UNIVERSIDAD NACIONAL AUTÒNOMA DE MÈXICO

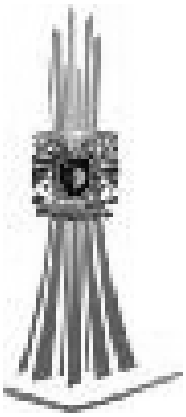
FACULTAD DE ESTUDIOS SUPERIORES
ARAGON

INFORME DE MEMORIA DE DESEMPEÑO DE SERVICIO SOCIAL EN LA AUDITORIA SUPERIOR DE LA FEDERACIÓN

MEMORIA DE DESEMPEÑO DE SERVICIO SOCIAL

QUE PARA OBTENER EL TITULO DE:
INGENIERO EN COMPUTACIÒN
PRESENTA:

RIVAS MARTÌNEZ SAÙL



DIRECTOR: ING. ENRIQUE GARCIA GUZMAN

MÈXICO 2009

ÍNDICE

I.-INTRODUCCIÓN

Qué es el Servicio Social, desde el Artículo de la Constitución Política de los Estados Unidos Mexicanos.

1.1 CONSTITUCION POLITICA DE LOS ESTADOS UNIDOS MEXICANOS

TITULO PRIMERO
CAPITULO I DE LAS GARANTIAS INDIVIDUALES
Artículo 5

Articulo 5o.- a ninguna persona podrá impedirse que se dedique a la profesión, industria, comercio o trabajo que le acomode, siendo licitos. El ejercicio de esta libertad solo podrá vedarse por determinación judicial, cuando se ataquen los derechos de tercero, o por resolución gubernativa, dictada en los términos que marque la ley, cuando se ofendan los derechos de la sociedad. Nadie puede ser privado del producto de su trabajo, sino por resolución judicial.
(Reformado mediante decreto publicado en el diario oficial de la federación el 31 de diciembre de 1974)

La ley determinara en cada estado cuales son las profesiones que necesitan titulo para su ejercicio, las condiciones que deban llenarse para obtenerlo y las autoridades que han de expedirlo.
(reformado mediante decreto publicado en el diario oficial de la federación el 31 de diciembre de 1974. Modificado por la reimpresión de la constitución, publicada en el diario oficial de la federación el 6 de octubre de 1986)

Nadie podra ser obligado a prestar trabajos personales sin la justa retribución y sin su pleno consentimiento, salvo el trabajo impuesto como pena por la autoridad judicial, el cual se ajustara a lo dispuesto en las fracciones i y ii del articulo 123.
(reformado mediante decreto publicado en el diario oficial de la federación el 31 de diciembre de 1974)

En cuanto a los servicios públicos, solo podran ser obligatorios, en los términos que establezcan las leyes respectivas, el de las armas y los jurados, así como el desempeño de los cargos concejiles y los de elección popular, directa o indirecta. Las funciones electorales y censales tendrán carácter obligatorio y gratuito, pero serán retribuidas aquellas que se realicen profesionalmente en los términos de esta constitución y las leyes correspondientes. Los servicios profesionales de índole social serán obligatorios y retribuidos en los términos de la ley y con las excepciones que esta señale.
(reformado mediante decreto publicado en el diario oficial de la federación el 06 de abril de 1990)

El estado no puede permitir que se lleve a efecto ningún contrato, pacto o convenio que tenga por objeto el menoscabo, la perdida o el irrevocable sacrificio de la libertad de la persona por cualquier causa.
(reformado mediante decreto publicado en el diario oficial de la federación el 28 de enero de 1992)

Tampoco puede admitirse convenio en que la persona pacte su proscripción o destierro, o en que renuncie temporal o permanentemente a ejercer determinada profesión, industria o comercio.
(reformado mediante decreto publicado en el diario oficial de la federación el 31 de diciembre de 1974)

El contrato de trabajo solo obligara a prestar el servicio convenido por el tiempo que fije la ley, sin poder exceder de un año en perjuicio del trabajador, y no podrá extenderse, en ningún caso, a la renuncia, perdida o menoscabo de cualquiera de los derechos políticos o civiles.
(reformado mediante decreto publicado en el diario oficial de la federación el 31 de diciembre de 1974)

La falta de cumplimiento de dicho contrato, por lo que respecta al trabajador, solo obligara a este a la correspondiente responsabilidad civil, sin que en ningún caso pueda hacerse coacción sobre su persona.
(reformado mediante decreto publicado en el diario oficial de la federación el 31 de diciembre de 1974)

1.2 REGLAMENTO GENERAL DEL SERVICIO SOCIAL DE LA UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

El Consejo Universitario, en sesión del 26 de septiembre de 1985, aprobó este Ordenamiento en los siguientes términos:
CAPÍTULO I
DISPOSICIONES GENERALES
ARTÍCULO 1º.- El presente reglamento establece las bases y fija los lineamientos para la prestación del servicio social de los estudiantes de la Universidad Nacional Autónoma de México y de las escuelas incorporadas, de conformidad con los artículos 52 de la Ley Reglamentaria de los artículos 4º y 5º, Constitucionales, y 85 de su reglamento.
ARTÍCULO 2º.- El servicio social se regulará por los lineamientos generales establecidos en el presente reglamento y por las normas de la Legislación Universitaria que se señalan a continuación:
I. Reglamento General de Estudios Técnicos y Profesionales;
II. Reglamento General de Exámenes,
III. Y los reglamentos internos que para cada facultad o escuela dicten los consejos técnicos correspondientes.
ARTÍCULO 3º.- Se entiende por servicio social universitario la realización obligatoria de actividades temporales que ejecuten los estudiantes de carreras técnicas y profesionales, tendientes a la aplicación de los conocimientos que hayan obtenido y que impliquen el ejercicio de la práctica profesional en beneficio o en interés de la sociedad.
ARTÍCULO 4º.- El servicio social tiene por objeto:
I. Extender los beneficios de la ciencia, la técnica y la cultura a la sociedad;
II. Consolidar la formación académica y capacitación profesional del prestador del servicio social, y
III. Fomentar en el prestador una conciencia de solidaridad con la comunidad a la que pertenece.

CAPÍTULO II
REQUISITOS Y CARACTERÍSTICAS DEL SERVICIO SOCIAL
ARTÍCULO 5º.- De conformidad con los artículos 52 y 55 de la Ley Reglamentaria de los artículos 4º y 5º Constitucionales, los estudiantes de la Universidad Nacional Autónoma de México y los de las escuelas incorporadas deberán prestar su servicio social como requisito previo para la obtención del título profesional.
ARTÍCULO 6º.- El servicio social deberá prestarse durante un tiempo no menor de 6 meses ni mayor de 2 años y el número de horas que requiera será determinado por las características del programa al que se encuentre adscrito el estudiante, pero en ningún caso será menor de 480 horas. Los consejos técnicos propondrán la forma de cómputo del mínimo de horas en el reglamento interno.
ARTÍCULO 7º.- El tiempo de duración de la prestación del servicio social deberá ser continuo a fin de lograr los objetivos señalados en el artículo 4º de este reglamento. Se entenderá que existe discontinuidad cuando sin causa justificada se interrumpa la prestación del servicio social por más de 18 días durante 6 meses, o en su caso 5 días seguidos. Los días se entienden como hábiles.
ARTÍCULO 8º.- Cuando exista discontinuidad en los términos del artículo anterior, el servicio social deberá reiniciarse sin tormarse en cuenta las actividades realizadas antes de la interrupción. Los consejos técnicos de facultades y escuelas determinarán los casos de excepción.
ARTÍCULO 9º.- Los estudiantes de la Institución realizarán su servicio social de acuerdo con los programas unidisciplinarios, interdisciplinarios o multidisciplinarios que respectivamente se aprueben.
ARTÍCULO 10.- Para que los estudiantes puedan iniciar la prestación del servicio social en necesario que tengan un mínimo del 70% de créditos de su carrera y el 100% en los casos en que lo ameriten, y que se registren y obtengan la autorización de su plantel respectivo. Las facultades o escuelas, de común acuerdo con la Comisión Coordinadora del Servicio Social, determinarán los casos excepcionales de menor porcentaje de créditos.
ARTÍCULO 11.- El servicio social podrá realizarse en todas las áreas profesionales. Sin embargo los consejos técnicos, la Comisión Coordinadora del Servicio Social y las unidades responsables de cada facultad o escuela deberán orientar la prestación del servicio social, hacia las ramas y modalidades de cada profesión que se consideren prioritarias para las necesidades del país.
ARTÍCULO 12.- Los programas del servicio social, podrán ser carácter interno en la Universidad Nacional Autónoma de México y externo en el sector público y social.
ARTÍCULO 13.- La prestación del servicio social, por ser éste en beneficio de la comunidad, no creará derechos ni obligaciones de tipo laboral.
ARTÍCULO 14.- La retribución del servicio social, se apegará a lo dispuesto en la Ley Reglamentaria de los artículos 4º y 5º Constitucionales y su reglamento.
ARTÍCULO 15.- Los prestadores del servicio social no tendrán derecho a ayuda económica cuando sean trabajadores y disfruten de licencia con goce de salario para tal efecto.
CAPÍTULO III
DE LA ORGANIZACIÓN Y PROCEDIMIENTOS DEL SERVICIO SOCIAL UNIVERSITARIO
ARTÍCULO 16.- En la organización del servicio social universitario intervendrán:
I. Los consejos técnicos de las facultades y escuelas;

