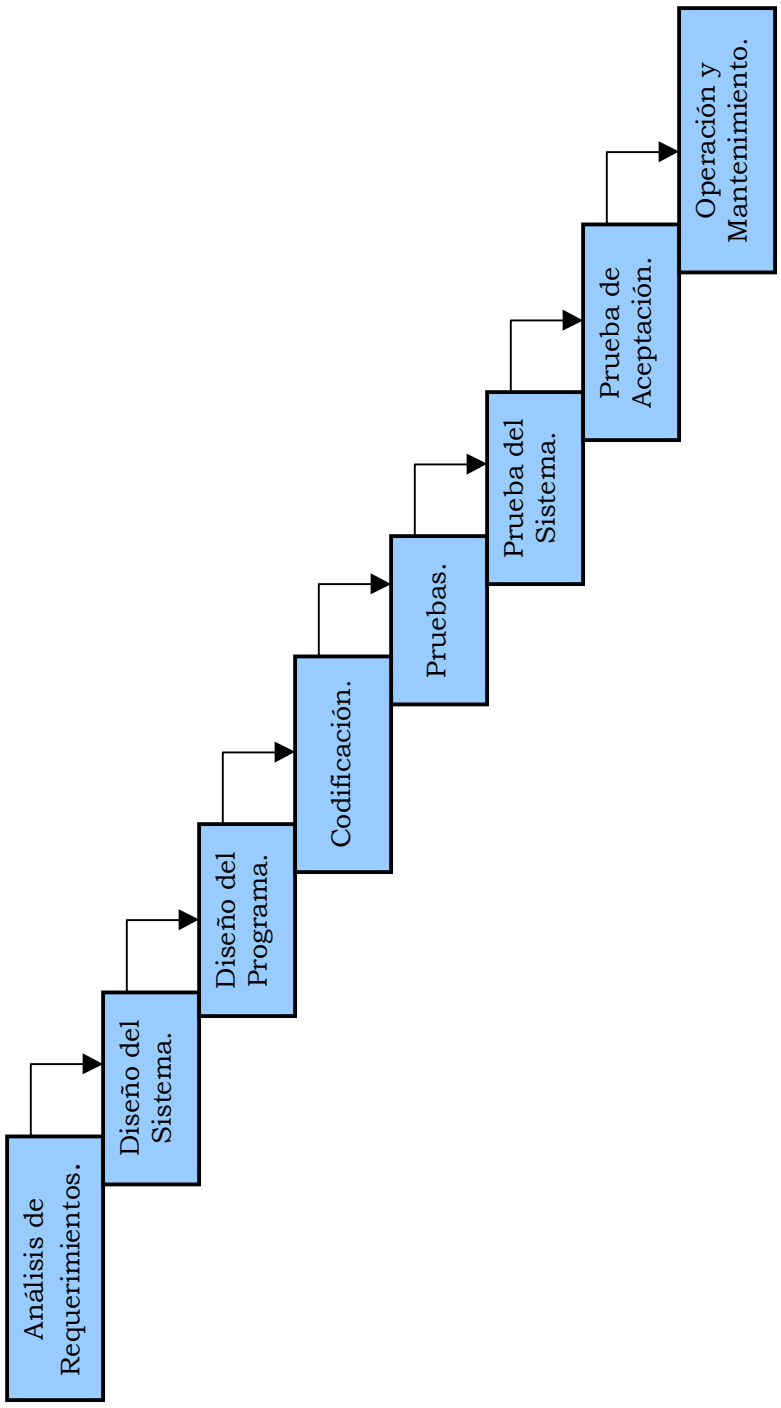


Figura 1.3: Diagrama del Modelo Cascada.

Fuente: Diagrama creado por el Autor.

CAPÍTULO II
GENERALIDADES Y APLICACIONES DE SQL Y VISUAL STUDIO.

2.1 Requisitos del Sistema.

Los requerimientos de software pueden variar, se puede utilizar para el desarrollo de aplicaciones Web: Visual Studio.NET en sus diferentes versiones, Visual Basic.NET Standard Edition o ASP.NET; así como también tenemos diferentes software para la creación de bases de datos como lo son: Acces, SQL-Server en sus diferentes versiones, MySQL y Oracle que son de los más comunes; este sistema estará desarrollado con Visual Studio.NET y SQL-Server.

2.1.1 Software requerido para desarrollar aplicaciones Visual Studio.NET.

- Windows Server 2003, Windows XP Profesional (Recomendado) ó Windows Vista, en todas sus versiones.
- Internet Explorer 5.0, ó superior.
- Microsoft SQL Server 2000.
- Visual Studio.NET 2005, en cualquiera de sus versiones.

2.1.2 Hardware Requerido para desarrollar aplicaciones Visual Studio.NET.

- **Procesador:** Cualquiera de 1 GHZ o superior, en este caso se utilizara un AMD Sempron a 2 GHZ.
- **Memoria Ram:** 1024 MB, pero se recomienda algo más superior.
- Unidad de CD-ROM o DVD-ROM.
- Súper VGA (1024x768) ó mayor resolución con 256 colores.
- Mouse.

2.2 Que es una Base de Datos.

En una empresa o negocio tenemos diferentes tipos de información de manera desordenada, y al momento de necesitarla no se es posible tener acceso rápido a esta información, debido a que no se encuentra ordenada, por este motivo fueron creadas las bases de datos que se definen como:

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

Una base de datos es un archivo como lo es un documento de texto o una imagen. La diferencia es que la base de datos tiene una estructura interna mucho más compleja que cualquier otro, ya que a diferencia de los demás, cuenta con medidas de seguridad para resguardar los datos y sus cambios.

2.2.1 Componentes principales de una base de datos.

- **Datos.** Son la Base de Datos propiamente dicha.
- **Hardware.** Se refiere a los dispositivos de almacenamiento en donde reside la base de datos, así como a los dispositivos periféricos (unidad de control, canales de comunicación, etc.) necesarios para su uso.
- **Software.** Está constituido por un conjunto de programas que se conoce como Sistema Manejador de Base de Datos (SMBD). Este

sistema maneja todas las solicitudes formuladas por los usuarios a la base de datos.

- **Usuarios.** Existen dos clases de usuarios relacionados con una Base de Datos:

    - El programador de aplicaciones: Quien crea programas de aplicación que utilizan la base de datos.
    - El usuario final: Quien accede a la Base de Datos por medio de un lenguaje de consulta o de programas de aplicación.

2.2.2 Ventajas de las bases de datos.

- Globalización de la información: permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.
- Eliminación de información inconsistente: si existen dos o más archivos con la misma información, los cambios que se hagan a éstos deberán hacerse a todas las copias del archivo de facturas.
- Permite compartir información
- Permite mantener la integridad en la información.
- Independencia de datos: es la separación entre programas y datos.
- Restricciones de seguridad.
- Reduce redundancia: Acciones lógicamente únicas.

2.3 Introducción al SQL-Server.

SQL Server 2000 es un sistema de gestión de bases de datos relacionales (SGBDR) diseñado para trabajar con grandes cantidades de información y con la capacidad de cumplir con los requerimientos de proceso de información para aplicaciones comerciales y sitios Web.

También ofrece el soporte de información para las tradicionales aplicaciones Cliente/Servidor, las cuales están conformadas por una interfaz a través de la cual los clientes pueden acceder a los datos por medio de una red.

La plataforma NET exige un gran porcentaje de distribución de recursos y un entorno descentralizado, para ello sus clientes deben ser livianos, tales como los navegadores de Internet los cuales accederán a los datos por medio de servicios como el Internet Information Services (IIS).

SQL Server 2000 está diseñado para trabajar con dos tipos de bases de datos:

• **OLTP (OnLine Transaction Processing)** Son bases de datos caracterizadas por mantener una gran cantidad de usuarios conectados concurrentemente realizando ingreso y/o modificación de datos. Por ejemplo: entrada de pedidos en línea, inventario, contabilidad o facturación.

• **OLAP (OnLine Analytical Processing)** Son bases de datos que almacenan grandes cantidades de datos que sirven para la toma de decisiones, como por ejemplo las aplicaciones de análisis de ventas.

Los entornos Cliente/Servidor, están realizados manera que la información se guarde de forma centralizada en una computadora central llamada Servidor**,** siendo este el responsable del mantenimiento de la relación entre los datos, asegurándose del correcto almacenamiento de los datos, establecer restricciones que controlen la integridad de datos, etc.

Por el lado del cliente, este corre típicamente en distintas computadoras las cuales acceden al servidor a través de una aplicación, para realizar la solicitud de datos los clientes emplean el **Structured Query Language (SQL)**, este lenguaje tiene un conjunto de comandos que permiten especificar la información que se desea recuperar o modificar.

Existen muchas formas de organizar la información pero una de las formas más efectivas de hacerlo está representada por las **bases de datos relacionales**, donde los datos están organizados en tablas.

Una tabla representa una clase de objeto que tiene importancia para una organización. Por ejemplo, se puede tener una base de datos con una tabla para empleados, otra para clientes y otra para productos el almacén. Las tablas están compuestas de columnas y filas.

Al organizar los datos en tablas, se pueden encontrar varias formas de definirlas. La teoría de las bases de datos relacionales define un proceso, la normalización, que asegura que el conjunto de tablas definido organizará los datos de manera eficaz.

2.4 Que es y para que sirve el SQL.

Debido a la gran variedad de lenguajes y de bases de datos existentes, la manera de comunicarse entre estos sería realmente complicada de no ser por la existencia de estándares que nos permiten el realizar las operaciones básicas de una forma universal.

Es de eso de lo que trata el SQL que no es más que un lenguaje estándar de comunicación con bases de datos. Por tanto de un lenguaje normalizado que permite trabajar con cualquier tipo de lenguaje, en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL, etc.).

El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. Cada base de datos implementa funciones específicas que no tienen necesariamente que funcionar en otras. Aparte de esto, el SQL posee otras dos características muy apreciadas. Por una parte, presenta una potencia y versatilidad notables que contrasta, por otra, con su accesibilidad de aprendizaje.

2.4.1 Tipos de Campo.

Una base de datos esta compuesta de tablas donde se almacenan registros catalogados en función de distintos campos (características). Un aspecto previo a considerar es la naturaleza de los valores que se introducen en esos campos. Dado que una base de datos trabaja con todo tipo de informaciones, es importante especificar qué tipo de valor se

le está introduciendo, por un lado, facilitar la búsqueda posteriormente y por otro, optimizar los recursos de memoria.

Cada base de datos introduce tipos de datos de campo que no necesariamente están presentes en otras. Sin embargo, existe un conjunto de tipos que están representados en la totalidad de estas bases. Estos tipos comunes son los siguientes: véase figura 2.1

Figura 2.1: Tipos de Campos

Tipo.	Descripción.
Alfanuméricos	Contienen números y letras contienen una capacidad de 255 caracteres.
Numéricos	Existen de varios tipos, principalmente, enteros (sin decimales) y reales (con decimales).
Voléanos	Poseen dos formas: Verdadero y falso (Sí o No).
Fechas	Almacenan fechas facilitando la posibilidad de ordenar los registros por fechas o calcular los días entre una fecha y otra.
Memos	Son campos alfanuméricos de longitud ilimitada.
Autoincrementables	Son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta más que evidente: Servir de clave ya que resultan exclusivos de un registro.

Fuente: Tabla creada por el Autor.

2.4.2 Comandos.

Existes dos tipos de comandos en el lenguaje SQL:

- **DLL (Lenguaje de Definición de Datos).-** Se puede crear y definir nuevas bases de datos, campos e índices, véase figura 2.2.
- **DML (Lenguaje de Manipulación de Datos).-** Se puede generar consultas para ordenar, filtrar y extraer datos de la base de datos, véase figura 2.3.

Figura 2.2: Comandos DLL.

Comando.	Descripción.
Create	Utilizado para crear nuevas tablas, campos e índices.
Drop	Empleado para eliminar tablas e índices.
Alter	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Fuente: Tabla creada por el Autor.

Figura 2.3: Comandos DML.

Comando.	Descripción.
Select	Se utiliza para consultar los registros.
Insert	Se utiliza para insertar los datos a la base de datos.
Update	Se utiliza para modificar los datos que se encuentren en la base de datos.
Delete	Se utiliza para borrar los datos de los registros en la base de datos.

Fuente: Tabla creada por el Autor.

2.4.3 Cláusulas.

Son condiciones de modificación, utilizadas para definir los datos que se desee seleccionar o modificar, véase figura 2.4.

Figura 2.4: Cláusulas

Cláusula.	Descripción.
From	Especifica la tabla de la cual se van a seleccionar los registros.
Where	Especifica las condiciones que deben reunir los registros que se va a seleccionar.
Group by	Se utiliza para separar los registros seleccionados en grupos específicos.
Order by	Se utiliza para ordenar los registros seleccionados de acuerdo a un orden específico.
Having	Se utiliza para expresar la condición que se debe da satisfacer.

Fuente: Tabla creada por el Autor.

2.4.4 Operadores Lógicos.

Los operadores lógicos comprueban la veracidad de alguna condición, y devuelven el tipo de datos **Boolean** con el valor TRUE, FALSE o UNKNOWN, véase figura 2.5.

Figura 2.5: Operadores Lógicos.

Operador.	Descripción.
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad, solo si las dos son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad, si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Fuente: Tabla creada por el autor.

2.4.5 Operadores de Comparación.

Los operadores de comparación comprueban si dos expresiones son iguales. Se pueden utilizar en todas las expresiones excepto en las de los tipos de datos **text**, **ntext** o **image**, véase figura 2.6.

Figura 2.6: Operadores de Comparación.

Operador.	Descripción.
<	Menor que.
>	Mayor que.
<>	Diferente de.
<=	Menor o igual que.
>=	Mayor o igual que.
=	Igual.
Between	Especifica un intervalo de valores.
Like	Se utiliza en la comparación de un modelo.
In	Se utiliza para especificar registros de una base de datos.

Fuente: Tabla creada por el Autor.

2.4.6 Funciones de Agregado.

Se utilizan dentro de una cláusula **select** en grupos de registros para devolver un único valor que se aplica a un grupo de registros, véase figura 2.7.

Figura 2.7: Funciones de Agregado.

Función.	Descripción.
AVG	Se utiliza para calcular el promedio de los valores de un campo.
COUNT	Se utiliza para devolver el número de registros de la selección.
SUM	Se utiliza para devolver la suma de todos los valores de un campo.
MAX	Se utiliza para devolver el valor más alto de un campo.
MIN	Se utiliza para devolver el valor más bajo de un campo.

Fuente: Tabla Creada por el Autor.

2.4.7 Orden de Ejecución de Comandos.

En una sentencia SQL de selección que incluye todas las posibles cláusulas, el orden de ejecución debe ser el siguiente:

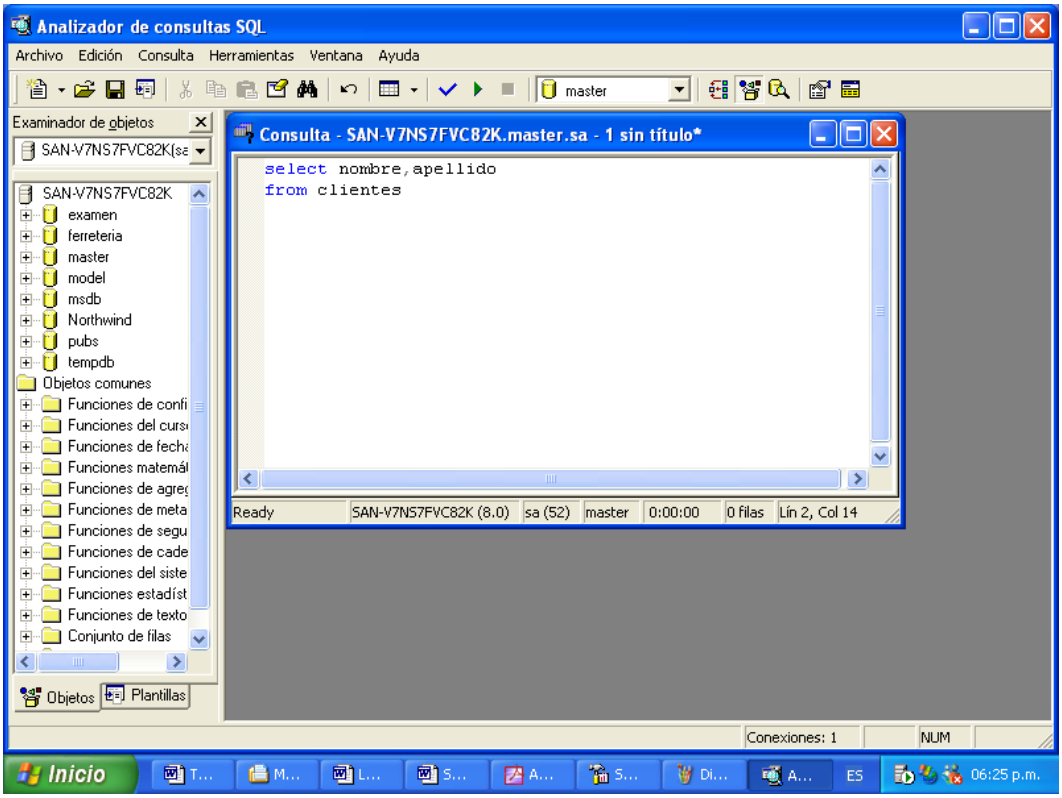
- Cláusula FROM.
- Cláusula WHERE.
- Cláusula GROUP BY.
- Cláusula HAVING.
- Cláusula SELECT.
- Cláusula ORDER BY.