II. La Comisión Coordinadora del Servicio Social, y
III. Las unidades responsables del servicio social en cada una de las facultades y Escuelas.
ARTÍCULO 17.- Corresponde a los consejos técnicos de las facultades y escuelas:
I. Establecer las modalidades para el cumplimiento del servicio social en cada una de las facultades y escuelas a través de los respectivos reglamentos internos, y
II. Proponer programas interdisciplinarios y multidisciplinarios, así como ejercer las demás facultades que deriven del presente reglamento.
ARTÍCULO 18.- Las unidades responsables del servicio social de las facultades y escuelas tienen las siguientes funciones y actividades:
I. Planear los programas de su facultad o escuela;
II. Fijar los criterios para la asignación de prestadores a los programas de servicio social;
III. Aprobar, promover, supervisar y evaluar la realización de programas de servicio social;
IV. Controlar la prestación del servicio social de los estudiantes de su facultad y escuela y llevar los siguientes registros correspondientes, y
V. Extender el certificado de cumplimiento del servicio social de los estudiantes de su facultad o escuela o validarlo en su caso.
ARTÍCULO 19.- La Comisión Coordinadora del Servicio Social dependerá de la Secretaría de la Rectoría y será presidida por un coordinador nombrado y removido libremente por el Rector.
ARTÍCULO 20.- La Comisión Coordinadora del Servicio Social tiene las siguientes atribuciones:
I. Coordinar la prestación del servicio universitario;
II. Establecer vínculos con el sector público y social con el fin de celebrar convenios para prestación del servicio social;
III. Elaborar y proponer programas interdisciplinarios y multidisciplinarios, así como proponer los criterios para la adscripción de los prestadores del servicio social a cada programa;
IV. Mantener relaciones con las unidades responsables del servicio social en las facultades y escuelas para realizar labores conjuntas de planeación, promoción y apoyo del servicio social;
V. Coordinar, con las unidades responsables del servicio social en las facultades y escuelas, la integración de las brigadas que realizarán los programas multidisciplinarios del servicio social;
VI. Supervisar y evaluar cuando proceda la realización de los programas multidisciplinarios del servicio social y remitir la información correspondiente a las unidades responsables de las facultades y escuelas para su certificación;
VII. Determinar y especificar las normas relativas al servicio social que efectúen los estudiantes de las escuelas incorporadas a la Universidad Nacional de México, normas cuyo cumplimiento supervisará la Dirección de General de Incorporación y Revalidación de Estudios;
VIII. Presentar a la Dirección General de Profesiones anualmente los planes y programas del servicio social, y
IX. Las demás que le establezcan en este reglamento.
ARTÍCULO 21.- Son obligaciones de los prestadores del servicio social:

I. Inscribirse en los programas de servicio social previamente aprobados por los órganos competentes. Para tal efecto deberán realizar los trámites administrativos que sean establecidos por su facultad o escuela y en su caso por la Comisión Coordinadora del Servicio Social;
II. Realizar las actividades señaladas en el programa al cual estén adscritos, y
III. Informar periódicamente de sus actividades en los términos que señale su facultad o escuela y en su caso la Comisión Coordinadora del Servicio Social.

ARTÍCULO 22.- Los responsables del servicio social en las facultades y escuelas y en su caso la Comisión Coordinadora del Servicio Social evaluarán la prestación del servicio por parte de los estudiantes una vez que concluyan su servicio social para comprobar el cumplimiento de las actividades programadas. En caso de ser satisfactoria la prestación del servicio social, se procederá a certificarlo. En caso contrario indicarán al estudiante las actividades complementarias que estimen convenientes para poder otorgarle la certificación.

CAPÍTULO IV
DEL SERVICIO SOCIAL EN LAS ESCUELAS DE ENSEÑANZA SUPERIOR CON ESTUDIOS INCORPORADOS A LA UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
ARTÍCULO 23.- El servicio social que presten los estudiantes de las escuelas con estudios incorporados a la Universidad Nacional Autónoma de México deberá realizarse conforme a las disposiciones que se establecen en el presente reglamento.
ARTÍCULO 24.- Las escuelas con estudios incorporados a la Universidad Nacional Autónoma de México deberán contar con un responsable del servicio social, aprobado por la Dirección General de Incorporación y Revalidación de Estudios, según normas de la Comisión Coordinadora del Servicio Social. También deberán cumplir con las normas que par cada carrera se aprueben por los diferentes órganos competentes de la UNAM.
ARTÍCULO 25.- Los responsables del servicio social de las escuelas incorporadas deberán enviar la documentación relativa al servicio social de sus estudiantes, con el fin de que la Dirección General de Incorporación y Revalidación de Estudios esté en posibilidad de aprobarlo y supervisarlo e informar sobre ello a la Comisión Coordinadora del Servicio Social.
Artículos Transitorios
ARTÍCULO PRIMERO.- Este reglamento entrará en vigor a partir de la fecha de su publicación en la *Gaceta UNAM.*
ARTÍCULO SEGUNDO.- Quedan derogadas todas las disposiciones de los reglamentos de las facultades o escuelas que contravengan al presente ordenamiento.
ARTÍCULO TERCERO.- Los estudiantes y pasantes que se encuentren presentando su servicio social a la fecha de la entrada en vigor del presente reglamento podrán concluirlo de conformidad con las disposiciones aplicables anteriormente.
ARTÍCULO CUARTO.- Quienes deseen acogerse a la disposición del artículo 91 del Reglamento de la Ley Reglamentaria del Artículo 5º Constitucional deberán cumplir los trámites y requisitos que señale su facultad o escuela.

1.3 TITULACION POR MEMORIA DE DESEMPEÑO DE SERVICIO SOCIAL

Modalidad individual para egresados que hayan empezado su servicio social una vez concluida la carrera y que comprueben haber realizado su Servicio Social en actividades de apoyo a la comunidad o en alguna institución pública o privada que haya establecido convenios con la UNAM, en un proyecto vinculado con el área específica, donde a juicio del Comité de Planes y Programas de Estudio y Titulación el solicitante haya aplicado el criterio, conocimientos y habilidades propios de un profesional de la ingeniería.

Como requisito cubierto para poder acceder a la modalidad de titulación el servicio social debe ser comenzado cuando se han cubierto el 100% de créditos.

1.4 PLAN DE ESTUDIOS FES ARAGON 1992

Formar profesionales en Ingeniería en Computación líderes, con conocimientos teóricos y prácticos útiles para la solución de problemas computacionales que la sociedad demanda, comprometidos con las necesidades y desarrollo del país.
Perfil Del Egresado (Plan 1992)

El perfil del egresado es el de un profesional con conocimientos sólidos en sistemas de programación (software), así como la aplicación de esos conocimientos en diferentes áreas con las que interactúa, las cuales le permitirán responder a las diversas necesidades que se presenten en el campo de trabajo de la Ingeniería en Computación. De acuerdo con lo anterior el egresado tendrá las siguientes características:

1. Conocimientos:
    - Poseer conocimientos sólidos de las matemáticas que le permitan el modelado de sistemas físicos.
    - Tener conocimientos básicos en todos los campos de la ingeniería en computación para la solución de problemas reales.
    - Contar con una formación metodológica, apoyada en el método científico y en la teoría general de sistemas.
    - Adquirir los conocimientos de vanguardia generados en las ciencias de la Computación.
    - Comprender por lo menos una lengua extranjera.
2. Habilidades:
    - Capacidad para diseñar, construir, operar y mantener sistemas de cómputo y de programación, contemplando el aseguramiento de la calidad de los mismos.
    - Manejar las técnicas y lenguajes de programación que el apoyen en la solución y programación de problemas reales.
    - Manejar eficientemente la información mediante el uso de la computadora.
    - Evaluar, comparar y seleccionar equipos de cómputo.
    - Diseñar e instalar redes de teleinformática.
    - Conceptualizar, planear, diseñar, construir, operar y mantener sistemas automáticos de control digital para la industria.

- Desarrollar nuevos lenguajes de computadora.
- Diseñar y construir sistemas de interfaz máquina-máquina, hombre-máquina y máquina-hombre.
- Resolver problemas con orientación teórica tales como: diseño de autómatas, modelado de estructuras de datos, desarrollos de sistemas operativos, desarrollo de manejadores de bases de datos, compiladores.
- Organizar, dirigir y administrar centros de cómputo.
- Trabajar conjuntamente con otros especialistas en la solución de problemas en otros campos de acción.
- Comunicar en forma verbal y escrita los resultados de su actividad.

3. Actitudes:
    - Actualizar los conocimientos y prácticas de acuerdo con el avance tecnológico, a fin de permanecer constantemente en el desarrollo del arte de la computación.
    - Mantener una posición objetiva en su labor profesional, fuera de prejuicios y de presiones por interés particulares.
    - Tener respeto e interés por la cultura.
    - Desarrollar su actividad profesional con un sentido de servicio social y con apego a la ética.

1.5 SERVICIO SOCIAL EN LA AUDITORIA SUPERIOR DE LA FEDERACION

Perfil requerido:

- Actitud de servicio y capacidad para solución de problemas.
- Deseos de aprender.
- Buena presentación.
- Disponibilidad para trabajar en equipo
- Experiencia no necesaria.
- 70% de créditos mínimo

¿Qué ofrecen?

1. Desarrollo profesional que permita su integración al ámbito laboral.
2. Facilidades de horario matutino o vespertino, mixto.
3. Experiencia en áreas de tu interés.
4. Agradable ambiente de trabajo.

Actitud constructiva

EL personal debe mantener una actitud constructiva en las auditorías que se practiquen con motivo de la revisión de la Cuenta Pública Federal, para lo cual considerará que el fin último de la ASF es contribuir a mejorar la gestión gubernamental.

Como conviene a su carácter de Entidad de Fiscalización Superior de la Federación, la ASF está obligada a señalar todas las irregularidades y deficiencias que detecte en su revisión de la Cuenta Pública Federal, pero también a recomendar las medidas que estime pertinentes para su prevención y corrección, con el propósito de mejorar el desempeño de las entidades públicas.

Por consiguiente, al tiempo que los auditores deben aplicar rigurosamente todos los procedimientos que consideren necesarios para el cumplimiento de los objetivos de las auditorias, deben asumir una actitud positiva a fin de que en sus informes los resultados de las revisiones se presenten en su justa dimensión y se formulen las recomendaciones más idóneas.
Las recomendaciones deben ser tan precisas como se requiera para facilitar su comprensión y atención, pero corresponderá a las entidades auditadas determinar las medidas concretas para subsanar las deficiencias.
Al respecto, debe tenerse presente que no es función de la ASF asesorar a las entidades sujetas de fiscalización y que los auditores no deben participar en las decisiones de éstas, ni asumir responsabilidades de su competencia.

1.6 LA CUENTA PÚBLICA

Que es en sí misma la revisión a la cuenta pública, qué o a qué se llama cuenta pública

La Cuenta de la Hacienda Pública Federal es el informe que rinde anualmente el Poder Ejecutivo ala Honorable Cámara de Diputados para mostrar los resultados de su gestión financiera y los alcances que la acción reguladora del gobierno ha tenido en el desarrollo económico y social del país.

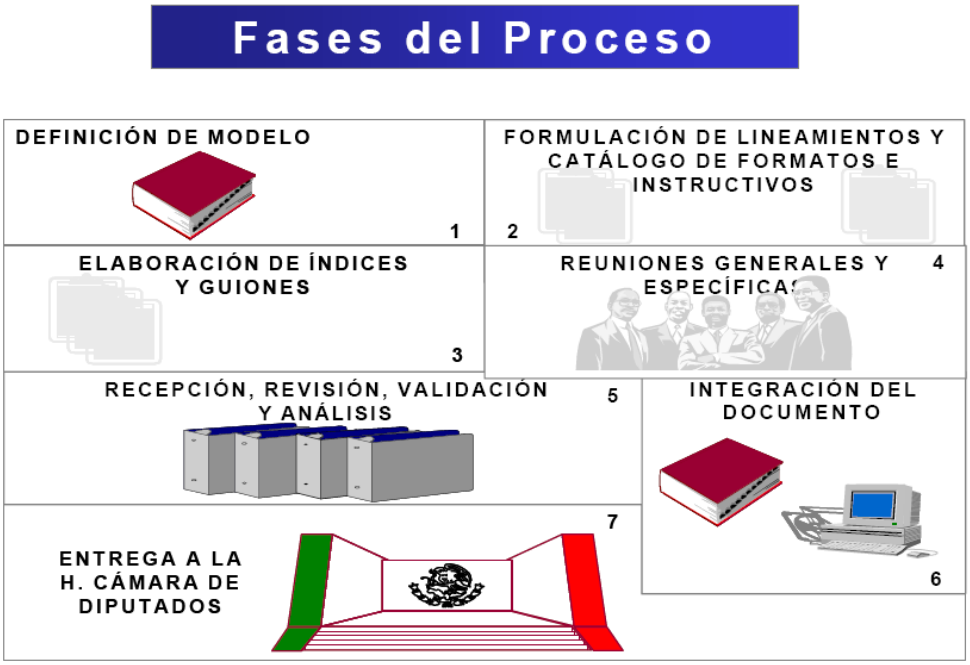
También incluye los resultados de la gestión de gasto por parte de los Poderes Legislativo y Judicial,y de los órganos autónomos como el IFE, los Tribunales Agrarios y el Tribunal Fiscal de la Federación.

Se diferencia de otros informes por su enfoque contable y financiero, y por ser el único documento que presenta los resultados definitivos de la ejecución de la Ley de Ingresos, del ejercicio del Presupuesto de Egresos y administración de la deuda del sector público presupuestario.

La gestión financiera es el control que se lleva sobre la Ley de Ingresos y del ejercicio del Presupuesto de Egresos de la Federación -PEF-, durante el lapso señalado, a través de diversos estados financieros, presupuestarios y programáticos generados por los poderes Legislativo y Judicial, los entes autónomos, las dependencias del Ejecutivo Federal, los ramos generales y las entidades paraestatales de control presupuestario directo e indirecto, bajo la normatividad y criterios metodológicos establecidos en el Sistema Integral de Contabilidad Gubernamental, lo que le otorga un sólido sustento técnico y un alto grado de confiabilidad.

1.7 ¿CÓMO SE ELABORA Y QUIÉN INTERVIENE EN LA CUENTA PUBLICA?

La Secretaría de Hacienda y Crédito Público, a través de la Unidad de Contabilidad e Informes sobre la Gestión Pública (UCGIGP) de la Subsecretaría de Egresos, es integrar la Cuenta Pública. Las actividades que se llevan a cabo se inician con la Modelo y finalizan con la entrega de la Cuenta a la H. Cámara de Diputados.



1.8 DIRECCION GENERAL DE SISTEMAS

Principales Funciones de la Dirección General de Sistemas

Establecer

Las normas y emitir las políticas en materia de informática y comunicaciones en la Auditoria Superior de la Federación, previa aprobación de su superior jerárquico

Elaborar

Estudios de factibilidad para optimizar la plataforma tecnológica de la Auditoria Superior de la Federación, mediante la adquisición, ampliación o sustitución de los recursos informáticos.

Implantar

Los sistemas de información automatizados y participar en la capacitación de su personal, para el manejo de los equipos y la operación de los programas.

Aprobar

La adquisición de equipos informáticos, licencias, paquetería, y consumibles, administrarlos y asignarlos a las Unidades Administrativas y, en su caso, solicitar su mantenimiento preventivo y correctivo, así como elaborar los dictámenes de no utilidad de los bienes y consumibles informáticos.

Dentro de la Dirección General de Sistemas hay 3 direcciones que la integran que son:

-Dirección de Tecnología.

-Dirección de Operación.

-Dirección de planeación y desarrollo de sistemas.

En la dirección de planeación y desarrollo de sistemas desarrolle mi servicio social.