2.5 Consultas de Selección.

Se utilizan para indicar al motor de datos que devuelva información de las bases de datos, ésta es devuelta en forma de conjunto de registros que se pueden almacenar en un objeto recordset[1], el cual se puede modificar.

Todas las consultas se deben realizar en el analizador de consultas que es parte del software de SQL-Server, véase figura 2.8.

Figura 2.8: Analizador de Consultas.



Fuente: Imagen creada por el Autor.

[1] Es una tabla que almacena el resultado de las consultas realizadas a la base de datos a la que nos conectemos.

Para que el analizador de consultas funcione correctamente el administrador de servicios se debe estar ejecutando correctamente, véase figura 2.8.1.

Figura 2.8.1: Administrador de Servicios.

Fuente: Imagen Creada por el Autor.

2.5.1 Consultas Básicas.

La sintaxis básica de una consulta de selección es la siguiente:

SELECT *Campos* **FROM** *Tabla*

Donde la lista de campos que se quieran recuperar y la tabla es el origen, por ejemplo:

SELECT *Nombre, Teléfono* **FROM** *Clientes*

Aquí devuelve el conjunto de resultados con los campos: nombre y teléfono de la tabla clientes.

2.5.2 Ordenar los registros.

Se pude poner el orden en el que se deseen mostrar los registros de las tablas mediante la cláusula **ORDER BY** y la lista de campos que se desea ordenar:

SELECT *Nombre, Teléfono* **FROM** *Clientes* **ORDER BY** *Nombre*

Este ejemplo devolverá una lista con el nombre y teléfono de la tabla clientes pera estará ordenada de acuerdo al nombre.

2.5.3 Consultas con predicado.

Se incluye entre la cláusula y el primer nombre del campo a recuperar, véase figura 2.9.

Figura 2.9: Predicados.

Predicado.	Descripción.
ALL	Devuelve todos los campos de la tabla.
TOP	Devuelve un determinado número de registros de la tabla.
DISTINCT	Omite los registros cuyos campos seleccionados coincidan totalmente.
DISTINCTOW	Omite los registros duplicados basándose en la totalidad del registro y no solo en los campos seleccionados.

Fuente: Tabla creada por el Autor.

2.5.4 Recuperar Información de una Base de Datos Externa.

En ocasiones es necesario la recuperación de información que se encuentra contenida en una tabla que no se encuentra en la base de datos que se ejecutará la consulta ésta se puede salvar con la palabra reservada **IN** de la siguiente manera:

SELECT Apellido as Empleado **FROM** Empleados
IN C:\databases\empleados.mdb

2.5.5 Consultas de Acción.

2.5.5.1 DELETE.

Crea una consulta de eliminación que elimina los registros de una o más tablas listadas en la cláusula **FROM** que satisfagan la cláusula **WHERE** esta consulta elimina los registros completos, su sintaxis es:

DELETE FROM *Tabla* **WHERE** *criterio*

Una vez borrado el registro no se puede deshacer la operación, hay que tener cuidado con esta instrucción ya que si no especificamos una condición con **WHERE**, lo que hacemos es **borrar toda la tabla**.

2.5.5.2 INSERT INTO.

Agrega un registro a la tabla. Esta consulta puede ser de dos formas:

- Insertar un único registro.
- Insertar en una tabla los registros contenidos en otra tabla.

La sintaxis utilizada cuando se inserta un único registro es la siguiente:

Insert Into *nombre_tabla (nombre_campo1, nombre_campo2,...)* **Values** *(valor_campo1, valor_campo2...)*

Un ejemplo podría ser:

Insert Into *clientes (nombre, apellidos, dirección, población, código postal, email, pedidos)* **Values** *('Camila', 'Gasca', 'Morelos n°13', 'Celaya', '123456', 'camila@hotmail.com', 33).*

En los campos numéricos o se eliminan los apóstrofos (‘ ’).

Si no insertamos uno de los campos en la base de datos se inicializará con el valor por default que hayamos dado a la hora de crear la tabla. Si no hay valor por default, probablemente se inicialice como NULL (vacío), en caso de que este campo permita valores nulos. Si ese campo no permite valores nulos (eso se define también al crear la tabla) lo más seguro es que la ejecución de la sentencia SQL nos de un error.

Para Insertar en una tabla los registros contenidos en otra tabla se realiza lo siguiente:

INSERT INTO *Tabla* **SELECT** *Tabla origen* ***** **FROM** *Tabla origen.*

De esta forma los campos de la tabla origen se grabarán en la tabla para que esto ocurra de forma adecuada es necesario que todos los campos de la tabla origen sean idénticos a la tabla destino.

2.5.5.3 UPDATE.

Update es la instrucción que nos sirve para modificar nuestros registros. Como para el caso de **Delete**, necesitamos especificar por medio de **Where** cuáles son los registros en los que queremos hacer efectivas nuestras modificaciones. Además, tendremos que especificar cuáles son los nuevos valores de los campos que deseamos actualizar.
La sintaxis es de este tipo:

Update *nombre_tabla* **Set** *nombre_campo1 = valor_campo1, nombre_campo2 = valor_campo2,...* **Where** *condiciones_de_selección*

Un ejemplo es:

Update *clientes* **Set** *nombre='José'* **Where** *nombre='Pepe'*

Mediante esta sentencia cambiamos el nombre Pepe por el de José en todos los registros cuyo nombre sea Pepe.

Update es útil cuando se desea cambiar gran número de registros, se pueden cambiar varios campos a la vez.

Update no genera ningún resultado, para saber que registros van a cambiar hay que ver primero el resultado de la consulta que utilice el mismo criterio y después ejecutar la consulta de actualización.

Si en la consulta de actualización no se pone la palabra **Where** todos los registros de la tabla serán actualizados.

2.5.6 Consultas de Combinación entre Tablas.

Las consultas entre tablas se realizan mediante la cláusula **INNER** que combina los registros de dos tablas siempre y cuando concuerden los valores en un campo común, su sintaxis es:

SELECT campos **From** Tb1 **INNER JOIN** Tb2 **ON** Tb1.campo1 **COMP** Tb2.campo2

Donde:

Tb1, Tb2	Nombre de las tablas.
Campo1, Campo2	Nombre de los campos, sin no son numéricos, los campos deben ser del mismo tipo de datos.
Comp	Es cualquier operador de comparación

Fuente: Tabla creada por el Autor.

2.5.7 Estructuras de las Tablas.

Una base de datos en un sistema relacional está compuesta por un conjunto de tablas que corresponden a las relaciones del modelo relacional.

2.5.7.1 CREACIÓN DE TABLAS NUEVAS.

El siguiente código es el que se utiliza para crear una tabla con sus diferentes campos en el analizador de consultas.

```
Create table Producto (
Claveprod int not null,
Precio float,
Existencia int null,
Nomproducto varchar (60),
Codbarras int null,
Constraint pr primary key (Claveprod)
)
```

La siguiente tabla muestra el nombre de la taba y sus diferentes campos, así como el tipo de datos que estos son, véase figura 2.10.

Figura 2.10: Descripción de la tabla.

Nombre	Descripción
Producto	Es el nombre de la tabla.
Claveprod	Es la clave del producto, es de tipo entero y no acepta valores nulos.
Precio	Su nombre lo dice y es de tipo flota.
Nomproducto	Es el nombre del producto de tipo varchar con un máximo de caracteres de 60.
Existencia	Existencia del producto es de tipo entero y no acepta valores nulos.
Codbarras	Es el código de barras y es de tipo entero y no acepta valores nulos.

Fuente: Tabla creada por el Autor.

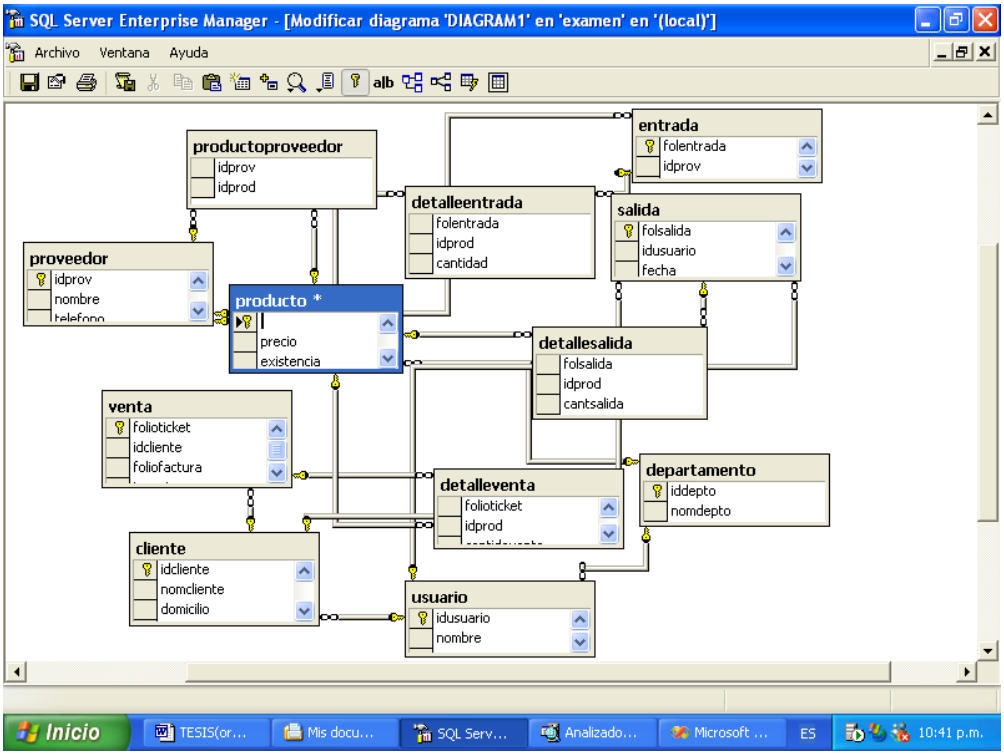
Se utiliza la cláusula **CONSTRAINT** para crear o eliminar índices y sus tipos pueden ser de tres maneras, véase figura 2.11.

Figura 2.11: Tipos de Constraint.

Índice	Descripción
Primary Key	Genera un índice primario el campo o los campos especificados, deben ser únicos y no nulos, cada tabla únicamente debe tener una clave principal.
Foreign key	Genera un índice externo.
Unique	Genera un índice de clave única.

Fuente: Tabla creada por el Autor.

Este es un ejemplo de diagrama de una base de datos, figura 2.12:



Fuente: Imagen Creada por el Autor.

2.6 HISTORIA DE VISUAL STUDIO .NET.

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 español 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 de la máquina virtual Java.

La mejora 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.

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[2]). El SDK[3] 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

[2] Primer microprocesador de la arquitectura Intel Itanium (antes llamada IA64, creada por Hewlett-Packard y desarrollada conjuntamente por HP e Intel)
[3] kit de desarrollo de software: Es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema bastante concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, ordenadores, videoconsolas, sistemas operativos, etcétera.

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.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft, 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.

2.7 INTRODUCCIÓN A LA PLATAFORMA .NET.

Simplificando mucho las cosas para poder dar una definición corta y comprensible, se podría decir que la **plataforma .NET** es un amplio conjunto de bibliotecas de desarrollo, que pueden ser utilizadas por otras aplicaciones para acelerar enormemente el desarrollo y obtener de manera automática características de seguridad, rendimiento, etc…

En realidad .NET es mucho más que eso ya que ofrece un entorno encargado de la ejecución de aplicaciones, nuevos lenguajes de programación y compiladores, y permite el desarrollo de todo tipo de funcionalidades: desde programas de consola o servicios Windows hasta aplicaciones para dispositivos móviles, pasando por desarrollos de escritorio o para Internet.

2.7.1 El entorno de ejecución CLR.

.NET ofrece un entorno de ejecución para sus aplicaciones conocido como **Common Language Runtime** o **CLR**. La CLR es la implementación de Microsoft de un estándar llamado **Common Language Infrastructure** o **CLI**. Éste fue creado y promovido por la propia Microsoft pero desde hace años es un estándar reconocido mundialmente por el ECMA[4].

El CLR/CLI define un entorno de ejecución virtual independiente en el que trabajan las aplicaciones escritas con cualquier lenguaje .NET. Este entorno virtual se ocupa de multitud de cosas importantes para una aplicación: desde la gestión de la memoria y la vida de los objetos hasta la seguridad y la gestión de subprocesos.

Todos estos servicios unidos a su independencia respecto a arquitecturas computacionales convierten la CLR en una herramienta extraordinariamente útil puesto que, en teoría, cualquier aplicación

[4] **Ecma International:** es una organización internacional basada en membresías de estándares para la comunicación y la información

escrita para funcionar según la CLI puede ejecutarse en cualquier tipo de arquitectura de hardware.

2.7.2 El Lenguaje Intermedio y el CLS[5].

Al contrario de otros entornos, la plataforma .NET no está atada a un determinado lenguaje de programación ni favorece a uno determinado frente a otros. En la actualidad existen implementaciones para varias decenas de lenguajes que permiten escribir aplicaciones para la plataforma .NET. Los más conocidos son **Visual Basic .NET**, **C#** o **J#**, pero existen implementaciones de todo tipo.

Lo mejor de todo es que cualquier componente creado con uno de estos lenguajes puede ser utilizado de forma transparente desde cualquier otro lenguaje .NET.

¿Cómo se consigue esta potente capacidad?

Dentro de la CLI, existe un lenguaje llamado **IL** (Lenguaje Intermedio) que está pensado de forma independiente al procesador en el que se vaya a ejecutar. Cuando se compila una aplicación escrita en un lenguaje .NET cualquiera, el compilador lo que genera en realidad es un nuevo código escrito en este lenguaje intermedio. Así, todos los lenguajes .NET se usan como capa de más alto nivel para producir código IL.

[5] **CLS**.- Especificación Común de Lenguajes.

La especificación común de los lenguajes y el sistema de tipos comunes.

Para conseguir la operación entre lenguajes no sólo se utiliza el lenguaje intermedio, sino que es necesario definir un conjunto de características que todos los lenguajes deben incorporar. A este conjunto regulador se le denomina **Common Language Specification** (**CLS**).

Entre las cosas que regula la **CLS** se encuentran la nomenclatura, la forma de definir los miembros de los objetos, los metadatos[6] de las aplicaciones, etc. Una de las partes más importantes de la CLS es la que se refiere a los tipos de datos.

La **CLS** define un conjunto de tipos de datos comunes (**Common Type System** o **CTS**) que indica qué tipos de datos se pueden manejar, cómo se declaran y se utilizan éstos y de qué manera se deben gestionar durante la ejecución.

Si las bibliotecas de código utilizan en sus interfaces hacia el exterior datos definidos dentro de la **CTS** no existirán problemas a la hora de utilizarlos desde cualquier otro código escrito en la plataforma .NET.

Cada lenguaje .NET utiliza una sintaxis diferente para cada tipo de datos, por ejemplo, el tipo correspondiente a un número entero de 32 bits se denomina **Integer** en Visual Basic .NET, pero se llama **Int** en C#.

[6] Los metadatos son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos.

2.7.3 Biblioteca de Clases de .NET.

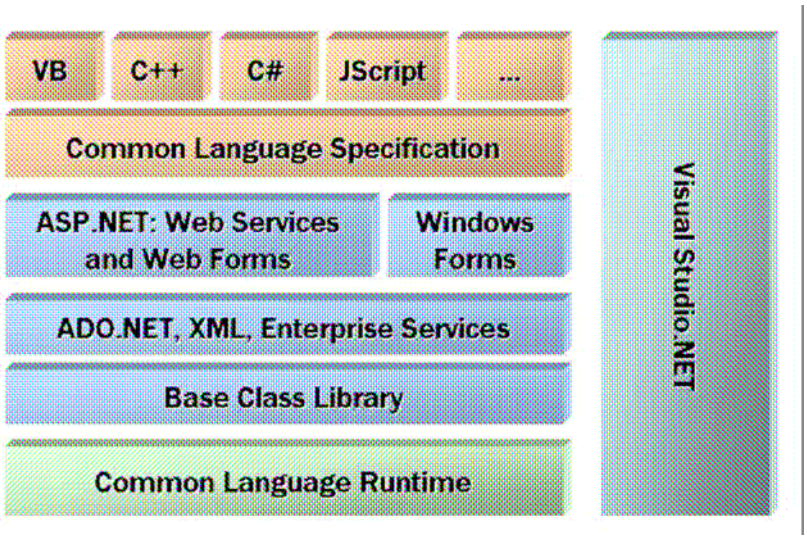
La plataforma .NET nos ofrece infinidad de funcionalidades que se utilizan como punto de partida para crear las aplicaciones. Existen funcionalidades básicas (por ejemplo todo lo relacionado con la E/S[7] de datos o seguridad) y funcionalidades avanzadas en las que se fundamentan categorías enteras de aplicaciones (acceso a datos, creación de aplicaciones Web, etc.).

Toda esta funcionalidad está implementada en forma de bibliotecas de funciones que físicamente se encuentran en diversas **DLL** (bibliotecas de enlazado dinámico). A su conjunto se le denomina **Base Classes Library** (**BCL**[8]) y forman parte integral de la plataforma .NET, es decir, no se trata de complementos que se deban obtener o adquirir aparte.

La siguiente figura 2.13 muestra la arquitectura conceptual de la plataforma .NET, se pueden observar los elementos que se han mencionado en apartados anteriores (lenguajes, CLR, CLS, etc.) y en qué lugar de se ubican las bibliotecas de clases base:

[7] Entrada y Salida.
[8] Biblioteca de Clases Base.