ORGANIGRAMA DE LA ASF

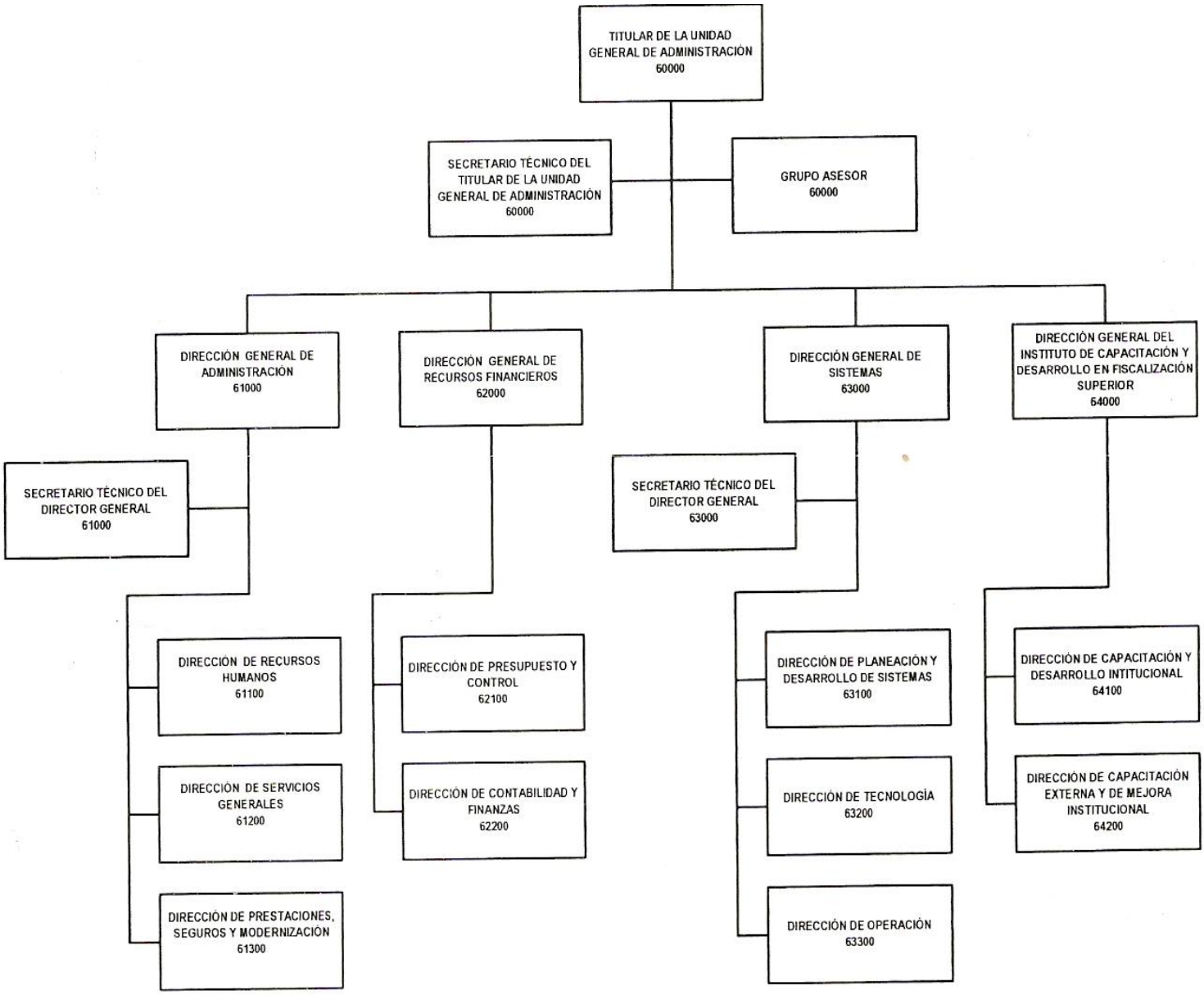


Figura 1.1 Organigrama de la ASF

II.-DESARROLLO DE ACTIVIDADES Y PROYECTOS EN LA AUDITORÌA SUPERIOR DE LA FEDERACIÒN

Las actividades en las cuales se integró mi participación, bajo la categoría de servicio social que desarrolle en la Auditoria Superior de la Federación(ASF) dentro de la Dirección General de Sistemas (DGS) se describen a continuación detalladamente.

2.1 MICROSOFT .NET

Mi labor dentro de la DGS era desarrollar aplicaciones Web con la tecnología de Microsoft .net.

La cual es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo nos brinda todas las herramientas y servicios que se necesitan para desarrollar modernas aplicaciones empresariales y de misión crítica, sino que también nos provee de mecanismos robustos, seguros y eficientes para asegurar que la ejecución de las mismas sea óptima. Los componentes principales de la plataforma .NET son:

- Un entorno de ejecución de aplicaciones, también llamado “Runtime”, que es un componente de software cuya función es la de ejecutar las aplicaciones .NET e interactuar con el sistema operativo ofreciendo sus servicios y recursos.
- Un conjunto de bibliotecas de funcionalidades y controles reutilizables, con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones.
- Un conjunto de lenguajes de programación de alto nivel, junto con su compilador, que permitirán el desarrollo de aplicaciones sobre la plataforma .NET.
- Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones
- Documentación y guías de arquitectura, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de aplicaciones .NET

Por otra parte, .NET representa la evolución COM (Component Object Model), la plataforma de desarrollo de Microsoft anterior a .NET y sobre la cual se basaba el desarrollo de aplicaciones Visual Basic 6 (entre otros tantos lenguajes y versiones).

Describiremos a continuación algunas de las características principales de la plataforma Microsoft .NET:

- Se dice que es una plataforma de ejecución intermedia, ya que las aplicaciones .NET no son ejecutadas directamente por el sistema operativo, como ocurre en el modelo tradicional de desarrollo. En su lugar, las aplicaciones .NET están diseñadas para ser ejecutadas contra un componente de software llamado Entorno de Ejecución (muchas veces también conocido como “Runtime”, o , “Máquina Virtual”). Este componente es el encargado de manejar el ciclo de vida de cualquier aplicación .NET, iniciándola, deteniéndola, interactuando con el Sistema Operativo y proveyéndole servicios y recursos en tiempo de ejecución.

- La plataforma Microsoft .NET está completamente basada en el paradigma de Orientación a Objetos (para más información acerca de este tema puede consultar el material de estudio de la Estrella 0 del programa)
- .NET es multi-lenguaje: esto quiere decir que para poder codificar aplicaciones sobre esta plataforma no necesitamos aprender un único lenguaje específico de programación de alto nivel, sino que se puede elegir de una amplia lista de opciones.
- .NET es una plataforma que permite el desarrollo de aplicaciones empresariales de misión crítica, entendiéndose por esto que permite la creación y ejecución de aplicaciones de porte corporativo que sean críticas para la operación de tipos variados de organizaciones. Si bien también es muy atrayente para desarrolladores no profesionales, estudiantes y entusiastas, su verdadero poder radica en su capacidad para soportar las aplicaciones más grandes y complejas.

- .Net fue diseñado de manera tal de poder proveer un único modelo de programación, uniforme y consistente, para todo tipo de aplicaciones (ya sean de formularios Windows, de consola, aplicaciones Web, aplicaciones móviles, etc.) y para cualquier dispositivo de hardware (PC's, Pocket PC's, Teléfonos Celulares Inteligentes, también llamados "SmartPhones", Tablet PC's, etc.). Esto representa un gran cambio con respecto a las plataformas anteriores a .NET, las cuales tenían modelos de programación, bibliotecas, lenguajes y herramientas distintas según el tipo de aplicación y el dispositivo de hardware.
- Uno de los objetivos de diseño de .NET fue que tenga la posibilidad de interactuar e integrarse fácilmente con aplicaciones desarrolladas en plataformas anteriores, particularmente en COM, ya que aún hoy existen una gran cantidad de aplicaciones desarrolladas sobre esa base.
- .NET no sólo se integra fácilmente con aplicaciones desarrolladas en otras plataformas Microsoft, sino también con aquellas desarrolladas en otras plataformas de software, sistemas operativos o lenguajes de programación. Para esto hace un uso extensivo de numerosos estándares globales que son de uso extensivo en la industria. Algunos ejemplos de estos estándares son XML, HTTP, SOAP, WSDL y UDDI.

2.2 .NET FRAMEWORK

El .NET Framework traducido como "Marco de Trabajo"es el componente fundamental de la plataforma Microsoft .NET, necesario tanto para poder desarrollar aplicaciones como para poder ejecutarlas luego en entornos de prueba o producción.
El .NET framework tiene tres variantes principales, todas descargables gratuitamente desde Internet
.NET Framework Redistributable Package: este es el mínimo componente de la plataforma .NET que se necesita para poder ejecutar aplicaciones. Normalmente ésta es la variante que se instala en los entornos productivos, una vez que el desarrollo y las pruebas de la aplicación han finalizado.
Está compuesto por:

- El entorno de ejecución de la plataforma .NET
- Las bibliotecas de funcionalidad reutilizable
- .NET Framework SDK: esta versión contiene herramientas de desarrollo de línea de comandos (compiladores, depuradores, etc.), documentación de referencia, ejemplos y manuales para desarrolladores de aplicaciones. Normalmente ésta variante se instala en los entornos de desarrollo de aplicaciones, y es más útil a los programadores que a los usuarios finales.

Para poder instalar la versión SDK (Software Development Kit) es necesario instalar previamente el Redistributable Package.

- .NET Compact Framework: esta es una versión reducida del .NET Framework Redistributable, especialmente pensada para ser instalada en dispositivos móviles como Pocket PC's y SmartPhones.

El .NET Framework puede ser instalado en cualquier sistema operativo de la familia Windows superior a Windows 98.
Actualmente, Windows 2003 Server y Windows XP SP2 traen el .NET Framework preinstalado.

- .NET no es un Lenguaje de Programación: si bien la plataforma Microsoft .NET incluye lenguajes de programación de aplicaciones, su concepto es más amplio y va más allá de éstos.
- .NET no es un Entorno de Desarrollo: si bien la plataforma Microsoft .NET incluye entornos de desarrollo integrados (IDEs), su concepto es más amplio y va más allá de éstos.
- .NET no es un servidor de aplicaciones (Application Server)
- .NET no es un producto empaquetado que se pueda comprar como tal, sino que es una plataforma que engloba distintas aplicaciones, servicios y conceptos y que en conjunto permiten el desarrollo y la ejecución de aplicaciones.

El .NET Framework debe estar instalado en cualquier dispositivo de hardware para que la ejecución de una aplicación .NET sea posible. En el caso de las aplicaciones de escritorio (también llamadas "De Formularios Windows") y las aplicaciones de consola (aplicaciones cuya interfaz de usuario es una consola de comandos), el Framework debe estar presente del lado del cliente (computadora donde se ejecuta la parte de la aplicación que interactúa con el usuario), y en el servidor sólo en caso de que la aplicación sea distribuída y tenga parte de su funcionalidad centralizada en una única computadora Figura 2.1.

En el caso de las aplicaciones Web, el único requisito del lado del cliente es tener un navegador y una conexión de red al servidor, el cual debe tener instalado el .NET Framework

Para las aplicaciones móviles, que se ejecutan sobre Windows Mobile en algún dispositivo tipo Pocket PC o SmartPhone, es necesario tener instalado el .NET Compact Framework en el dispositivo.

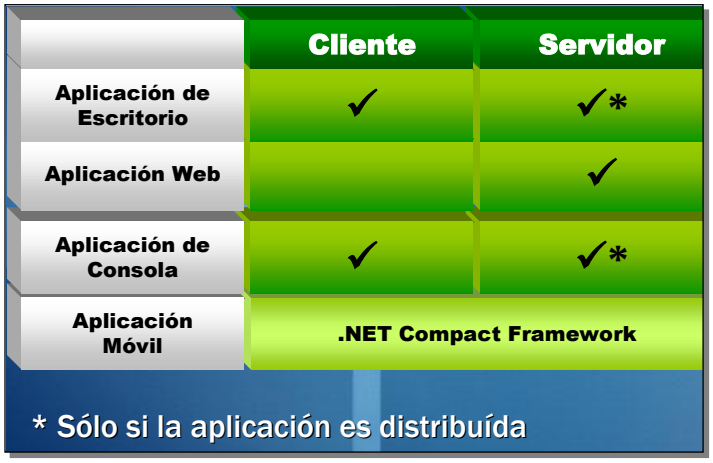


Figura 2.1 Donde instalar el .net Framework

.NET unifica todos esos modelos de programación ofreciendo una única API, un único entorno de ejecución, un único conjunto de bibliotecas y una única herramienta de desarrollo para cualquier tipo de aplicación.

En la figura 2.2 podemos ver que la plataforma Microsoft.NET es denominada "de Ejecución Intermedia" justamente porque se ubica entre el Sistema Operativo y las aplicaciones finales con las que interactúan los usuarios, actuando como intermediario entre ambos.

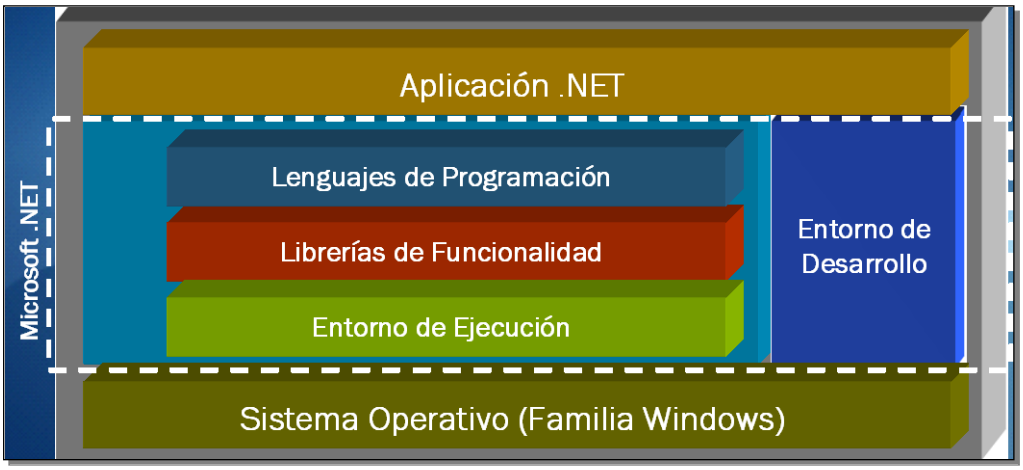


Figura 2.2 Plataforma .net

En la figura 2.3 se pueden apreciar las distintas partes que componen al .NET Framework, incluídas el entorno de ejecución de aplicaciones (CLR, en verde), el conjunto de bibliotecas de funcionalidad reutilizable (.NET Framework Class Library, en azul) y los compiladores y herramientas de desarrollo para los lenguajes .NET (en naranja). Todos estos componentes se motan por encima de la familia de sistemas operativos Windows.

Dentro del conjunto de la .NET Framework Class Library se distinguen 4 sub-componentes principales:
La Base Class Library (BCL - Biblioteca de Clases Base), que contiene la funcionalidad más comúnmente utilizada para el desarrollo de todo tipo de aplicaciones. Algunos ejemplos de la funcionalidad provista por la BCL son el manejo de colecciones, cadenas de texto, entrada/salida, threading, operaciones matemáticas y dibujos 2D.
ADO.NET, que contiene un conjunto de clases que permiten interactuar con bases de datos relacionales y documentos XML como repositorios de información persistente.
ASP.NET, que constituye la tecnología dentro del .NET Framework para construir aplicaciones con interfaz de usuario Web (es decir, aplicaciones cuya lógica se encuentra centralizada en uno o varios servidores y que los clientes pueden acceder usando un browser o navegador mediante una serie de protocolos y estándares como HTTP y HTML).
Windows Forms (o simplemente WinForms), que constituye la tecnología dentro del .NET Framewok que permite crear aplicaciones con interfaz de usuario basada en formularios y ventanas Windows de funcionalidad rica y que se ejecutan directamente en los clientes.

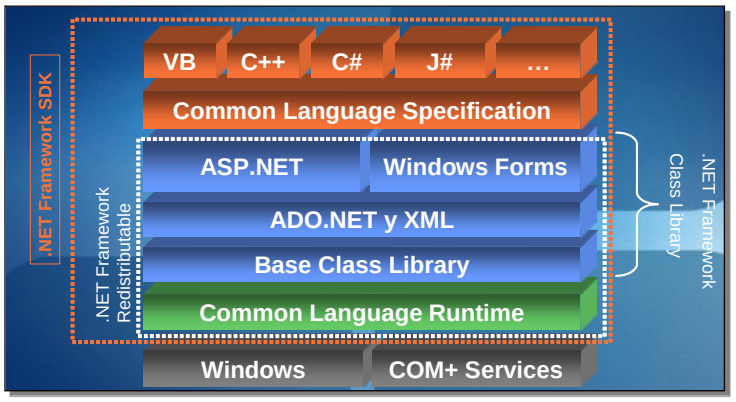


Figura 2.3 Arquitectura de .net

El modelo de ejecución que propone la plataforma .NET se suele definir como "virtual", o "de máquina virtual", ya que las aplicaciones no son desarrolladas directamente contra las APIs de programación expuestas por el sistema operativo, ni es éste el que se encarga de su ejecución y ciclo de vida, sino que .NET provee un entorno de ejecución (el CLR) que corre por sobre el sistema operativo y que es el encargado de ejecutar las aplicaciones y proveerles servicios en tiempo de ejecución. A los componentes de software que se ejecutan de esta manera se los conoce comúnmente como "componentes manejados", ya que su ejecución es controlada por un entorno intermedio. En la figura podemos ver las diferencias entre las arquitecturas de ejecución de los componentes tradicionales (como los COM) y los componentes manejados.
Una de las principales ventajas de contar con una plataforma virtual es que no están "atadas" de ninguna forma con el sistema operativo y la plataforma de hardware subyacente. Es sabido que una aplicación compilada para que utilice directamente las APIs y servicios expuestas por un sistema operativo "x" muy difícilmente pueda ser ejecutada en otro sistema operativo distinto sin ser recompilada. Las aplicaciones manejadas, en cambio, descansan la tarea de su compilación a un código de máquina específico en el entorno de ejecución. De esta manera, si existen distintos entornos de ejecución intermedia para diferentes Sistemas Operativos, la misma aplicación puede ejecutarse en todos ellos si necesidad de recompilarse.

2.3 VISUAL STUDIO 2005

Para el desarrollo de todos los proyectos dentro de la ASF utilice Visual Studio 2005 Team Edition que es edición diseñada para desarrolladores profesionales y cuenta con todas las herramientas de Microosoft.net.

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

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

Visual Studio es la herramienta de desarrollo por excelencia de la plataforma .NET, siendo una herramienta única que permite desarrollar cualquier tipo de aplicación (Web, Windows, de Consola, para dispositivos Móviles, para Microsoft Office, de Bases de Datos y más) en cualquiera de los lenguajes .NET provistos por Microsoft (C#, VB.NET, C++.NET y J#).