Figura 2.13 Arquitectura de la plataforma .NET



Fuente: Alarcón, José; Desarrollo Web con Visual Studio 2005.

Los espacios de nombres.

Debido a la gran cantidad de clases que existen debe haber algún modo de organizarlas de un modo relacionado. Además hay que tener en cuenta que podemos adquirir más funcionalidades de otros fabricantes, por no mencionar que se continuamente nuevas clases propias.

Para solucionar este problema existen en todos los lenguajes .NET los **espacios de nombres** o **namespaces**, que son un identificador que permite organizar de modo hermético las clases que estén contenidas en el así como otros espacios de nombres.

Por ejemplo, todo lo que tiene que ver con el manejo de estructuras de datos XML en la plataforma .NET se encuentra bajo el espacio de nombres **System.Xml.** La funcionalidad fundamental para

crear aplicaciones Web está en el espacio de nombres **System.Web**. Él cual contiene otros espacios de nombres más especializados como **System.Web.Caching** para la persistencia temporal de datos, **System.Web.UI.WebControls,** que contiene toda la funcionalidad de controles Web para interfaz de usuario, etc.

2.7.4 Acceso a datos con ADO.NET

El acceso a fuentes de datos es algo indispensable en cualquier lenguaje o plataforma de desarrollo. La parte de la **BCL** que se especializa en el acceso a datos se denomina de forma genérica como **ADO.NET**.

ADO.NET es un modelo de acceso mucho más orientado al trabajo desconectado de las fuentes de datos de lo que nunca fue **ADO**. Si bien este último ofrecía la posibilidad de desconectar los **Recordsets**[9] y ofrecía una forma de autonumeración de estos a través de las diferentes capas de una aplicación, **ADO** no es ni de lejos tan potente como el que nos ofrece **ADO.NET** que controla directamente los requisitos del usuario para programar, se diseño específicamente para la Web.

ADO.NET utiliza algunos objetos ADO, como **Connection** y **Command**, y también agrega objetos nuevos. Algunos objetos clave de **ADO.NET** son **DataSet**, **DataReader** y **DataAdapter**.

[9] **Recordset:** representa los registros de una tabla o los registros del resultado de ejecutar una consulta"

La diferencia más importante entre **ADO.NET** y las arquitecturas de datos anteriores es que existe un objeto llamado **DataSet**, que es independiente y diferente de los almacenes de datos. Por ello, **DataSet** funciona como una entidad independiente. Se puede considerar el objeto **DataSet** como un conjunto de registros que siempre está desconectado y que no sabe nada sobre el origen y el destino de los datos que contiene. Dentro del **DataSet**, como en una base de datos hay tablas, columnas, relaciones, restricciones, vistas, etc.

El objeto **DataAdapter** se conecta a la base de datos para llenar el objeto **DataSet**. Después, se vuelve a conectar a la base de datos para actualizar los datos de dicha base de datos a partir de las operaciones realizadas en los datos contenidos en el objeto **DataSet**. Con el fin de proporcionar a las aplicaciones multinivel mayor eficacia, se está adoptando para el procesamiento de datos una dirección basada en mensajes que manipulan fragmentos de información. En el centro de este enfoque se sitúa el objeto **DataAdapter**, que proporciona un puente entre un objeto **DataSet** y un almacén de datos de origen para recuperar y guardar datos. Para ello, envía solicitudes a los comandos SQL apropiados que se ejecutan en el almacén de datos.

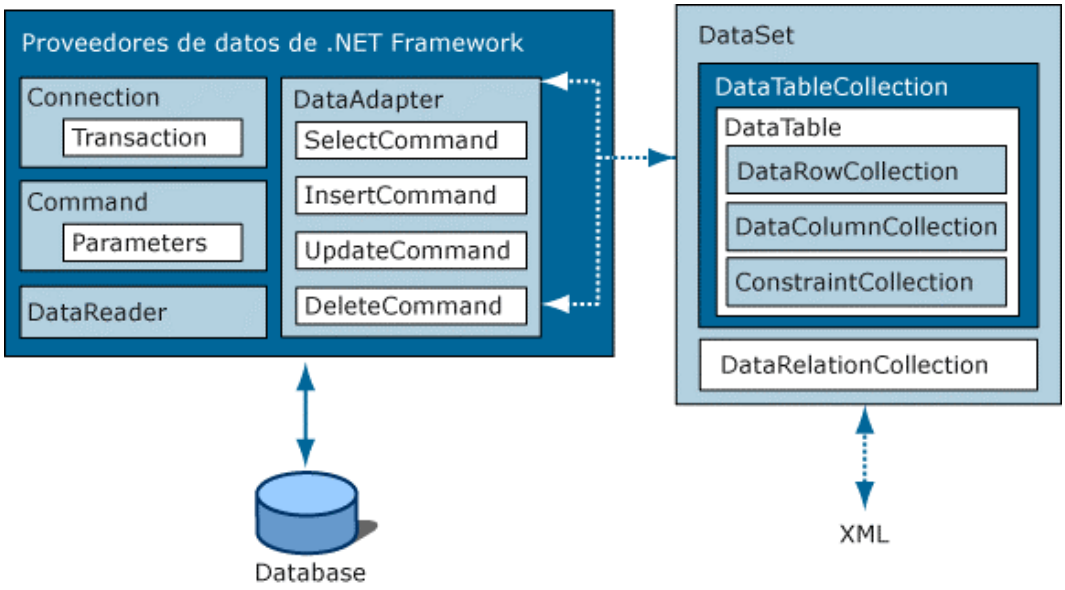
Los objetos son los siguientes:

- Objetos **Connection**. Para conectar con una base de datos y administrar las transacciones en una base de datos.
- Objetos **Command**. Para emitir comandos SQL a una base de datos.

- Objetos **DataReader**. Proporcionan una forma de leer una secuencia de registros de datos sólo hacia delante desde un origen de datos SQL Server.
- Objetos **DataSet**. Para almacenar datos sin formato, datos XML y datos relacionales, así como para configurar el acceso remoto y programar sobre datos de este tipo.
- Objetos **DataAdapter**. Para insertar datos en un objeto **DataSet** y reconciliar datos de la base de datos.

Arquitectura de ADO.NET. El concepto más importante que hay que tener claro sobre ADO.NET es su modo de funcionar, que se observa al analizar su arquitectura, véase figura 2.14.

Figura 2.14: Arquitectura ADO.NET



Fuente: http://msdn.microsoft.com/es-es/library/27y4ybxw(VS.80).aspx

Existen dos capas fundamentales dentro de su arquitectura: la **capa conectada** y la **desconectada.**

Capa Conectada

Contiene objetos especializados en la conexión con los orígenes de datos. Así, la clase genérica **Connection** se utiliza para establecer conexiones a los orígenes de datos. La clase **Command** se encarga de enviar comandos de todo tipo al origen de datos. La clase **DataReader** está especializada en leer los resultados de los comandos mientras se permanece conectado al origen de datos.

La clase **DataAdapter** hace uso de las tres anteriores para actuar de puente entre la capa conectada y la desconectada, estas clases no tienen una implementación real de la que se pueda hacer uso directamente. Aquí es donde entran en juego los **Proveedores de Datos**.

Así, por ejemplo, las clases específicas para acceder a SQL Server se llaman **SqlConnection, SqlCommand, SqlDataReader** y **SqlDataAdapter** y se encuentran bajo el espacio de nombres **System.Data.SqlClient**. Es decir, al contrario que en ADO clásico no hay una única clase **Connection** o **Command** que se use en cada caso, si no que existen clases especializadas para conectarse y recuperar información de cada tipo de origen de datos.

Existen **proveedores nativos**, que son los que se comunican directamente con el origen de datos (por ejemplo el de SQL Server o el de Oracle), y **proveedores "puente"**, que se utilizan para acceder a través de

ODBC u OLEDB cuando no existe un proveedor nativo para un determinado origen de datos.

Capa desconectada

Una vez que ya se han recuperado los datos desde cualquier origen de datos que requiera una conexión ésta ya no es necesaria. Sin embargo sigue siendo necesario trabajar con los datos obtenidos de una manera flexible. Es aquí cuando la capa de datos desconectada entra en juego, además en muchas ocasiones es necesario tratar con datos que no han sido obtenidos desde un origen de datos relacional con el que se requiera una conexión. A veces únicamente necesitamos un almacén de datos temporal pero que ofrezca características avanzadas de gestión y acceso a la información.

Por otra parte las conexiones con las bases de datos son uno de los recursos más escasos con los que contamos al desarrollar. Su mala utilización es la causa más frecuente de saturación en las aplicaciones y que éstas no progresen como es debido, esto es importante en las aplicaciones Web en las que se pueden recibir muchas solicitudes simultáneas de cualquier parte del mundo.

Otro motivo por el que es importante el uso de los datos desconectado de su origen es la transferencia de información entre capas de una aplicación, estas pueden encontrarse distribuidas por diferentes equipos, e incluso en diferentes lugares del mundo gracias a Internet. Por esto es necesario disponer de algún modo genérico y eficiente de poder transportar los datos entre diferentes lugares,

utilizarlos en cualquiera de ellos y posteriormente tener la capacidad de ajustar los cambios realizados sobre ellos con el origen de datos del que proceden.

Esto y mucho más es lo que nos permite el uso de los objetos **DataSet**, los cuales están pensados y diseñados con estos objetivos en mente.

Otra interesante característica de los **DataSet** es que permiten tramitar simultáneamente diversas tablas de datos, cada una de un origen diferente si es necesario, teniendo en cuenta las restricciones y las relaciones existentes entre ellas.

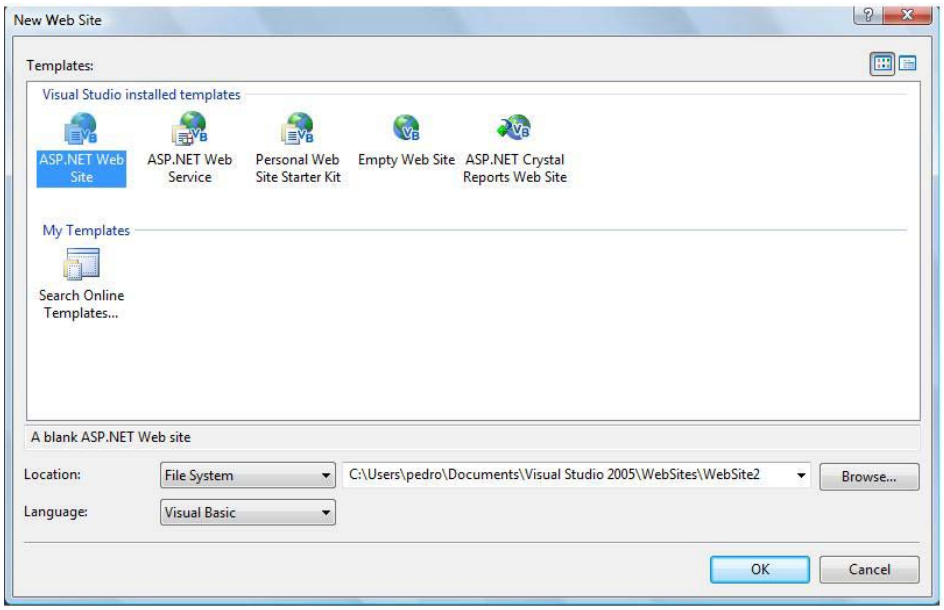
CAPÍTULO III
FUNDAMENTOS DE APLICACIONES WEB

3.1 ENTORNO DE DESARROLLO: Visual Studio 2005

Visual Studio 2005 es el entorno de desarrollo de aplicaciones Web para la versión ASP.NET 2.0 de la plataforma .NET. Tiene todo tipo de herramientas para facilitar el trabajo: diseñadores gráficos de páginas y clases, asistentes de uso de bases de datos, un servidor Web de desarrollo, ayuda a la escritura de código, y en general todo lo que se espera de un entorno de desarrollo rápido moderno.

Para crear un nuevo proyecto ASP.NET se abre el menú **Archivo, Nuevo Sitio Web.** A continuación aparecerá un cuadro de diálogo véase figura 3.1.

Figura 3.1: Cuadro de Dialogo Nuevo Proyecto Web.

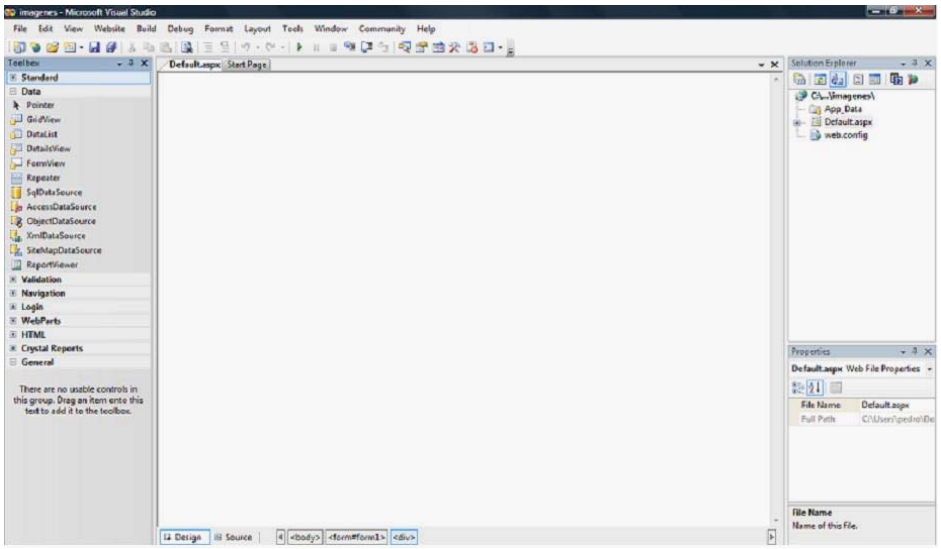


Fuente: Imagen creada por el Autor.

Explorando el entorno

Después de crear un proyecto nuevo, el entorno tiene este aspecto, véase figura 3.2.

Figura 3.2: Entorno de Visual Studio 2005 después de crear un proyecto.



Fuente: Imagen creada por el Autor.

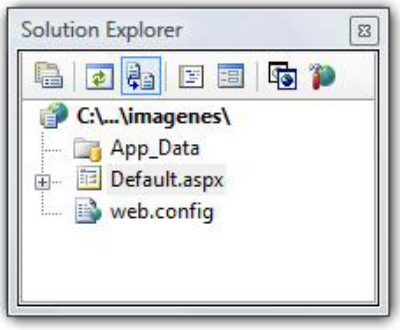
3.2 Elementos que forman el entorno Visual Studio .NET.

- **Explorador de Soluciones.-** Este elemento contiene un árbol con los proyectos en los que estamos trabajando y los diferentes archivos y carpetas que forman parte de ellos; cuando un nuevo proyecto Web sólo existe una carpeta llamada **App_Data**[1]**,** y una página **ASPX** creada de forma predeterminada que está vacía y es con la que se trabajara; los botones de la parte superior realizan diversas acciones sobre el elemento que se tenga seleccionado. Por

[1] Contiene los archivos de datos de aplicación incluso los archivos MDF, archivos XML, así como otros archivos de almacén de datos. Se utiliza la carpeta App_Data para almacenar la base de datos local de una aplicación, que se puede utilizar para mantener información sobre suscripciones y funciones.

ejemplo en el caso de la página podemos abrir su diseño o su código presionando respectivamente el tercer y cuarto botón a la derecha, véase figura 3.3.

Figura 3.3: Explorador de Soluciones.



Fuente: Imagen creada por el Autor.

3.3 TIPOS DE DATOS DE .NET.

El uso de las tecnologías de Internet ha tenido gran avance para el lenguaje **VBScript**[2], debido a que es parte fundamental en la ejecución de aplicaciones de servidor programadas en **ASP**[3].

En **VBScript** solo existe un tipo de datos llamado **Variant**. Este es un tipo de datos especial que puede almacenar diferentes tipos de información, en el caso más sencillo de un **Variant**, contiene información numérica o textual, en caso de tener una relación textual con números estos deberán estar entre comillas para que también se maneje como texto.

[2] Abreviatura de Visual Basic Script Edition.
[3] Active Server Pages.

Dentro de **Variant** se pude tener más precisión en el tipo de información gracias a los **Subtipos** (véase figura 3.4), que se encuentran incluidos dentro de este, además VBScript pone a nuestra disposición funciones para convertir los datos de un tipo a otro.

Figura 3.4: Descripción de los Subtipos.

Subtipo.	Descripción.	Valor de Vartype.
Empty.	Variable sin inicializar.	0
Null.	Variable intencionadamente vacía.	1
Boolean.	Dos valores posibles **True** o **False.**	11
Byte.	Entero entre 0 y 255.	17
Integer.	Entero entre -32.768 y 32.768.	2
Currency.	Numero entre -922.337.203.685.477,5808 y 922.337.203.685.477,5807	6
Long.	Numero entre -2.147.483.648 y 2.147.483.647	3
Single.	Numero de precisión simple.	4
Double.	Numero de doble precisión.	5
Date.	Fecha.	7
String.	Cadena de longitud variable hasta 2.000.000.000 de caracteres.	8
Object.	Contiene un Objeto.	9
Error.	Contiene un número de error.	10

Fuente: Tabla creada por el Autor.

3.3.1 Conversión de Tipos.

Todas las funciones (véase figura 3.5) de conversión de tipos tienen la misma sintaxis:

Función (expresión)

Donde expresión es el dato que se desea convertir.

Figura 3.5: Tabla de Funciones.