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System. Éstas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los 4 principales roles de la programación: Architects, Software Developers, Testers, y Database Professionals. La funcionalidad combinada de las 4 ediciones Team System se ofrecía como la edición Team Suite Figura 2.4.

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.

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

Figura 2.4 Ediciones de Visual Studio 2005.

2.4 APLICACIONES WEB

Después de varios días de introducción se me encargo la realización del primer proyecto de una aplicación web llamada CheckUp el cual serviría para hacer reservaciones a revisiones médicas para todos los empleados de la ASF, esta reservación la realizaría cada usuarios a través de su computadora la cual estaba conectada a al intranet institucional, si el usuario no tenia acceso a una podía llamar a la DGS para que se realizara su reservación. Esta aplicación Web fue desarrollada con asp.net.

Las aplicaciones web son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, etc.) en la que se confía la ejecución al navegador.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

Es importante mencionar que una página Web puede contener elementos que permiten una comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como por ejemplo rellenar y enviar formularios, participar en juegos diversos y acceder a gestores de base de datos de todo tipo.

Una ventaja significativa es que las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación web se escribe una vez y se ejecuta igual en todas partes.

Aunque existen muchas variaciones posibles, una aplicación web está normalmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web ofrece la primera capa y un motor capaz de usar alguna tecnología web dinámica (ejemplo: PHP, Java Servlets o ASP, ASP.NET, CGI, ColdFusion, embPerl, Python (programming language) o Ruby on Rails) constituye la capa de enmedio. Por último, una base de datos constituye la tercera y última capa.

El navegador web manda peticiones a la capa de en medio que ofrece servicios valiéndose de consultas y actualizaciones a la base de datos y a su vez proporciona una interfaz de usuario.

Existen numerosos lenguajes de programación empleados para el desarrollo de Aplicaciones Web, entre los que destacan:

-PHP
-ASP/ASP.NET
-Java, con sus tecnologías Java Servlets y JavaServer Pages (JSP)
-Python
-HTML
-XML

ASP.NET

ASP.NET es el framework de programación web dentro de .NET
Permite desarrollar aplicaciones Web con un modelo “similar” al utilizado para aplicaciones Windows
El componente fundamental de ASP.NET es el WebForm
Independencia del cliente (navegador, S.O., dispositivo físico, etc.)
Permite utilizar cualquier lenguaje .NET
Permite desarrollar Servicios Web XML

La “parte ejecutable” de una aplicación ASP.NET es COMPILADA
Implementación y actualización de las aplicaciones sin reiniciar el servidor!
Acceso a toda la .NET Class Library
Independiente del lenguaje de programación
Encapsulamiento de funcionalidad a través de controles de servidor y controles de usuario

Permite usar ADO.NET para acceso a datos
Soporta XML, Hojas de estilo CSS, etc.
Detección automática del navegador cliente, generando el lenguaje de marcas soportado por el mismo
Mecanismo de Caching incorporado para páginas completa o partes de la misma frecuentemente solicitadas

WebForms (Formularios Web)

Uno o más archivos con extensión .aspx
Archivos Code-Behind
Archivos asociados a WebForms que contienen código del lado del servidor (Ej. VB.NET, C#, etc.)
Archivos de configuración con formato XML
Un archivo Web.config por c/aplicación
Un único archivo Machine.config por servidor
Global.asax
Eventos a nivel de aplicación

Directorio BIN
Contiene el assembly de la aplicación (Ej.: MiAplic.dll)
Cero o más assemblies (Componentes externos)
Enlaces a Servicios Web XML
Permiten a la aplicación ASP.NET enviar y recibir datos desde Servicios Web

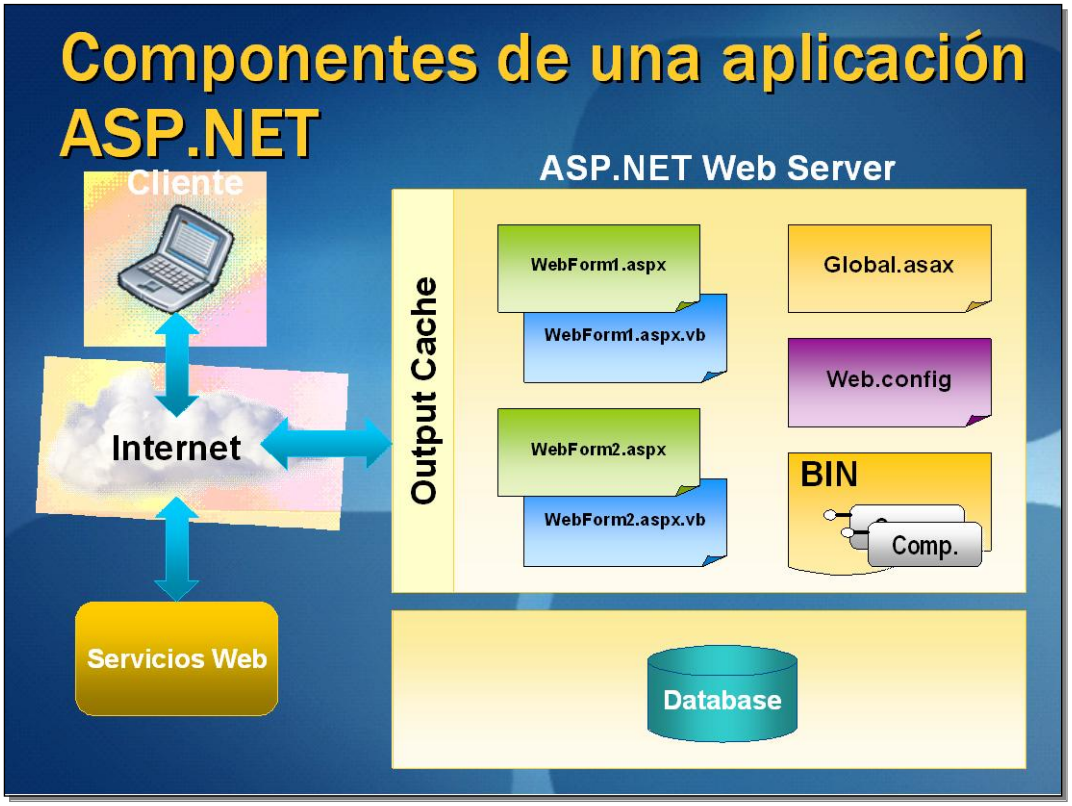


Figura 2.5 Componentes de una aplicación asp.net

Formulario Web (ASP.NET web form)
Es una página expresada en lenguaje de marcas que es compilada y ejecutada dinámicamente en el servidor para generar la salida solicitada por el cliente (explorador ó dispositivo).

Code Behind
Es el código que se ejecuta del lado del servidor para lograr el comportamiento deseado en un formulario web.

Partial Class
Un nuevo concepto, que es aplicado en ASP.NET para vincular las páginas aspx (la interfaz del usuario) con su Code Behind (comportamiento).

Los Formularios Web son archivos de texto que poseen la extensión .aspx que, generalmente, expresan la interfaz del usuario. Estos archivos son interpretados por ASP.NET vinculando la interfaz del usuario con el código del lado del servidor (code behind). Para lograr la vinculación, la directiva Page entra en juego:
<%@Page compilewith="MiPagina.aspx.cs"classname="MiEspacioDeNombres.MiClase" %>

Code Behind (código detrás): logra el comportamiento deseado de un formulario web.
Definir clases como Partial Class permite que una misma clase esté definida de forma parcial en múltiples archivos.

El archivo de configuración web.config permite una fácil administración e instalación de una aplicación web ASP.NET conteniendo toda la información relevante de la aplicación (autenticación, sesiones, autorización, etc.) incluyendo valores referentes a la funcionalidad propia de la aplicación.
Al modificar el archivo de configuración web.config, no es necesario reiniciar la aplicación en el servidor, ésta lo detecta automáticamente.
El sistema de configuración de ASP.NET se compone de dos archivos:

- Machine.config: donde se guarda información de configuración para todo el servidor. (Existe un solo archivo machine.config instalado por cada versión de ASP.NET)
- Web.config: donde se guarda información de configuración específica de la aplicación, se pueden colocar múltiples archivos Web.Config en una aplicación, por ejemplo uno en la raíz y uno en cada uno de los subdirectorios que lo necesiten. Pero no puede haber más de uno por directorio.
-

Otra ventaja es que se pueden agregar archivos de configuración externos, referenciados desde el web.config, facilitando enormemente el trabajo para grandes equipos de desarrollo.

2.5.1 Visual Basic .NET

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET

2.5.2 C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma.

C# contiene dos categorías generales de tipos de datos integrados: tipos de valor y tipos de referencia. El término tipo de valor indica que esos tipos contienen directamente sus valores.