Función.	Descripción.
Cbool.	Convierte una expresión a tipo Boolean.
Cbyte.	Convierte una expresión a tipo Byte.
Clnt.	Convierte una expresión a tipo Integer.
Clng.	Convierte una expresión a tipo Long.
Csng.	Convierte una expresión a tipo Single.
Cdbl.	Convierte una expresión a tipo Double.
Ccur.	Convierte una expresión a tipo Curency.
Cdate.	Convierte una expresión a tipo Date.
Cstr.	Convierte una expresión a tipo String.

Fuente: Tabla creada por el Autor.

También se puede conocer el subtipo de una variable a través de la función **Vartype (variable)** que devuelve el valor en la tercera columna de la tabla 3.4.

3.3.2 Variables.

Son nombres que se les da a los lugares donde se almacenará información. Aunque no es necesario declarar variables antes de usarlas.

Para esto tenemos tres palabras reservadas: **Dim**, **Public** y **Private**. La diferencia entre ellas es el lugar en el que las variables declaradas serán válidas.

- **Public.-**La variable será accesible desde cualquier parte del código.
- **Dim**.-Se utiliza sólo en la función o procedimiento en que se declara; pero si se declara fuera de ellos será accesible para todos.

- **Private**.- La variable sólo será accesible en el nivel en el que está. Ningún procedimiento podrá acceder a una variable privada declarada en fuera de él.

Ejemplos:

```
Dim     Variable1    'Acceso dentro y fuera de Pruebas.
Public  Variable2    'Acceso dentro y fuera de Pruebas.
Private Variable3    'Acceso sólo fuera de Pruebas.

Sub Pruebas
    Dim     Variable4    'Acceso sólo dentro de Pruebas.
    Public  Variable5    'Acceso dentro y fuera de Pruebas.
    Private Variable6    'Acceso sólo dentro de Pruebas.
End Sub
```

Se pueden declarar varias variables en una misma sentencia separándolos por comás, ejemplo:

Dim a, b, c, d

Si se utiliza una variable sin ser declarada, automáticamente se asigna a **Dim** en el lugar en que la estés usando.

Restricciones del los nombres de variables:

- Debe comenzar con un carácter alfabético.
- No puede contener punto.
- No debe superar los 255 caracteres.

Asignación de valores a una variable escalar.

Son variables que contiene un **único** valor, ejemplo:

MiVariable = "monitor"

Para asignar valores a las variables se debe toma en cuenta los siguientes puntos:

- Los valores de cadena se asignan entre comillas.- MiVariable = "monitor".
- Los valores numéricos se asignan sin comillas.- MiVariable = 33.
- Los valores de fecha se asignan entre signos de gato.- MiVariable = #12-1-1999#.

Declaración de Matrices:

Se declaran de la misma manera que las escalares, con la diferencia de que las matrices utilizan paréntesis ( ) después del nombre de la variable; dentro del paréntesis va el número de elementos que consta la matriz, ejemplo:

Dim MiMatriz (10)[4]

Para asignar un valor a un elemento de la matriz, se indica el índice de la matriz, ejemplo:

MiMatriz (2) = 122

[4] VBScript comienza a numerar a partir de cero, debido a esto MiMatriz contiene 11 elementos.

Y para recuperar un valor almacenado lo siguiente:

MiVariable = MiMatriz (2)

Las matrices en VBScript pueden tener hasta 60 dimensiones separadas por comás, por ejemplo, la siguiente instrucción crea una matriz de 6 filas y 11 columnas:

Dim MiMatriz (5,10)

3.3.3 Constantes.

Es una variable a la que no se puede cambiar el valor que se le asigno, Se declaran mediante **Const**, ejemplo:

Const administrador = "José"

3.4 ÁREA DE DOCUMENTOS.

Es el lugar en el centro del entorno de desarrollo, contiene los diferentes editores de código así como diseñadores de diversos tipos (de interfaz de usuario, de clases, de DataSets, etc.). Es en donde se realiza la mayor parte del trabajo, véase figura 3.6.

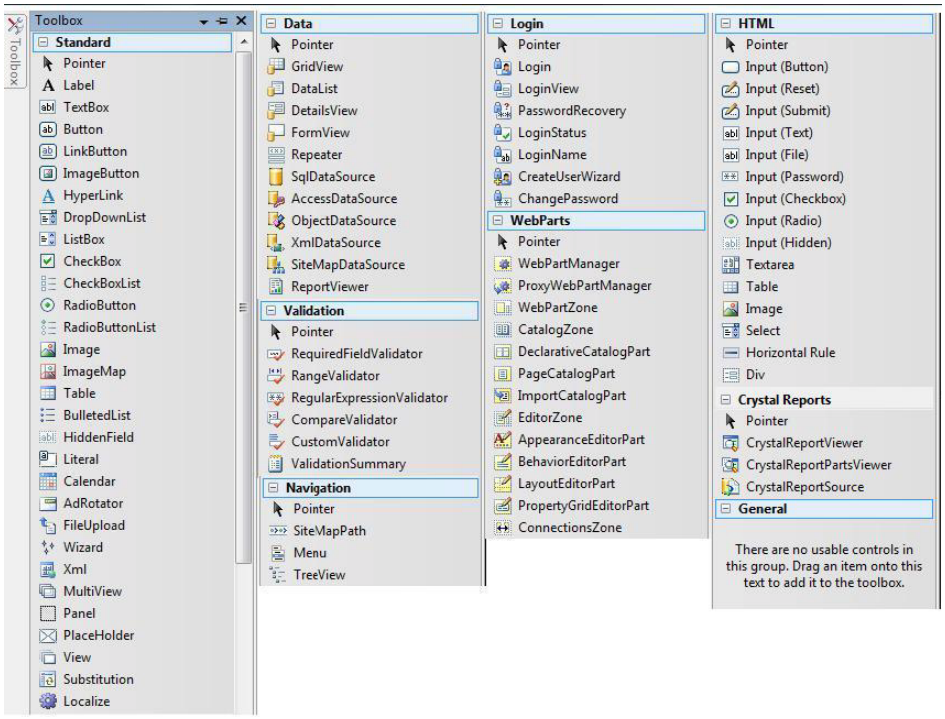
Figura 3.6: Área de Documentos.

Fuente: Imagen creada por el Autor.

3.5 CUADRO DE HERRAMIENTAS.

El cuadro de herramientas contiene los diferentes elementos que podemos utilizar para la definición de la interfaz de usuario de nuestra aplicación, así como algunos otros componentes no visuales que también se pueden arrastrar hacia el diseñador visual de páginas Web. Está situado por defecto en el lateral izquierdo del entorno, véase figura 3.7.

Figura 3.7: Cuadro de Herramientas.



Fuente: Imagen creada por el Autor.

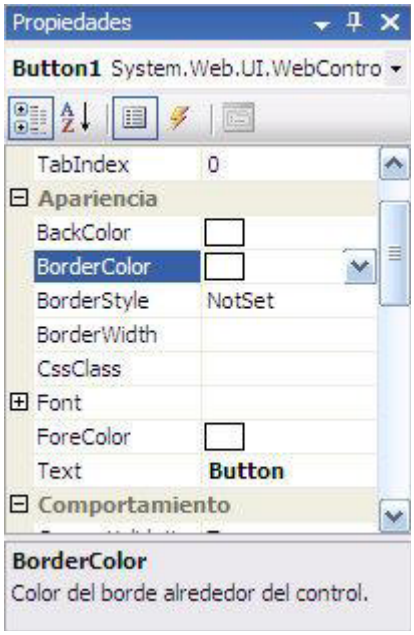
3.6 EDITOR DE PROPIEDADES.

Al igual que en Visual Basic 6.0, FrontPage y otros entornos, el editor de propiedades nos permite acomodar los valores de las propiedades en tiempo de diseño de los objetos que hayamos escogido.

El editor sirve para ajustar las propiedades de todos los objetos que podamos utilizar en el entorno: de los controles que se arrastran sobre un diseñador y de los propios archivos del explorador de soluciones y las etiquetas HTML o XML, etc.

Algunas propiedades se editan directamente en el espacio disponible y otras lanzan un asistente o un diseñador que nos ayuda con la tarea, se pueden ordenar alfabéticamente o agrupadas en diferentes categorías, las propiedades también pueden ajustar los gestores de eventos de los objetos usando el penúltimo botón por la derecha. En la parte inferior se obtiene una ayuda sobre cada propiedad seleccionada, véase figura 3.8.

Figura 3.8: Editor de Propiedades.



Fuente: Imagen creada por el Autor.

3.7 ARCHIVOS DE CÓDIGO.

Existen dos tipos de archivos de código: uno que define la interfaz de usuario y otro para la lógica de la aplicación. Veamos cómo están formados y cuál es el nexo de unión entre ellos.

3.7.1 El archivo .aspx de interfaz de usuario.

Todo el código se ejecuta en el servidor y por la experiencia de un programador, el evento desencadenado por la pulsación se gestiona en el servidor, no en el cliente. Veamos cómo funciona.

Para crear la interfaz de usuario sólo hemos tenido que arrastrar controles Web desde el cuadro de herramientas al diseñador, y lo que ha estado ocurriendo en el área de documentos (fuente) es que el código HTML de la página va aumentando.

3.7.2 El archivo .vb de lógica de la aplicación.

Por otro lado existe un archivo con extensión .vb, dependiente, según el explorador de proyectos, del archivo .aspx anterior.

Éste contiene la lógica de la aplicación, es decir, lo que hace que una interfaz de usuario se comporte de un determinado modo. Desde este archivo de código podemos responder a cualquier evento de los controles de interfaz de usuario o de la propia página, y acceder a sus métodos y propiedades.

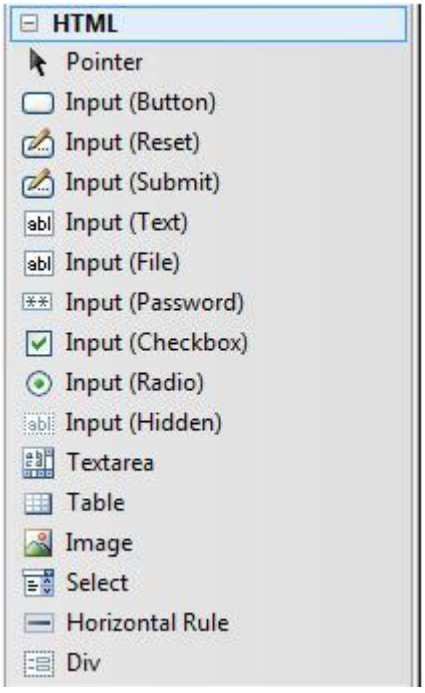
3.8 CONTROLES ASP.NET.

ASP.NET ofrece una gran cantidad de controles que se pueden usar en los desarrollos de aplicaciones Web, anterior hemos podido ver algunos de ellos al verlos en el cuadro de herramientas.

3.8.1 Controles HTML.

En el cuadro de herramientas tenemos el grupo **HTML** (véase figura 3.9) que son controles equivalentes a los de HTML, estos podemos arrastrarlos y soltarlos sobre el formulario igual que los demás, pero éstos no se ejecutan por defecto en el servidor. Sólo aparecerá su sintaxis HTML.

Figura 3.9: Controles HTML.



Fuente: Imagen creada por el Autor.

Son controles muy útiles en determinadas ocasiones en las que no se necesitan todas las ventajas que ofrecen los controles de servidor, por ejemplo:

- No se puede acceder a sus métodos y propiedades desde el servidor.
- Tal vez no se necesita que mantengan su estado o respondan a evento alguno.
- El uso del campo oculto ViewState puede cargar la página en exceso si hay muchos controles, también hay que crear clases en el servidor que los representen cuando se procesa la página. Todo ello reduce la respuesta de la página.

En el área del diseño del formulario los controles de servidor son muy fáciles de distinguir de los HTML porque los primeros, tienen un pequeño triángulo verde que los marca.

Los controles HTML, son mucho más sencillos que los otros controles Web. Tienen menos propiedades y eventos, son más adecuados cuando no se requiere una gran flexibilidad y queremos cargar la página lo mínimo posible.

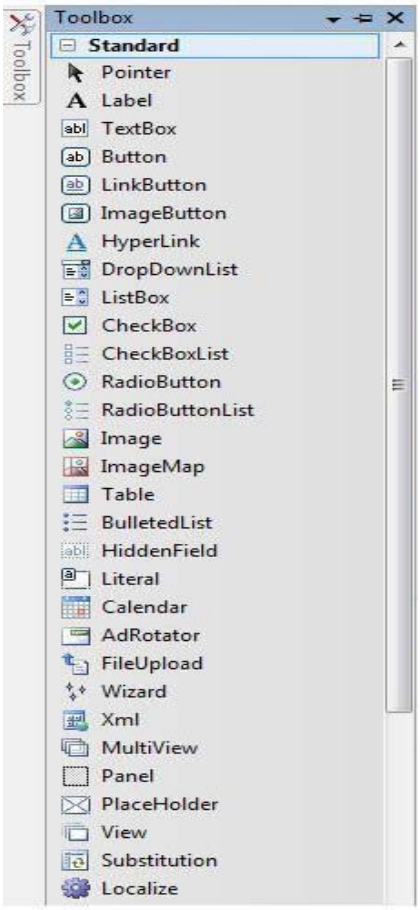
3.8.2 Controles Web.

Son controles naturales de **ASP.NET.** Aunque algunos parecen controles HTML, todos van mucho más allá en cuanto a características y capacidades, de hecho aunque algunos son relativamente sencillos (como una etiqueta, un botón o un cuadro de texto), existen controles muy complejos que sería difícil recrear desde cero con HTML y

JavaScript. Por ejemplo el control calendario, las rejillas de datos, los controles maestro-detalle, etc.

Sus métodos y propiedades tienen nombres consistentes con el resto de la plataforma. Por ejemplo, para fijar el texto de un botón o de una etiqueta se usa la misma propiedad **Text.** Para establecer el color de fondo todos usan **BackColor**. Esto hace que sea más fácil el desarrollo porque no hay que memorizar nomenclaturas diferentes para cada control, véase figura 3.10.

Figura 3.10: Controles Web.

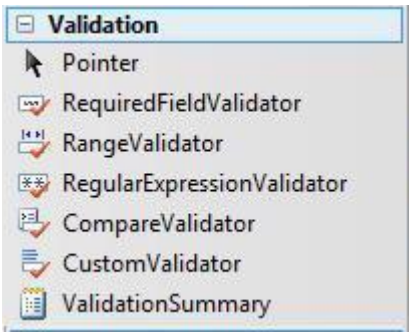


Fuente: Imagen creada por el Autor.

3.8.3 Controles Web de Validación.

Dentro de la variedad de controles de **ASP.NET** existe un grupo "**Validación",** que contiene una serie de controles que permiten realizar la validación de datos introducidos por los usuarios, véase figura 3.11.

Figura 3.11: Controles de Validación.



Fuente: Imagen creada por el Autor.

Estos controles permiten definir reglas de validación en la entrada de datos. Dichas reglas se asocian con otros controles que forman parte del formulario Web, estas se combinan para tener restricciones sobre los datos introducidos.

Las condiciones típicas son, por ejemplo, que un campo no se quede vacío, que tiene que estar dentro de un rango determinado o incluso cumplir con una expresión que indiquemos. También es posible definir reglas propias personalizadas.

La **ventaja principal** de estos controles es que permiten la definición de reglas de validación de forma **declarativa**, es decir, que no

hace falta escribir un código para usarlos. Esto facilita el desarrollo y mantenimiento de las reglas de validación.

Una vez definidas las reglas para un formulario los controles de validación se encargan automáticamente de validarlas **tanto en el cliente como en el servidor**. Por el lado del cliente se convertirán en código JavaScript, actuando de primera barrera y evitando viajes innecesarios al servidor. Las validaciones del lado del servidor evitan problemas cuando, el motivo que sea, no se han realizado las validaciones en el cliente.

Se puede desactivar la validación por el lado del cliente de un control cambiando la propiedad, **EnableClientScript** a **False**. También se puede deshabilitar la validación del lado cliente de todos los controles cambiando la propiedad **ClientTarget** de la página actual con la cadena **DownLevel** desde el evento de **load** de la página. Con esto sólo realizará la validación en el servidor.

Uso de los controles de validación

Para hacer uso de estos controles basta con arrastrarlos al formulario, al hacerlo se muestran como si fueran etiquetas normales, aunque con el texto de color rojo. Es el aspecto que tendrán si se hace necesario su actuación. Mientras no se produzca una situación en la que la validación fracase serán invisibles. Toda esta funcionalidad se

consigue utilizando JavaScript, es decir, que no se envía nada al servidor (no se hace un **post-back**[5]).

Cada control de validación que arrastremos se debe asociar al control que deberá cuidar que cumpla con los requisitos. Por supuesto es posible arrastrar varios validadores y asociarlos a un mismo control para así verificar diferentes condiciones pero no se puede usar un solo validador para verificar varios controles. El control a verificar se asigna mediante la propiedad **ControlToValidate** del control de validación.

El control **ValidationSummary** (abajo del grupo de controles de la figura 3.11) se utiliza para mostrar un resumen de todo lo que está mal en un formulario, en lugar de enviar cada uno de los mensajes de error individualmente.

La siguiente tabla (véase figura 3.12) indica el uso de cada uno de los controles disponibles:

[5] Es cuando una página Web y todo su contenido son enviados al servidor para procesar alguna información y después el servidor regresa la página con la información ya procesada.

Figura 3.12: Descripción de los Controles de Validación.

Control.	Descripción.
RequiredFiledValidator	Verifica que el control asociado no se encuentre vacío.
RangeValidator.	Genera un mensaje de error cuando el contenido de su control asociado está fuera de un rango de valores dado. Permite validar intervalos numéricos (enteros o decimales o monedas), fechas y cadenas de texto.
RegularExpressionValidator	Compara un texto introducido por el usuario con una expresión regular.
CompareValidator	Permite comparar el valor introducido por el usuario con una constante o con el valor de la propiedad de otro control.
CustomValidator	Se usa para implementar lógica de validación propia tanto en el cliente como en el servidor.

Fuente: Tabla creada por el Autor

No todos los controles se pueden validar con los controles de validación. Sólo un pequeño subconjunto de los controles Web es adecuados aunque estos realizan la mayor parte de las necesidades normales de introducción de datos, y son los siguientes, véase figura 3.13.

Además de mostrar la información de error al usuario, en los eventos de la página gestionados en el servidor se puede comprobar el valor de la propiedad **IsValid** del objeto **Page**. Ésta será **False** si alguno de los controles de validación ubicados en la página no pasan la prueba de

verificación. Esto es útil para realizar acciones complementarias en el servidor en caso de haber errores.

Figura 3.13: Controles que se pueden validar.

Control.	Tipo.	Propiedad.
HtmlinputText	Entrada de texto	Value
HtmlTextArea	Entrada de texto	Value
TextBox	Entrada de texto	Text
HtmlSelect	Lista de selección	Value
ListBox	Lista de selección	selectedItem.Value
DropDownList	Lista de selección	SelectedItem.Value
RadioButtonList	Botones de selección	SelectedItem.Value
HtmlInputFile	Envío de archivos	Value

Fuente: Tabla creada por el Autor.

Colocar el foco en el error.

Algo muy común después de validar datos en un formulario es colocar el foco sobre el control que contiene información errónea. De esta forma facilita al usuario la corrección del nuevo valor pues así no tiene que activar el control con el ratón. Esto se puede a través de los controles de validación únicamente estableciendo en **True** la propiedad **SetFocusOnError**.

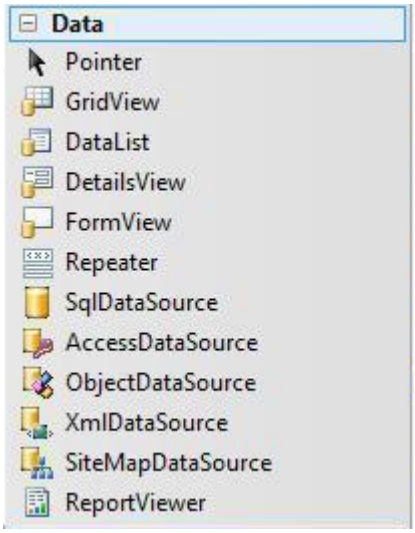
3.8.4 Controles de Datos.

Aparte de la escritura manual de código y el uso directo de objetos de ADO.NET, ASP.NET 2.0 proporciona un nuevo modelo de trabajo

para el acceso a datos que permite realzar tareas comunes de acceso a datos sin escribir código alguno.

Estos controles se encuentran agrupados en el cuadro de herramientas bajo el nombre de **Datos**, véase figura 3.14.

Figura 3.14: Controles de Datos.



Fuente: Imagen creada por el Autor.

Orígenes de datos.

Los controles de datos representan conexiones con diferentes tipos de orígenes de información que van desde bases de datos a objetos de negocio. Abstraen al programador de las complejidades relacionadas con el manejo de los datos, permitiendo de manera automática seleccionarlos, actualizarlos, ordenarlos, paginarlos, etc.

ASP.NET ofrece los siguientes controles de origen de datos que se pueden ver en la siguiente figura 3.15:

Figura 3.15: Descripción de Controles de Origen de Datos.

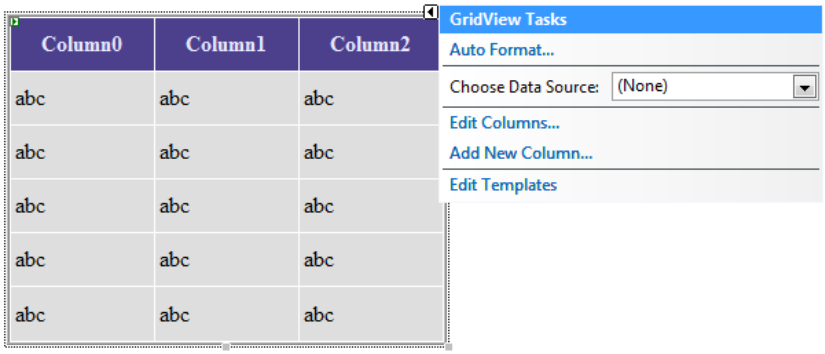
Control.	Descripción.
SqlDataSource	Enlaza con cualquier base de datos para que exista un proveedor de ADO.NET.
AccessdataSource	Esta especializado en trabajar con bases de datos Microsoft Access.
ObjectDataSource	Se enlaza con objetos de negocio y capas personalizadas de acceso a datos.
XmlDataSource	Trata datos contenidos en documentos XML.
SiteMapDataSource	Se enlaza con la jerarquía de clases expuesta por el modelo de navegación de sitios de ASP.NET 2.0.

Fuente: Tabla creada por el Autor.

3.8.4.1 GRIDVIEW.

Muestra los valores de un origen de datos en una tabla donde cada columna representa un campo y cada fila representa un registro. El control **GridView** permite seleccionar, ordenar y editar estos elementos, véase figura 3.16.

Figura 3.16: Control GridView.



Fuente: Imagen creada por el Autor.

El control **GridView** permite las siguientes características:

- Permite conectarse a controles de origen de datos, como SqlDataSource.
- Acción de ordenación.
- Actualizaciones y eliminaciones integradas.
- Paginación integrada.
- Permite la selección de filas.
- Diferentes campos de clave.
- Varios campos de datos para las columnas de hipervínculo.
- Personalización de la apariencia a través de temas y estilos.

Tipos Campos en las columnas.

Un objeto **DataControlField** representa cada columna del control **GridView**.

De forma predeterminada, la propiedad **AutoGenerateColumns** se establece en **true**, lo que genera un objeto **AutoGenerateField** para cada campo del origen de datos. Después cada campo se representa como una columna del control **GridView** en el orden de aparición de cada campo en el origen de datos.

Además se puede controlar de forma manual cuales campos aparecerán en las columnas del **GridView**; para esto, se debe cambiar la propiedad **AutoGenerateColumns** a **False** y después definir los campos de columnas que requiera.

Los distintos tipos de campo de columna determinan el comportamiento de las columnas del control, véase figura 3.17.

Figura 3.17: Tipos de Campos de Columnas.

Tipo de campo de columna.	Descripción.
BoundField.	Muestra el valor de un campo en un origen de datos. Es el tipo predeterminado del control **GridView**.
ButtonField.	Coloca un botón de comando para cada elemento del control **GridView**. El cual genera una columna de botones, como el botón Agregar o Quitar.
CheckBoxField.	Es cuadro de verificación para cada elemento del control **GridView**. Este tipo se utiliza para mostrar los campos con un valor booleano.
CommandField.	Muestra los botones de comando predefinidos para realizar operaciones de selección, edición o eliminación.
HyperLinkField.	Muestra el valor de un campo en un origen de datos como un hipervínculo el cual permite enlazar un segundo campo a la dirección URL del hipervínculo.
ImageField.	Permite una imagen para cada registro del control **GridView**.
TemplateField.	Muestra el contenido definido por el usuario para cada elemento del control **GridView** según una plantilla especificada.

Fuente: Tabla creada por el Autor.

Enlace a Datos.

El **GridView** se puede conectar a un control de origen de datos (como **Sql Data Source**, **Acces Data Source, Object Data Source**, **Site Map Data Source, Xml Data Source**).

Los siguientes puntos muestran algunas formas de conectar el **GridView** al tipo de origen de datos adecuado:

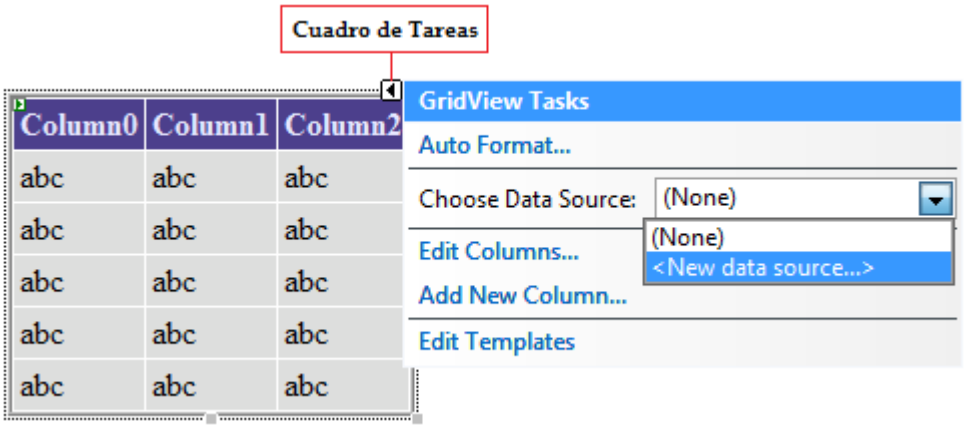
- Para conectar a un control de origen de datos, se establece la propiedad **DataSourceID** del control **GridView** en el valor conocido del control de origen de datos. El control **GridView** se enlaza automáticamente al control de origen de datos especificado o de ser necesario crear una nueva conexión, además con las funciones del control de origen de datos se pueden realizar operaciones de ordenación, actualización, eliminación y paginación, esta es la forma recomendada.

- Para conectar a un origen de datos que implementa la interfaz **System.Collections.IEnumerable**, establezca mediante programación la propiedad DataSource del control **GridView** en el origen de datos y, a continuación, utilice el método DataBind. Con este método, el control **GridView** no maneja las funciones integradas de ordenación, actualización, eliminación y paginación.

Pasos para Conectarse a una Base de Datos con el GridView.

- En primer lugar se debe seleccionar el **GridView** en los controles de datos y colocarlo en la plantilla de diseño, después despejar el cuadro de tareas del **GridView** ubicado en la esquina superior derecha, en la opción de **Choose Data Source** seleccionar **New Data Source**, véase figura 3.18.

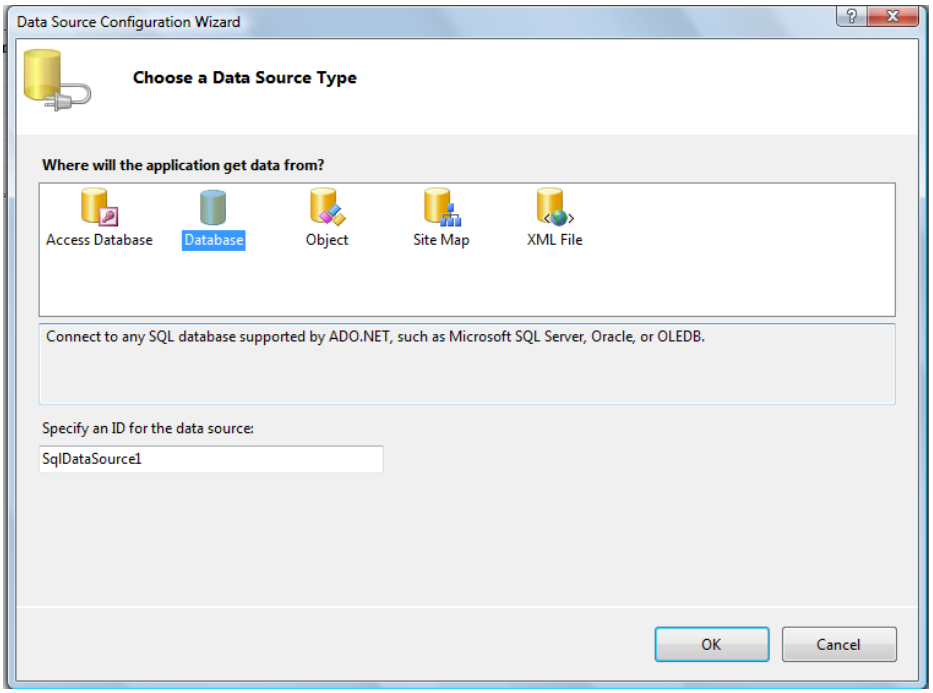
Figura 3.18: Cuadro de Tareas del GridView.



Fuente: Imagen creada por el Autor.

- Después de seleccionar el **New Data Source**, aparecerá lo siguiente, véase figura 3.19, aquí se seleccionará el tipo de origen de datos.

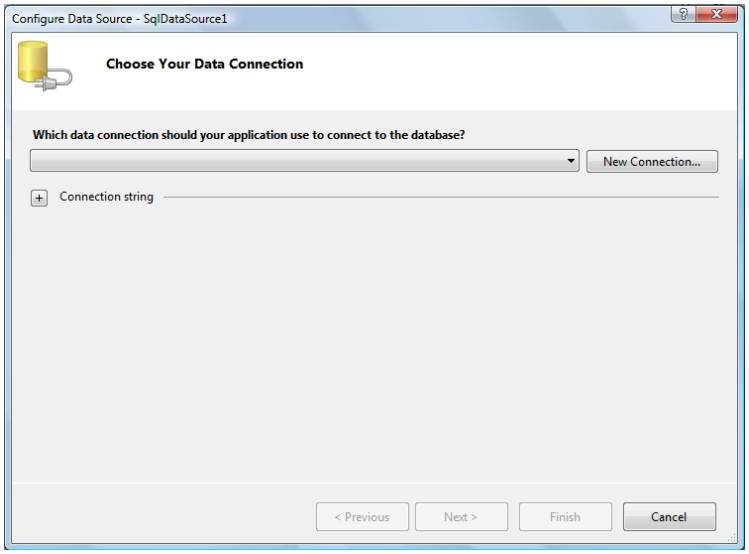
Figura 3.19: Cuadro de Tipo de Origen de Datos.



Fuente: Imagen creada por el Autor.

- El cuadro siguiente mostrara la forma de realizar la conexión a los datos o seleccionar una ya existente, véase figura 3.20.

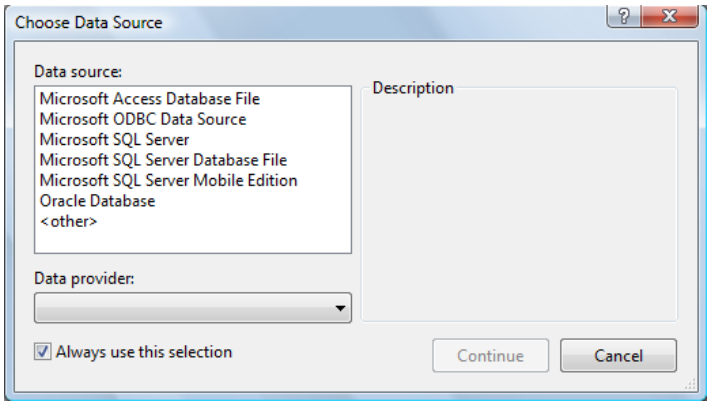
Figura 3.20: Cuadro de Conexión de Datos.



Fuente: Imagen creada por el Autor.

Si es una nueva conexión, se deberá escoger el tipo de base de datos a la cual se quiere tener acceso, véase figura 3.21.

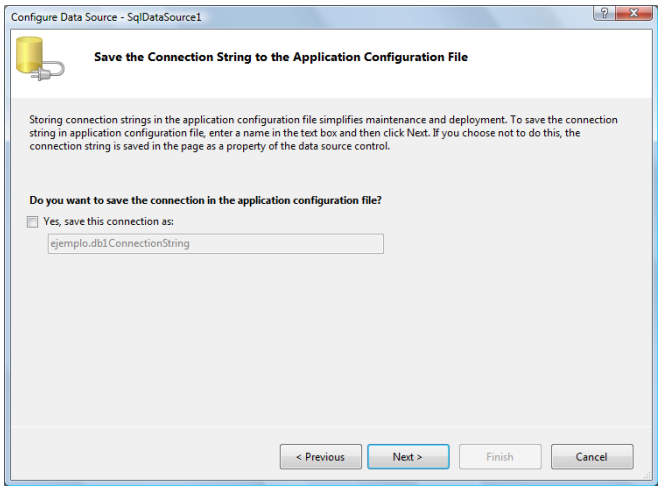
Figura 3.21: Cuadro para seleccionar una Base de Datos.



Fuente: Imagen creada por el Autor.

- Una vez escogida la base de datos mostrara el siguiente cuadro de dialogo donde muestra el nombre de la conexión, véase figura 3.22.

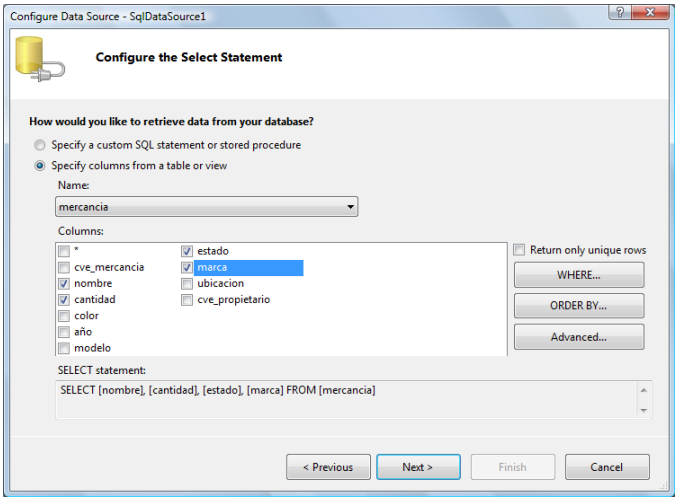
Figura 3.22: Nombre de la Conexión a la Base de Datos.