2.5.3 Tipos de datos

Los tipos de punto flotante pueden representar números con componentes fraccionales. Existen dos clases de tipos de punto flotante; float y double. El tipo double es el más utilizado porque muchas funciones matemáticas de la biblioteca de clases de C# usan valores double. Quizá, el tipo flotante más interesante de C# es decimal, dirigido al uso de cálculos monetarios. La aritmética de punto flotante normal está sujeta a una variedad de errores de redondeo cuando se aplica a valores decimales. El tipo decimal elimina estos errores y puede representar hasta 28 lugares decimales.

Los caracteres en C# no son cantidades de 8 bits como en otros muchos lenguajes de programación. Por el contrario, C# usa un tipo de caracteres de 16 bits llamado Unicode al cual se le llama char. No existen conversiones automáticas de tipo entero a char.

Categoría	Clase	Descripción	C# Alias	VB.NET Alias
Enteros	Byte	Un entero sin signo (8-bit)	byte	Byte
	SByte	Un entero con signo (8-bit)	sbyte	Sbyte
	Int16	Un entero con signo (16-bit)	short	Short
	Int32	Un entero con signo (32-bit)	int	Integer
	Int64	Un entero con signo (64-bit)	long	Long
Punto Flotante	Single	Un número de punto flotante de simple precisión (32-bit)	float	Single
	Double	Un número de punto flotante de doble precisión (64-bit)	double	Double
	Decimal	Un número decimal de 96-bit	decimal	Decimal
Lógicos	Boolean	Un valor booleano (true o false)	bool	Boolean
Otros	Char	Un caracter Unicode (16-bit)	char	Char
	Object	La raíz de la jerarquía de objetos	object	Object
	String	Una cadena de caracteres unicode inmutable y de tamaño fijo	string	String

Figura 2.6 Tipos de datos

2.5.4 Modificadores de acceso

Todos los tipos y miembros de tipo tienen un nivel de accesibilidad, que controla si pueden utilizarse por otro código de su ensamblado u otros ensamblados. La accesibilidad de un tipo o miembro se especifica al declararlo utilizando uno de estos modificadores de acceso:

C#: todo miembro es declarado como PRIVATE por default
VB.NET: todo miembro es declarado como PUBLIC por default
public
Puede obtener acceso al tipo o miembro cualquier otro código del mismo ensamblado o de otro ensamblado que haga referencia a éste.
private
Solamente puede obtener acceso al tipo o miembro código de la misma clase o estructura.
protected
Solamente puede obtener acceso al tipo o miembro código de la misma clase o estructura o de una clase derivada.
internal
Puede obtener acceso al tipo o miembro cualquier código del mismo ensamblado, pero no de un ensamblado distinto.

C#	VB.NET
public	Public
private	Private
internal	Friend
protected	Protected
protected internal	Protected Friend

Figura 2.7 Modificadores de acceso

2.5.5 Declaración de Variables

C#: el tipo de dato precede al identificador (nombre)

```
int x;
decimal y;
rectangle z;
Cliente cli;
```

Figura 2.9 Variables en c#

VB.NET: comienza con "Dim" o algún modificador de acceso (Public, Private, etc.) + identificador de la variable + "As" Tipo de Dato

```
Dim x As Integer        'Dim es = a Private por
defecto
Dim y As Decimal
Dim z As Rectangle
Dim cli As Cliente
```

Figura 2.10Variables en visual Basic.net

2.5.6 Operadores

Descripción	C#	VB.NET
Asignación	=	=
Adición	+	+
Sustracción	-	-
Multiplicación	*	*
División	/	/
Negación	!	not
Módulo (Parte entera de la división)	%	mod
Mayor	>	>
Menor	<	<
Mayor o Igual	>=	>=
Menor o Igual	<=	<=

C#	VB.NET	Operador
&&	And	Operador logico Y
||	Or	Operador logico O
!	Not	Negacion logica
==	=	Igual
!=	<>	Distinto

Figura 2.11 Operadores

2.5.7Namespaces

Los namespaces (espacios de nombre) funcionan como un método de organización de clases (tanto las que escribimos nosotros como las incluidas en .NET), agrupando en un conjunto a aquellas clases que están relacionadas lógicamente. Un namespace puede contener tanto tipos (clases) como otros namespaces, y dentro de un namespace no puede haber dos clases con el mismo nombre. El nombre completo de una clase se construye a partir de todos los namespaces que la contienen (la jerarquía se arma separándolos con ".") + el nombre mismo de la clase.

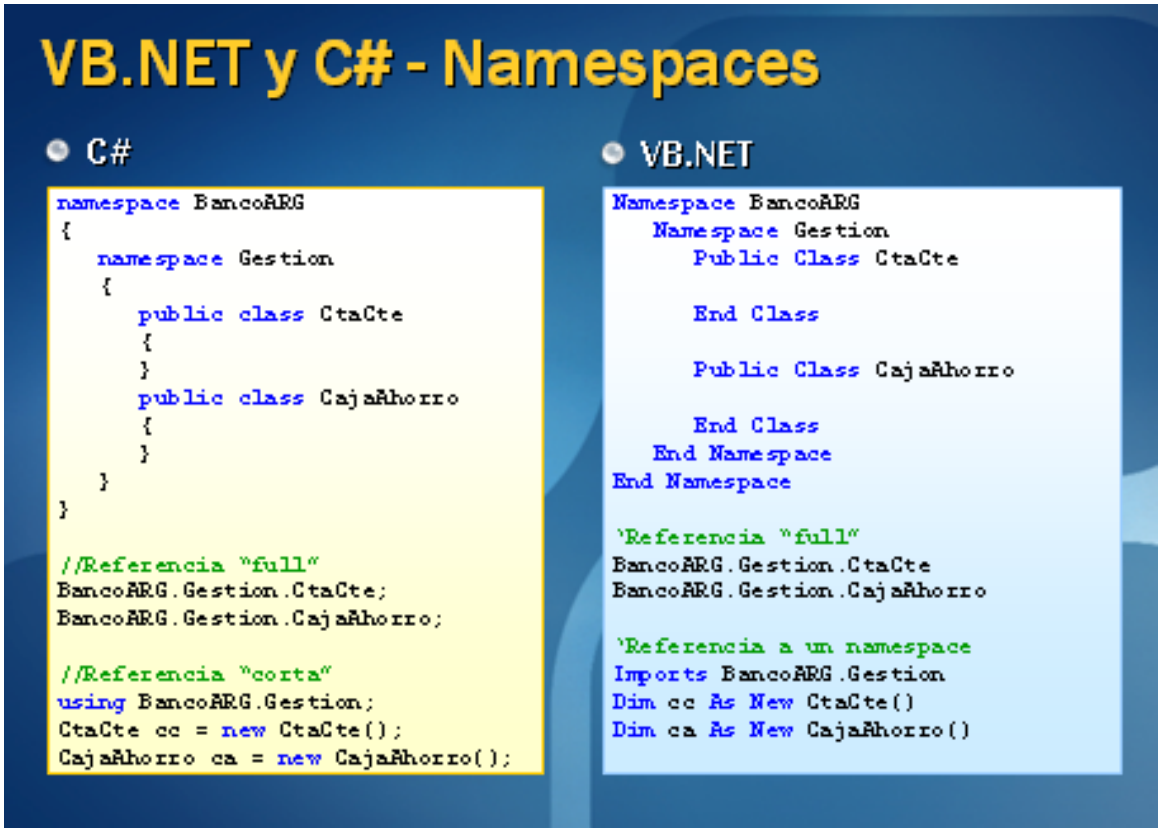


Figura 2.12 Namespaces

2.5.8 Administrador de Excepciones

.NET provee una forma estructurada de manejar los errores que ocurren en tiempo de ejecución (llamados excepciones), a través de los bloques Try/Catch.

- Todo código que sea susceptible a fallar en tiempo de ejecución debe ir dentro de un bloque "Try"
- Debemos proveer al menos un bloque "Catch" que "atrapará" una excepción en caso de producirse y hará algo con ella (mostrará un mensaje, enviará un mail, registrará el error, etc.)
- Si en tiempo de ejecución se produce una excepción dentro del código que se encuentra en el bloque Try, el flujo de control es automáticamente direccionado al primer bloque Catch capaz de atraparla, y nunca vuelve al código dentro del bloque Try.

- Independientemente de la ocurrencia o no de excepciones, la ejecución termina dirigiéndose al bloque Finally (optativo), que es utilizado típicamente para finalizar ordenadamente recursos que se hayan estado utilizando (archivos, conexiones a una base de datos, etc.).