Fuente: Imagen creada por el Autor.

- A continuación se deberá de escoger la tabla y los campos que se deseen aparezcan en las columnas del **GridView**, con sus respectivos registros, véase figura 3.23.

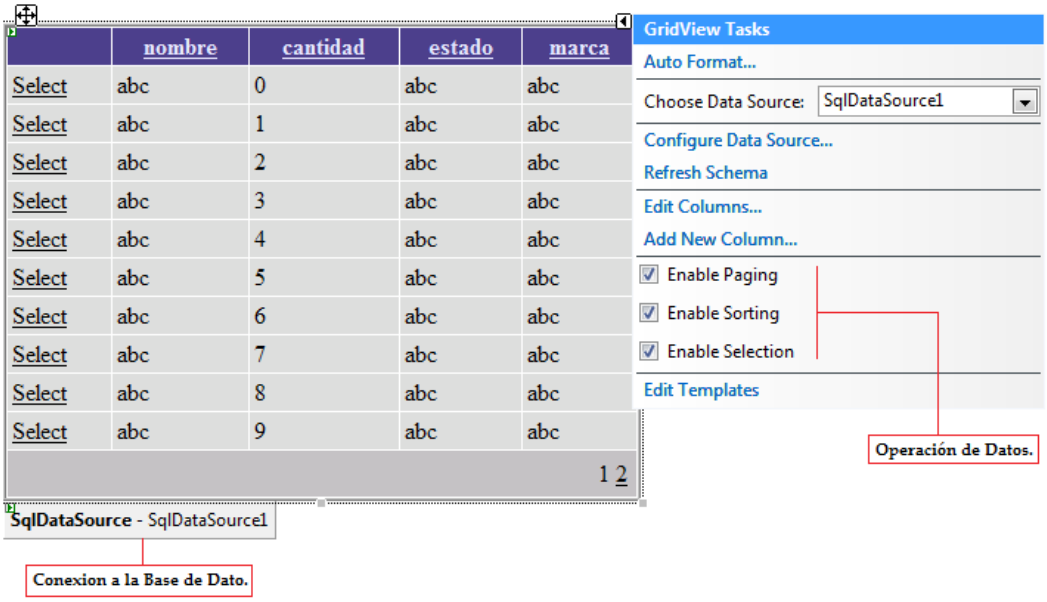
Figura 3.23: Consulta para llenado del GridView.



Fuente: Imagen creada por el Autor.

- Una vez realizado esto se dará por terminado el asistente, y en la forma del diseño aparecerá el **GridView** y en su cuadro de tareas las casillas de operaciones de datos deberán ser activadas, como se verán más adelante; véase figura 3.24.

Figura 3.24: Consulta para llenado del GridView.



Fuente: Imagen creada por el Autor.

Operaciones de los Datos.

El **GridView** tiene muchas funciones integradas que permiten al usuario ordenar, actualizar, eliminar, seleccionar y cambiar de página a página los elementos del control. Cuando el **GridView** se conecta a un origen de datos, este puede aprovechar la funcionalidad de ese control y proporcionar funciones de ordenación automática, actualización y eliminación, véase figura 3.25.

- **Ordenación.-** Permite al usuario ordenar los elementos del **GridView** en una columna específica al hacer clic en el encabezado de la columna. Para habilitar la ordenación, la propiedad **AllowSorting** en debe estar en **True**.
- Las funciones de actualización, eliminación y selección se pueden habilitar cuando se hace clic en un botón de un campo de columna **ButtonField** o **TemplateField** cuando nombre de comando sea **Editar**, **Eliminar** y **Seleccionar**. El **GridView** permite agregar un campo de columna **CommandField** automáticamente con los botones de **Editar**, **Eliminar** o **Seleccionar** de acuerdo a las necesidades del sistema, siempre y cuando las propiedades **AutoGenerateEditButton**, **AutoGenerateDeleteButton** o **AutoGenerateSelectButon**, respectivamente, se encuentren establecidas en **true**.
- **Paginación.-** Para no mostrar todos los registros del origen de datos al mismo tiempo, el **GridView** permite repartir los registros entre varias páginas automáticamente. Para lograr esto es necesario habilitar la paginación, estableciendo la propiedad AllowPaging en **true**.

Figura 3.25: Tipos de Operación de Datos.

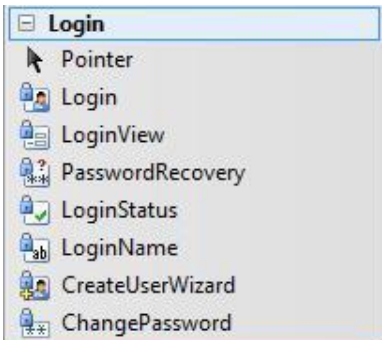
		nombre	cantidad	estado	marca
Editar Eliminar	Seleccionar	pantalon	2	excelente	levis
Editar Eliminar	Seleccionar	camisa	3	mal	Calvin Klein
Editar Eliminar	Seleccionar	reloj	4	bien	Rolex

Fuente: Imagen creada por el Autor.

3.8.5 Controles Web de Seguridad.

La seguridad en las aplicaciones se ha hecho mucho más fácil con ASP.NET. Debido a que nos facilita todavía más el trabajo relacionado con la seguridad gracias a la inclusión de los nuevos controles Web de seguridad. Los podemos encontrar en el grupo **Login** del cuadro de herramientas de Visual Studio, véase figura 3.26.

Figura 3.26: Controles de Seguridad.



Fuente: Imagen creada por el Autor.

Estos controles tienen hechas una gran cantidad de operaciones comunes de seguridad relacionadas con la interfaz de usuario. Por ejemplo, el control **Login** permite disponer de un completo diálogo de autentificación con sólo arrastrarlo sobre un formulario Web, véase figura 3.27a. El **CreateUserWizard** es un asistente con varios pasos que permite la creación automática de nuevos usuarios, véase figura 3.27b. Todos ellos permiten la personalización, tanto parcial por medio de propiedades, como absoluta usando plantillas. En los asistentes se tiene la libertad de agregar nuevos pasos o modificar los predeterminados.

Figura 3.27a: Control Login.

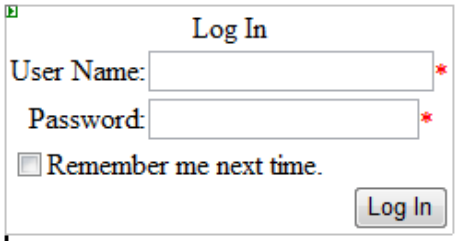


Figura 3.27b: Control CreateUserWizard.

Fuentes: Imágenes creadas por el Autor.

Control Login.

Este control permite definir un diálogo de autenticación en cualquier página, que además soporta el uso de temas, permite configurar enlaces para tener acceso a la creación de nuevos usuarios, muestra mensajes de error de autenticación, redirige automáticamente a otras páginas al autentificar si es necesario, etc. Todo esto se controla con facilidad en la ventana de **Propiedades** de Visual Studio.

Para validar a los usuarios se utiliza el método **ValidateUser** de la clase **Membership**. Si ya hay un usuario identificado, este control se oculta automáticamente al menos cuando se ubica en la página principal. Este comportamiento se cambia con la propiedad **AutoHide** en caso de necesitarlo. De esta manera se permite cambiar la sesión de usuario.

Control LoginStatus.

Muestra el estado actual de una conexión de un usuario y permite su desconexión. Si un usuario es identificado el control muestra por defecto un enlace que permite desconectarse, esto provoca que se elimine la referencia al usuario actual en la propiedad **User** de la clase **Page** y que se reenvíe a la página de autenticación para una iniciar nueva sesión.

Cuando no hay usuario identificado el sistema de enlace se dirige automáticamente a la página de autenticación definida.

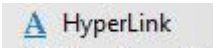
Control LoginName.

Muestra el nombre del usuario que se encuentra identificado.

3.9 NAVEGACIÓN ENTRE PÁGINAS.

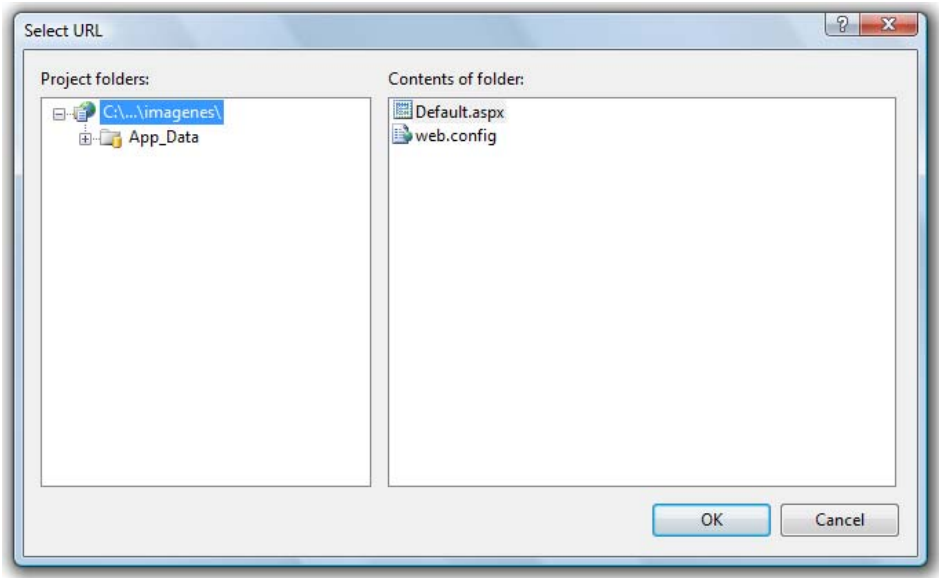
En **ASP.NET** se utilizan menos páginas que en el caso de que la misma aplicación estuviera en ASP clásico. Esto gracias a los **post-back** y los correspondientes eventos de servidor, por ejemplo en una aplicación ASP 3.0 clásica para recoger los datos de un usuario con probabilidad escribiríamos dos o más páginas: la página HTML con el formulario, la página ASP a la que se envían los datos del formulario y puede que una página o dos para informar al usuario del éxito o del fracaso de su acción. En ASP.NET todo esto se resuelve con una sola página ASPX, un evento de servidor y probablemente mostrando y ocultando controles Web desde dicho evento.

Una aplicación esta formada por un número más o menos elevado de páginas a las que se debe dirigir al usuario. Existen diversas maneras de dirigir a un usuario hacia otra página o recurso de la aplicación:

- La más sencilla, consiste en utilizar controles del tipo **HyperLink** que tiene este aspecto: 

, Estableciendo la propiedad **NavigateUrl** se indicara a qué página se enviará al usuario cuando de click sobre el link. Si la página pertenece a nuestra aplicación es muy fácil seleccionarla debido cuadro de diálogo especial que aparece para ello, véase figura 3.28.

Figura 3.28: Cuadro de dialogo de la propiedad NavigateUrl.



Fuente: Imagen creada por el Autor.

- La otra forma de cambiar a los usuarios de una página consiste en hacer uso del método **Redirect** de la clase **HttpResponse** del contexto de llamada de la página. Así se puede controlar desde un evento de servidor a dónde se enviara al usuario, por ejemplo:

Response.Redirect ("opcion1.aspx")

El cual enviará al usuario al momento de darle click en el link, a la página llamada **opcion1.aspx.**

CAPÍTULO IV

DISEÑO Y DESARROLLO

4.1 DISEÑO.

El diseño es un proceso de múltiples fases donde se simplifican representaciones de la estructura de los datos, la estructura del programa, las características de la interfaz y los detalles de procedimientos de los requisitos de la información.

La estructura de datos ha sido siempre una parte importante del diseño del software, debido a que son la parte esencial en la creación de aplicaciones de alta calidad. En una aplicación el conocimiento del modelo de datos en una base de datos es esencial para lograr los objetivos del sistema. En los negocios, la información almacenada en las bases de datos y guardada en el almacén de datos, facilita el descubrimiento de algunos puntos críticos que pudieran ser mejorados.

El almacén de datos es un conjunto de datos separados, el cual no esta integrado con lo realizado día a día, pero que contiene todos los datos utilizados por la empresa.

Los partes del análisis de requisitos del software son: los diagramas de flujo de información, la especificación de requerimientos y el diccionario de datos.

4.2 DIAGRAMA DE FLUJO DE INFORMACIÓN.

Los diagramas de flujo de información, representan gráficamente los procesos y flujos de datos de un sistema de información. En su estado original, los diagramas de flujo de datos muestran el panorama más amplio de las entradas, procesos y salidas del sistema.

Los elementos que componen a un diagrama de flujo de información son los siguientes:

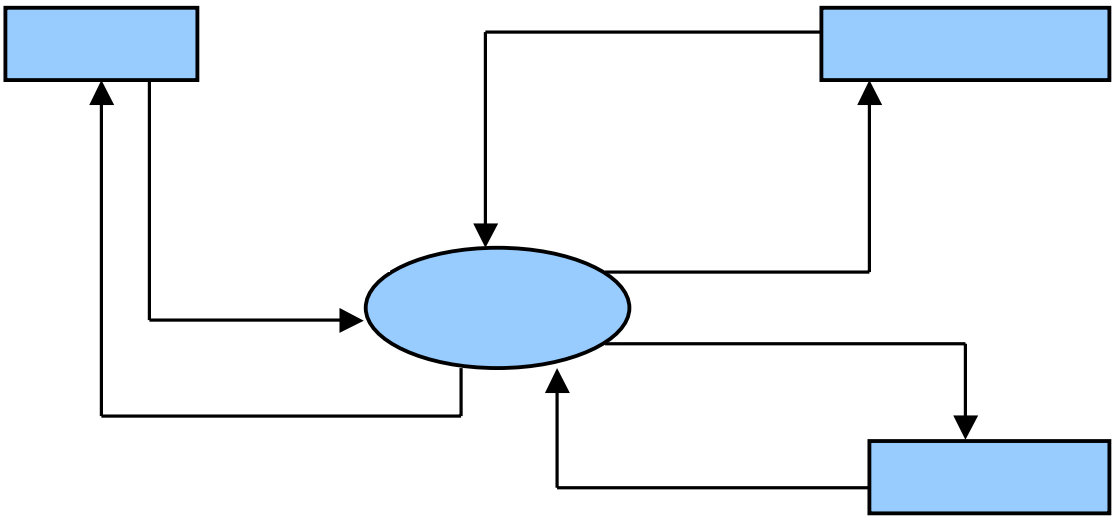
Función.- Es el cambio del flujo de datos, muestra cómo las entradas se transforman en salidas.

Flujo de datos.- Es la representación del flujo del dato del origen al destino, los datos siempre van hacia o vienen de una función.

Almacenamiento.- Generalmente son archivos, tablas, etc.

Para el sistema absoluto de control de mercancías, el diagrama de flujo de informaciones es el siguiente, véase figura 4.1.

Figura 4.1: Diagrama de Flujo de Información.



Fuente: Diagrama creado por el Autor.

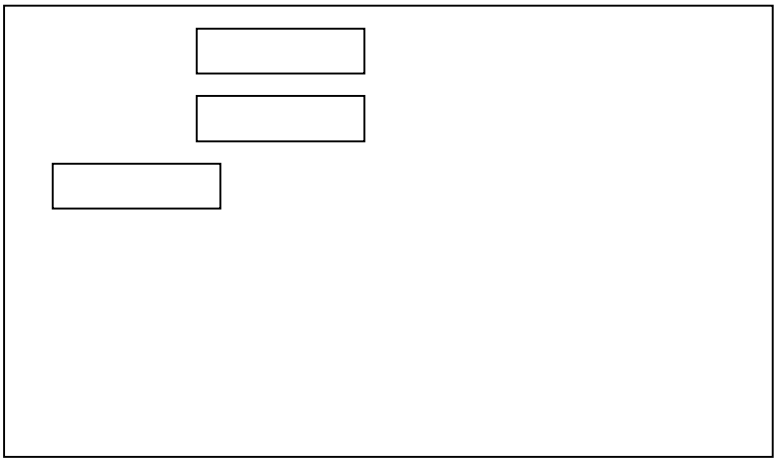
4.3 ESPECIFICACIÓN DE REQUERIMIENTOS.

Requerimientos de Entrada.

Requerimiento 1:0: Página de Acceso.

Se realizará una página de acceso, que dará entrada a los usuarios previamente registrados teniendo acceso a los diferentes tipos de menú de acuerdo al nivel al que hayan sido asignado en la base de datos, los cuales pueden ser administrador o usuario, de no estar registrado no tendrá ningún acceso debido a que puede haber robo de información confidencial, ejemplo1.

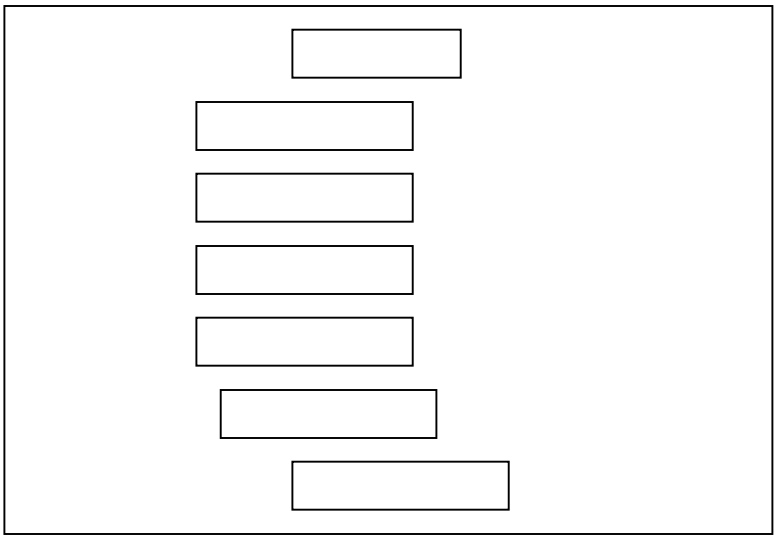
Ejemplo 1.



Requerimiento 2:0: Página de Registro.

A esta página se tendrá acceso únicamente desde el menú asignado a los administradores, y aquí los usuarios no registrados podrán ser dados de alta, ya que únicamente los usuarios con este nivel podrán hacerlo, y de esta manera podrán acceder a la(s) páginas que le sean asignadas según su cargo, ejemplo 2.

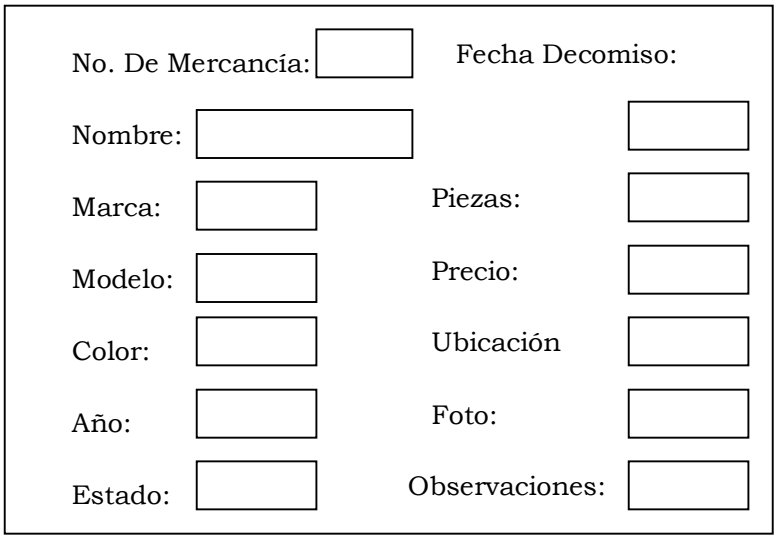
Ejemplo 2.



Requerimiento 3:0: Página de Altas de Mercancías.

En esta página se podrá dar de alta la mercancía incautada tomando en cuenta todas sus características, a esta página solo los administradores tendrán acceso, ejemplo 3.

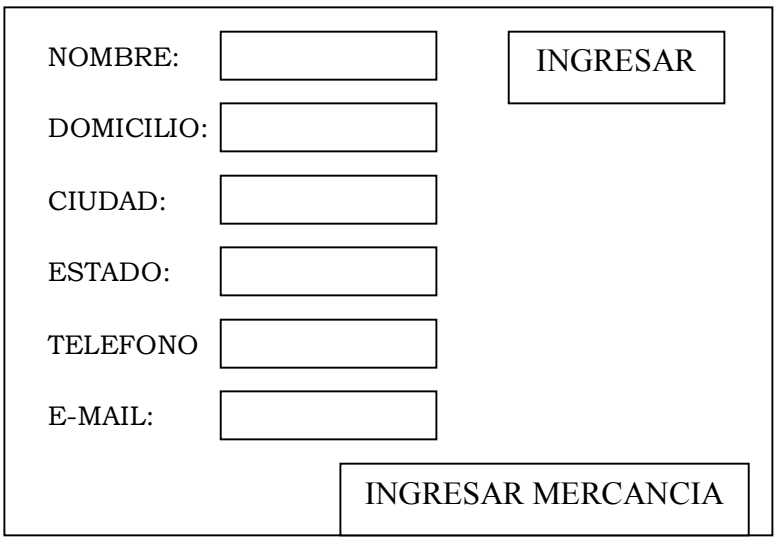
Ejemplo 3.



Requerimiento 4:0: Página de Altas de Propietario.

Aquí se dará de alta a los propietarios a los cuales le fue incautada la mercancía, agregando todas y cada uno de sus datos, esta página solo algunos administradores y usuarios tendrán acceso, ejemplo 4.

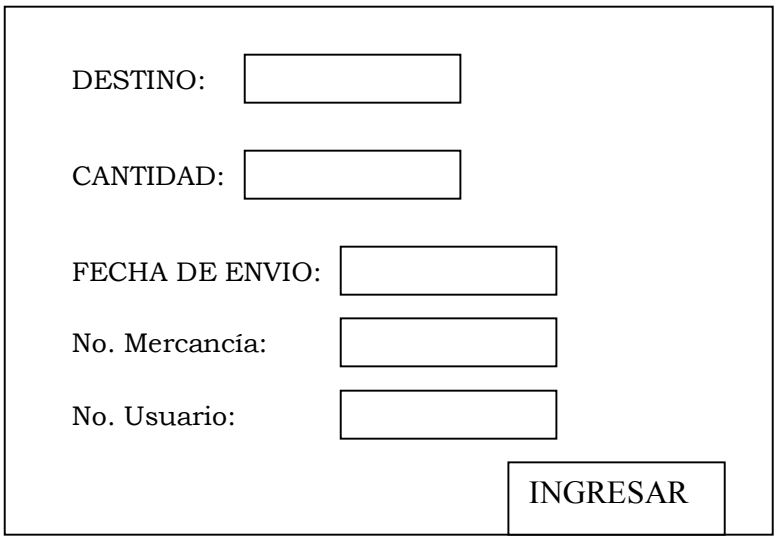
Ejemplo 4.



Requerimiento 5:0: Página de Destino.

En este lugar se asignará el destino de las mercancías incautadas, los cuales se determinaran por sus características y/o dimensiones, estas decisiones serán tomadas por los administradores, ejemplo 4.

Ejemplo 4.

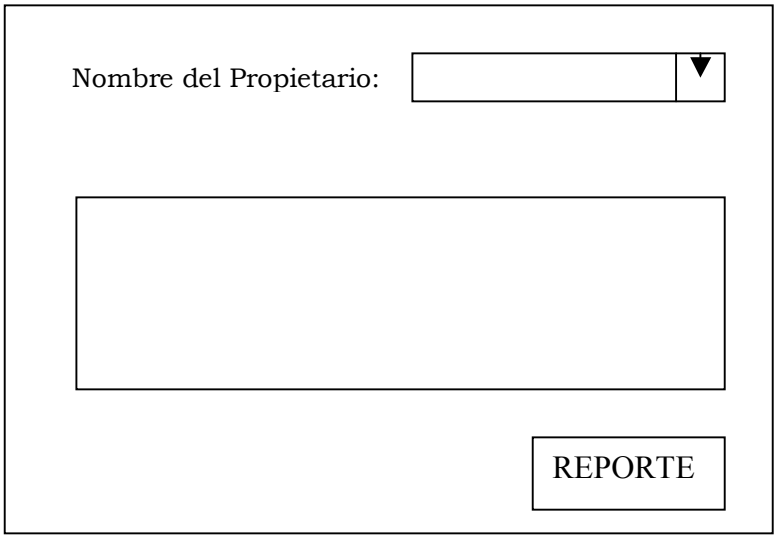


Requerimientos de Salida.

Requerimiento 1:0: Página de Consulta Propietario.

Esta página realizará consultas sobre las personas a las que les fue incautada mercancía mencionando cada una de estas con sus respectivas características, ejemplo 5.

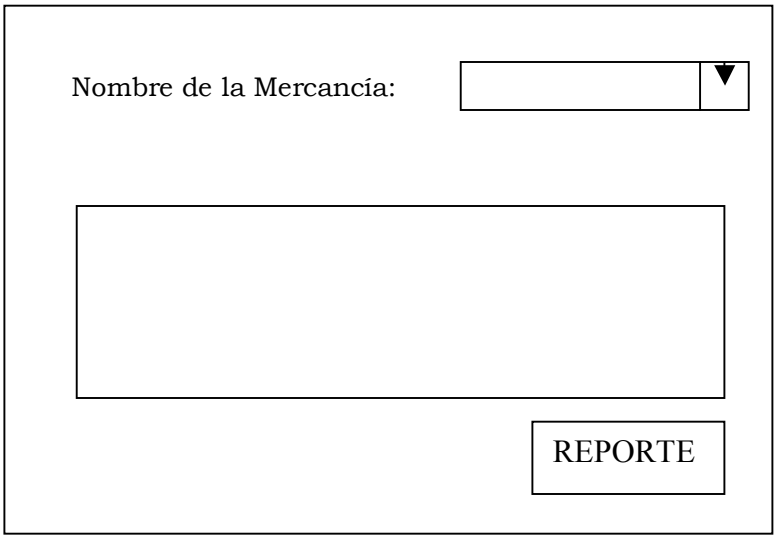
Ejemplo 5.



Requerimiento 2:0: Página de Consulta Mercancía.

Aquí se realizarán consultas sobre las mercancías que han sido incautadas desplegando todas sus características, ejemplo 6.

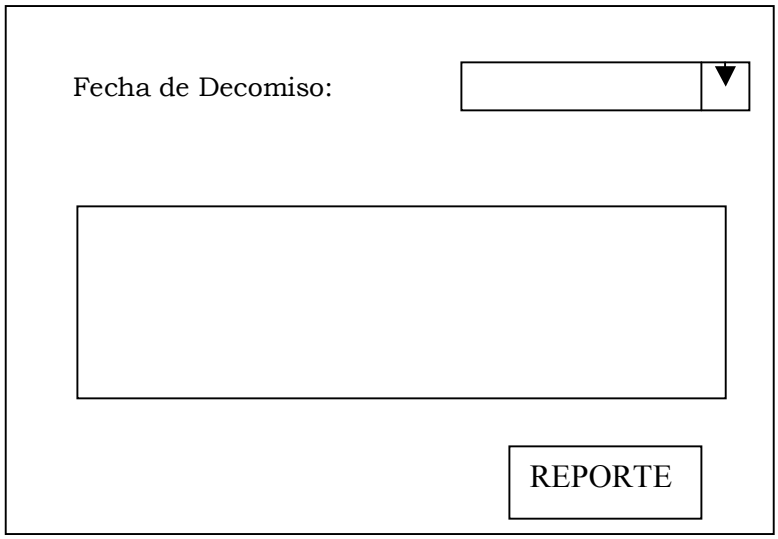
Ejemplo 6.



Requerimiento 3:0: Página de Consulta por Fechas.

En esta página se podrán visualizar las consultas que los usuarios y administradores deseen realizar sobre alguna mercancía que se encuentre incautada, la cual no cuente con alguna otra característica que no sea la fecha, en este lugar podrá realizar dicha consulta, únicamente escogiendo la fecha en la cual se realizo el decomiso, ejemplo 7.

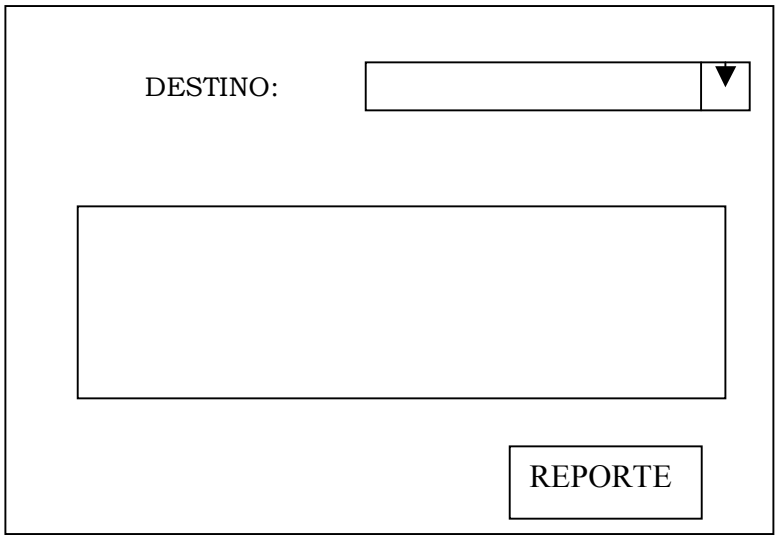
Ejemplo 7.



Requerimiento 4:0: Página de Consulta Destino.

En esta página se podrán observar consultas sobre los destinos que han tenido las mercancías, de acuerdo a los diferentes recintos fiscales, ejemplo 8.

Ejemplo 8.



4.4 DICCIONARIO DE DATOS.

El diccionario de datos proporciona información adicional sobre el sistema mostrando una lista de todos los elementos incluidos. Los elementos principales en un sistema son, el flujo de datos, el almacenamiento de datos y los procesos. El diccionario de datos almacena detalles y descripciones de estos elementos.

Si los analistas o usuarios quieren saber cuántos caracteres tiene un dato, donde se utilizan en el sistema aquí es donde se encontrará esta información, véase figura 4.2.

Figura 4.2: Diccionario de Datos.

Identificación.	Nombre.	Tipo de Dato.	Longitud.	Permite Nulos.	Requerimiento de Origen.
Iddepartamento	Número de departamento	Entero	4	No	Departamento.
Nombre	Nombre del departamento.	Varchar	30	Si	Departamento.
Idusuario	Número de Usuario.	Entero	4	No	Usuario.
Nombre	Nombre del usuario.	Varchar	60	Si	Usuario.
Apellido	Apellido del usuario.	Varchar	60	Si	Usuario.
Teléfono	Teléfono del usuario.	Varchar	30	Si	Usuario.

Continúa Página Siguiente.

Identificación.	Nombre.	Tipo de Dato.	Longitud.	Permite Nulos.	Requerimiento de Origen.
Email	Email del usuario.	Varchar	40	Si	Usuario.
RFC	RFC del usuario.	Varchar	10	Si	Usuario.
Password	Password del usuario	Varchar	20	Si	Usuario.
Tipousuario	Tipo de usuario asignado	Varchar	20	Si	Usuario.
Iddestino	Número de desatino	Entero	4	No	Destino.
Nombre	Nombre del Destino	Varchar	30	Si	Destino.
Cantidad	Cantidad de mercancía enviada al destino	Varchar	20	Si	Destino.
Fechaenvio	Fecha de envió de la mercancía a su destino	Varchar	20	Si	Destino.
Idmercancia	Número de la mercancía	Entero	4	No	Mercancía
Nombre	Nombre de la mercancía	Varchar	60	Si	Mercancía
Marca	Marca de la Mercancía	Varchar	60	Si	Mercancía

Continúa Página Siguiente.

Identificación.	Nombre.	Tipo de Dato.	Longitud.	Permite Nulos.	Requerimiento de Origen.
Modelo	Modelo de la Mercancía	Varchar	30	Si	Mercancía
Color	Color de la mercancía	Varchar	20	Si	Mercancía
Año	Año de la mercancía	Varchar	10	Si	Mercancía
Estado	Estado de la mercancía	Varchar	20	Si	Mercancía
Piezas	Cantidad de Piezas de la mercancía	Varchar	20	Si	Mercancía
Precio	Precio estimado de la mercancía	Numérico	9	si	Mercancía
Ubicación	Ubicación exacta de la mercancía	Varchar	30	Si	Mercancía
Foto	Foto de la mercancía	Varchar	80	Si	Mercancía
Matricula	Matricula vehículo	Varchar	59	si	Mercancía
Observaciones	Campo para inf. adicional	Varchar	200	Si	Mercancía
Fechadecomiso	Fecha del decomiso	Varchar	20	Si	Mercancía.

Continúa en la siguiente Página

Identificación.	Nombre.	Tipo de Dato.	Longitud.	Permite Nulos.	Requerimiento de Origen.
Idpropietario	Número de Propietario	Entero	4	No	Propietario
Nombre	Nombre del propietario	Varchar	60	Si	Propietario
Apellido	Apellido del propietario	Varchar	60	Si	Propietario
Domicilio	Domicilio del propietario	Varchar	60	Si	Propietario
Ciudad	Es la ciudad de origen del propietario	Varchar	50	si	Propietario
Estado	Es el estado de origen del propietario	Varchar	30	Si	Propietario
Teléfono	Es el teléfono del propietario	Varchar	25	Si	Propietario
Email	Correo electrónico del propietario	Varchar	30	si	Propietario

Fuente: Tabla creada por el Autor.

4.5 BASE DE DATOS.

El siguiente código es la forma en que se realizará la Base de Datos utilizando SQL-Server:

CREATE DATABASE TITULO

De esta forma ha sido creada una Base de Datos con el nombre de TITULO, a continuación se mostrará el código para crear las Tablas con sus respectivos campos de acuerdo al diccionario de datos.

Este es el código para crear la tabla Departamento con sus respectivos campos:

```
Create table Departamento (
Iddepartamento integer not null identity (1, 1) primary key,
Nombre varchar (30),
)
```

Aquí se muestra el código de la tabla Usuario:

```
Create table Usuario (
Idusuario integer not null identity (1, 1) primary key,
Nombre varchar (60),
Apellido varchar (60),
Teléfono varchar (30),
Email varchar (40),
Password varchar (20),
Tipousuario varchar (20)
)
```

Lo siguiente se refiere a la tabla Destino:

```
Create Table Destino (
Iddestino integer not null identity (1, 1) primary key,
Nombre varchar (30),
Cantidad varchar (20),
Fechaenvio varchar (20),
Idmercancia integer,
Idusuario integer,
Constraint kfusr foreign Key (idusuario) references Usuario (Idusuario),
Constraint fkmer foreign key (Idmercancia) references Mercancía (idMercancia)
)
```

A continuación tenemos lo correspondiente a la tabla Mercancía:

```
Create table Mercancia (
Idmercancia integer not null identity (1, 1) primary key,
Fechadecomiso varchar (20),
Nombre varchar (60),
Marca varchar (60),
Matricula varchar (60),
Modelo varchar (30),
Color varchar (20),
Año varchar (10),
Estado varchar (20),
Piezas varchar (20),
Precio varchar (20),
Ubicación varchar (30),
Foto varchar (80),
Observaciones varchar (60),
)
```

Y por último tenemos el código de la tabla Propietario:

```
Create table Propietario (
Idpropietario integer not null identity (1, 1) primary key,
Nombre varchar (60),
```

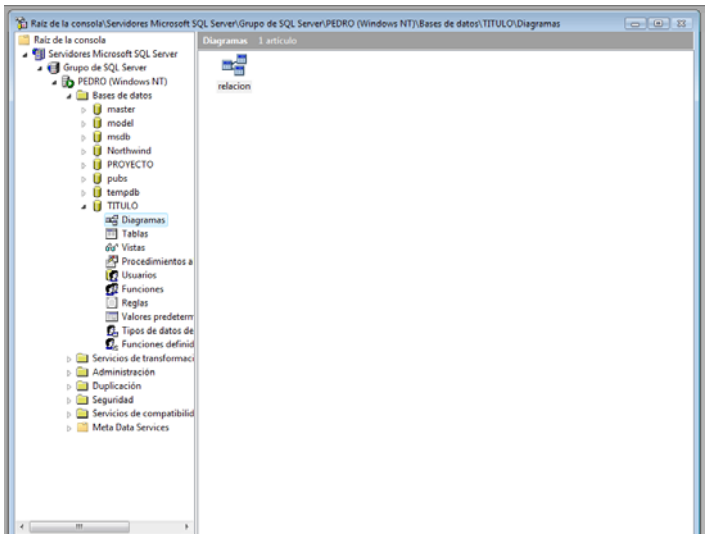
Apellido varchar (60),
Domicilio varchar (60),
Ciudad varchar (50),
Estado varchar (30),
Telefono varchar (25),
Email varchar (30),
)

Después de introducir estos códigos en el analizador de consultas, las tablas mencionadas son creadas con sus respectivos campos, respetando cada uno de los tipos de datos y longitudes que fueron asignados en el código

4.5.1 Diagrama de la Base de Datos.

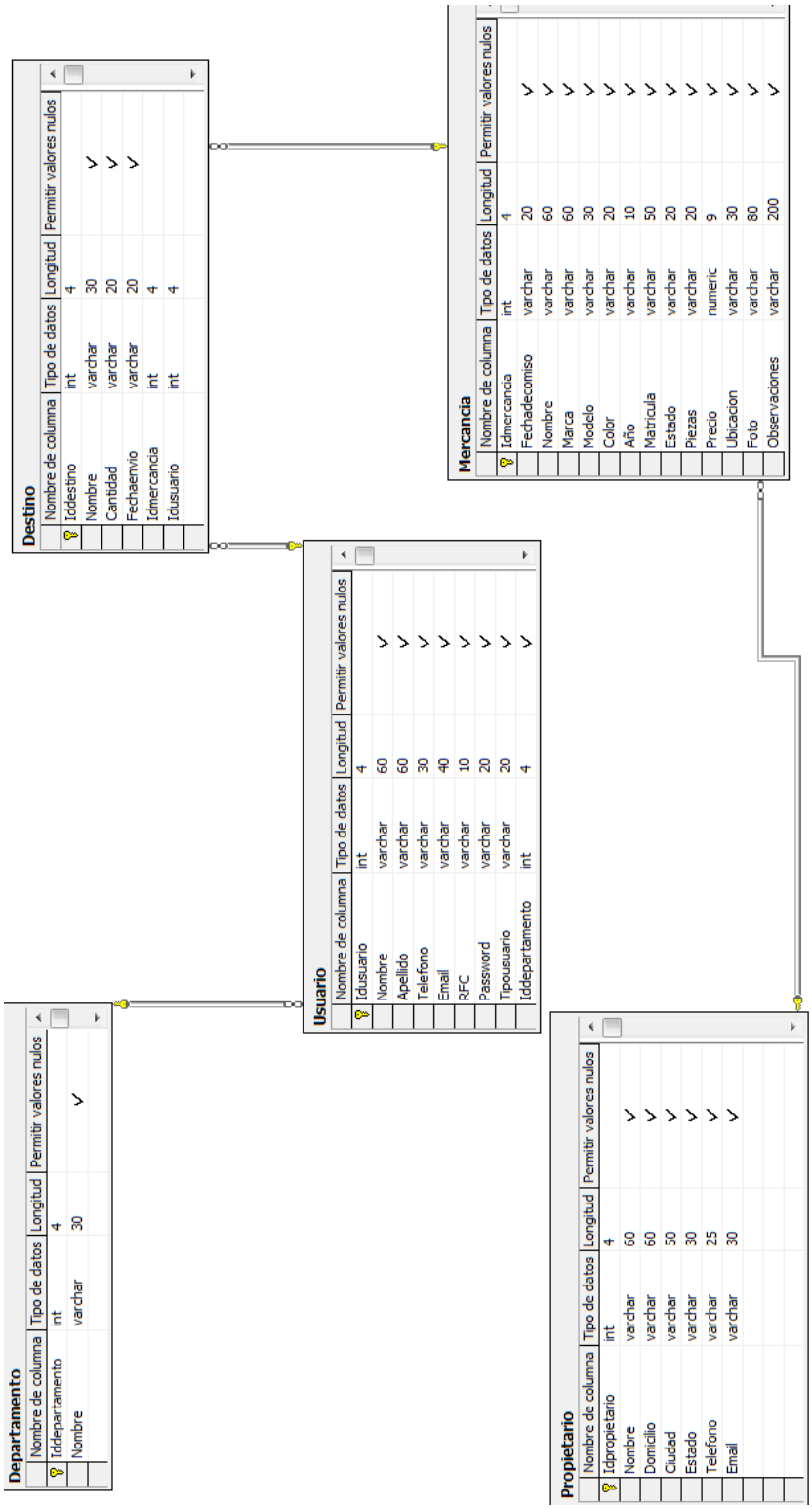
Puede ser observado en el área de diagramas de la base de datos creada en el administrador corporativo, véase figura 4.3. A continuación se podrá apreciar el diagrama de la base de datos creada anteriormente, véase figura 4.4.

Figura 4.3: Área de Diagramas.



Fuente: Diagrama creado por el Autor.

Figura 4.4: Diagrama de la Base de Datos creada.



Fuente: Diagrama creado por el Autor.

4.6 DISEÑO DE PÁGINAS WEB.

El diseño Web consiste en la planificación, diseño e implementación de páginas Web. No es simplemente una aplicación del diseño convencional sobre Internet ya que requiere tener en cuenta cuestiones tales como buena interactividad con el usuario.

Un buen diseño aumenta la eficacia de la página como medio de comunicación e intercambio de datos, además de que ofrece contacto directo entre el realizador y el consumidor de contenidos, característica destacable en el medio Internet.

4.7 PÁGINA INICIO.

Figura 4.5: Página de Inicio del SACM.

Fuente: Imagen creada por el Autor.

En la página de inicio (véase figura anterior) del SACM, se introducirá la información solicitada que es el Usuario y su Password para poder tener acceso al sistema de acuerdo al nivel que haya sido asignado, si no cuenta con una cuenta no logrará entrar al sistema.

4.7.1 Menú Administrador.

Si el usuario tiene nivel de administrador le aparecerá la siguiente pantalla, véase figura 4.6; la cual muestra un menú con las siguientes opciones: **Registrar, Decomisos, Consultas** y **Destino** a las cuales tendrá acceso el administrador.

Figura 4.6: Página de Menú del Administrador.

Fuente: Imagen creada por el Autor.

4.7.2 Página de Registro.

Para dar de alta a un usuario es necesario tener nivel de administrador, de otro forma no va ha ser posible dar de alta a nuevos usuarios, dentro del acceso del administrador en su menú contamos con una selección llamada registro, véase figura 4.7.

Figura 4.7: Menú del Administrador con la opción de Registrar.

Fuente: Imagen creada por el Autor.

Al momento de dar click en la opción de Registro aparecerá la siguiente pantalla, véase figura 4.8.

Figura 4.8: Página de Registro de Usuarios.

Fuente: Imagen creada por el Autor.

En esta página de deben de ingresar todos los datos que se piden, de lo contrario aparecerá un mensaje en el lugar que falte la información solicitada con la leyenda de **Requerido**, en el campo de tipo de usuario, únicamente son dos valores los que deben ser introducidos que son **Administrador** y **Usuario**, además en el apartado de Departamento se podrá seleccionar cualquiera de las dos opciones que marca, una vez introducidos todos los datos se debe dar un click en el botón de Registrar y los datos se almacenarán en la base de datos, cabe señalar que únicamente los administradores podrán registrar nuevos usuarios.

4.7.3 Página Decomisos.

En el menú del administrador contamos con una opción llamada Decomisos, véase figura 4.9.

Figura 4.9: Menú del Administrador con la opción de Decomisos.

Fuente: Imagen creada por el Autor.

En esta parte los administradores darán de alta a los propietarios de la mercancía incautada, véase figura 4.10 llenando los campos con los datos correspondientes, algunos de los campos no son obligatorios debido a que alguna veces no se cuentan con ellos o se niegan darlos; cuando ya se tiene la información se deberá dar click en el botón de ingreso, el cual introducirá los datos en la base de datos en la tabla correspondiente, después aparecerá un mensaje con la leyenda que el propietario esta registrado, a continuación se debe dar click en el botón

de introducir mercancía el cual abrirá una nueva página véase figura 4.11., donde se registrará la mercancía del propietario que acaba de ser registrado.

Figura 4.10: Página de Datos del Propietario.

Fuente: Imagen creada por el Autor.

Figura 4.11: Página de Datos de la Mercancía.

Fuente: Imagen creada por el Autor.

En esta página se introducirán las características de cada una de las mercancías, así como su cantidad, estado, marca, modelo, color, su fotografía, la fecha en que se realiza el decomiso, además se podrá escoger el propietario, dueño de dicha mercancía, también cuenta con un campo donde el administrador podrá hacer observaciones, si este lo considera apropiado una vez llenado los campos se debe dar click en el botón de ingresar mercancía para que esta sea guardada en la base de datos.

4.7.4 Página Consultas.

El menú administrador cuenta con la opción de consultas, véase figura 4.12, que a su vez cuenta con un submenú con diferentes opciones, a esta página también tienen acceso los usuarios de menor nivel.

Figura 4.12: Menú del Administrador con la opción de Consultas.

Fuente: Imagen creada por el Autor.

Opción Mercancía.

En esta parte tanto los administradores como los usuarios de más bajo nivel tendrán acceso para realizar las consultas de acuerdo a los nombres de las mercancías, dicho nombre se podrá escoger de una lista, cuando sea escogido el nombre de la mercancía se deberá dar click en el botón de buscar e inmediatamente aparecerá una tabla con las mercancías y sus características las cuales pueden ser modificadas o borradas, véase figura 4.13.

Figura 4.13: Página de Consulta de Mercancías.

Fuente: Imagen creada por el Autor.

También en el submenú de consultas se tiene la opción del propietario, véase figura 4.14.

Figura 4.14: Menú del Administrador con la opción de Propietario.

Fuente: Imagen creada por el Autor.

Opción Propietario.

En este caso se tiene la opción de conocer a todos los propietarios de mercancía incautada con cada una de sus referencias; al igual que en la consulta anterior únicamente se debe seleccionar un propietario en particular para que aparezca una tabla con todos los datos del propietario que ha sido escogido y también podrán ser modificados o borrados, véase figura 4.15.

Figura 4.15: Página de Consulta por Propietarios.

Fuente: Imagen creada por el Autor

Opción Fecha.

Esta opción también se encuentra en el submenú de consultas con los datos de la fecha en que fue incautada la mercancía, véase figura 4.16.

Figura 4.16: Menú del Administrador con la opción de Fecha.

Fuente: Imagen creada por el Autor.

En esta página como en las anteriores se podrá hacer una consulta donde los datos podrán ser modificados, únicamente que aquí la consulta se realizará por medio de la fecha en que fue incautada la mercancía, debido a que algunas veces los propietarios se niegan a dar sus datos personales, véase figura 4.17.

Figura 4.17: Página de Consulta por Fechas.

Fuente: Imagen creada por el Autor.

4.7.5 Página Destino.

El menú del administrador también cuenta con una opción llamada destino, véase figura 4.18.

Figura 4.18: Menú del Administrador con la opción de Destino.

Fuente: Imagen creada por el Autor.

Esta página brindará la opción a los administradores de este sistema, enviar la mercancía que sea necesaria o que haya cumplido con su proceso legal a los diferentes recintos fiscales, a donación o en su defecto a la destrucción de esta, en esta página se introducirá el nombre del destino, la fecha en que se enviará, la cantidad de piezas, además se escogerá el número de la mercancía que se le fue asignado al momento de su incautación y también el número del propietario al que pertenece dicha mercancía, véase figura 4.19.

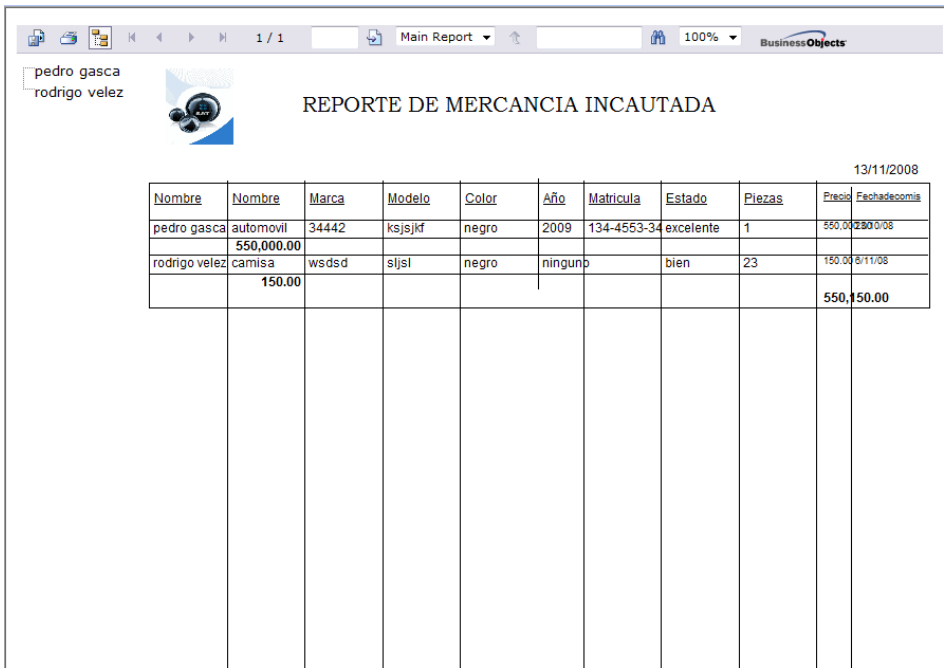
Figura 4.18: Página de Destino.

Fuente: Imagen creada por el Autor.

4.7.6 REPORTE DE MERCANCÍA INCAUTADA.

En este lugar se observará un reporte de la mercancía incautada con el nombre de los propietarios de cada una de estas y sus respectivas características, además contará con un área específica que manejará el costo estimado por el SAT, véase figura 4.19.

Figura 4.19: Reporte de Mercancía Incautada.



Fuente: Imagen creada por el Autor.

CONCLUSIONES.

Derivado del estudio y aplicación realizada se ha notado la importancia que ha tenido Internet a nivel laboral. El sistema realizado es una clara muestra de esto, ya que de esta forma se ha logrado una gran disminución de tiempo y trabajo al momento de registrar la información de los propietarios así como las descripciones de sus mercancías que sean embargadas y así lograr una gran disminución de horas hombre que antes se perdían.

Además se logró obtener de una manera rápida y sencilla el destino y ubicación de las mercancías que se encuentran en los recintos fiscales o que fueron donadas además de realizar consultas de una forma más fácil para conocer con precisión, rapidez y eficiencia la mercancía que se encuentra incautada así como de los datos de los propietarios.

También una consulta realizada por fecha de decomiso en caso de que no se cuente con los generales del propietario ya que no siempre son proporcionados y de esta manera se sigue teniendo un control de la mercancía, logrando con esto la finalidad con la que se realizó este estudio.

BIBLIOGRAFIA.

Kendall & Kendall; Análisis y Diseño de Sistemás; 3ª edición; Prentice Hall Hispanoamérica, México 1997, 913 Pág.

Roger S. Pressman; Ingeniería del Software; 4ª edición; McGraw Hill, España 1998, 518 Pág.

OTRAS FUENTES.

http://www.wikipedia.com

http://www.probandocodigo.com

http://lawebdelprogramador.com

http://manuales.com

http://monografias.com

http://jorgesaavedra.wordpress.com

www.sat.gob.mx

www.asp.net

www.cefp.gob.mx

http://msdn.microsoft.com

Operadores de comparación.

http://technet.microsoft.com/es-es/library/ms188074.aspx

Tipos de Datos.

http://www.kys.programacion.com/asp/tutorial/drh_vbscript/2

Tutorial ASP.

http://www.asptutor.com/asp/vart.asp?id=44#funciones

Servidor de Nombres de Dominio (DNS) dinámico.

http://www.fismat.umich.mx/~emurguia/mipagina/tesis

Apuntes.