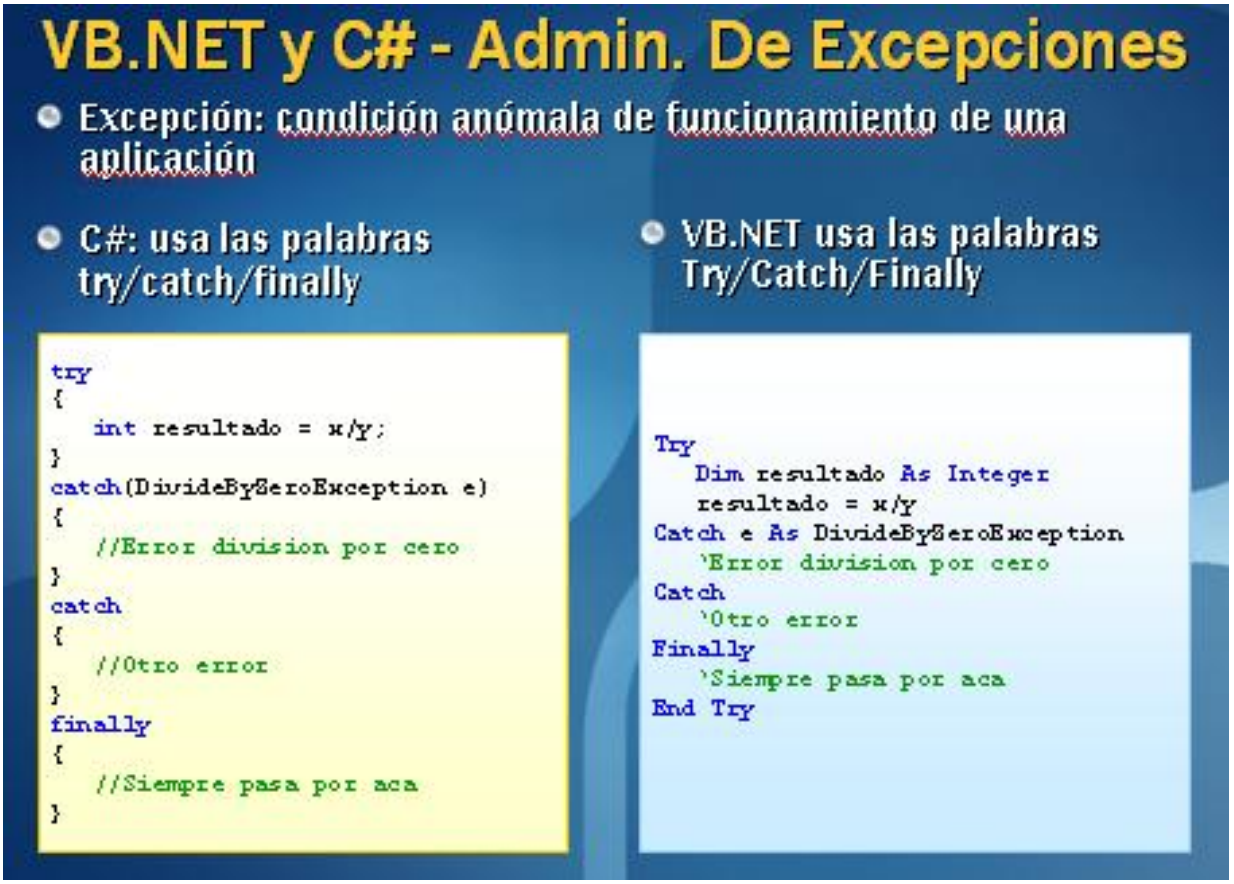


Figura 2.13 Excepciones

2.5.9 Master Pages

ASP.NET provee un nuevo mecanismo para la creación de aplicaciones web que permite, mediante la declaración de una página maestra (master page), que todo el sitio posea la misma apariencia.

En versiones anteriores, si había que utilizar un menú, lo mas usual era crear un control de usuario con la funcionalidad del menú y luego incluirlo en todas las páginas.
Ahora, el menú solo debe incluirse en la página maestra, y todas las páginas de contenido que estén vinculadas con ella tendrán el menú.
Esto permite un código más limpio y ordenado en la aplicación, haciendo al mismo más fácil de mantener.

La página maestra se puede configurar para toda la aplicación desde el web.config, o bien en la directiva @Page de las páginas que la utilicen.

Las páginas de contenido son páginas aspx que tienen una referencia a alguna página maestra (MasterPageFile).

Una página maestra permite añadir código, controles, estilos, etc. que deben ser compartidos por todas o la mayoría de las páginas de la aplicación.

Las páginas que están vinculadas con la página maestra heredando su contenido visualmente, se llaman páginas de contenido; estas son independientes de su página maestra, tanto en el código como en el lenguaje que se desee programar.
Es decir, puedo tener una master page codificada en C#, y las páginas de contenido en VB .Net.

Otro tema importante es que se pueden crear sitios con páginas maestras anidadas (Nested Master Pages). Es decir, un sitio puede tener distintos módulos, con una una página maestra que contiene un encabezado determinado para toda la aplicación, y luego crear otras páginas maestras para cada modulo, vinculadas con la ya definida, que posean el menú de cada modulo.

2.5.10 Autenticación

Para saber que usuario era el que se entraba a la aplicación Web se empleo un sistema de Autenticación de asp.net el cual se describe a continuación.

¿Qué es Autenticación?

Es el mecanismo que permite afirmar que la persona que esta ingresando al sistema es quien dice ser.

¿Cómo Funciona?

Se aceptan las credenciales ingresadas por el usuario (usuario – contraseña) y se validan contra una base de datos, el sistema operativo, un servicio web, u otro mecanismo definido según el tipo de autenticación.

Existen 3 tipos de autenticación en aplicaciones .Net.
La autenticación basada en windows, utiliza la infraestructura de Windows para validar las credenciales, es apropiada para aplicaciones intranet y esta es la que se utilizo para desarrollar CheckUp.

La autenticación basada en formularios, utiliza generalmente bases de datos para la validación de credenciales, es apropiada para aplicaciones web.

La autenticación basada en MS Passport, utiliza un servicio web para la validación de credenciales, por lo cual no necesita guardar los datos de los usuarios en una base de datos, pero su utilización tiene un costo económico.

Este modo de autenticación es muy utilizado en sitios web públicos.
Cuando se confirma que las credenciales de autenticación (usuario y contraseña) son válidas, se genera un ticket de autenticación con la identidad del usuario. Luego en las siguientes peticiones de página, se autoriza al usuario utilizando este "ticket".

Este ticket de autenticación se guarda en una cookie. Las cookies son archivos de texto que guardan información de estado en la PC cliente.

Las cookies son una parte muy importante en todas las aplicaciones Web.

En ASP.NET las cookies se manejan con HttpCookie, que está dentro del namespace System.Web.

Si el usuario es anónimo, redirecciona las peticiones a una página predeterminada para validar las credenciales del usuario

Figura 2.14 Autenticación

Como se mencionó anteriormente, las páginas HTML se transmiten por medio del protocolo HTTP de un servidor al cliente y viceversa.
Este protocolo es un protocolo "sin estado", por consiguiente, la información de las páginas web se pierde entre los idas y vueltas al servidor.

Como en muchos casos es necesario conservar estos datos, ASP .Net provee distintos mecanismos que permiten mantener el estado de sus páginas a través de los idas y vueltas.

Entre los mecanismos, se encuentra el "ViewState", "Application State", "Session State", es decir los datos se mantienen a nivel de aplicación y a nivel de sesión del usuario.
ASP.NET provee un mecanismo para mantener información individual de cada interacción de un navegador con el servidor web llamado sesión.
La sesión perdura un tiempo determinado, previamente establecido en el archivo de configuración; y se renueva por cada pedido del cliente.
La colección Session permite almacenar la información particular de una sesión.
Cuando un usuario solicita por primera vez una página .aspx al servidor web, ASP.NET crea una nueva sesión para él. En este momento es factible la carga en memoria de la información del usuario.
Por defecto la información de sesión se guarda en memoria.

Para mantener el estado de los controles de una página aspx entre postbacks, ASP.NET utiliza el View State.

Esta técnica es utilizada por defecto en todos los controles, pero puede deshabilitarse en caso de no ser necesario, mediante la propiedad *EnableViewState*.

El View State se puede declarar por página (afectando a todos los controles de la misma) o por cada control.

El tamaño del campo oculto donde se almacena puede crecer innecesariamente dificultando la navegación del sitio; es por eso necesario moderar el uso del ViewState.
En el servidor se implementa como una colección: ViewState.

HTTP define como los navegadores y los servidores Web se comunican uno con otro.
En una aplicación web, el componente principal es el HTML Form, que es el elemento de html que contiene los controles de la interfaz de usuario de cada página y captura la entrada de datos del usuario. El HTML Form es la porción de la página que es enviada a través del protocolo HTTP al servidor para procesar el pedido realizado por el usuario.
Hay dos modos de envío de formularios al servidor: **Get** y **Post**.
El primero envía los datos ingresados en el formulario como una cadena de consulta, el segundo envía los datos en el cuerpo del pedido.

2.5.11 Sentencias condicionales

C#: sentencia if con varios formatos

```
if (x > 10)
   HacerAlgo();

if (x < 10)
{
    Hacer1();
    Hacer2();
}

if (x < 10)
{
    Hacer1();
}
else
{
    Hacer2();
}

if (x < 10)
{
    Hacer1();
}
else if (x > 20)
{
    Hacer2();
}
else
{
    Hacer3();
}
```

Figura 2.15 Sentencia if en c#

VB.NET: la sentencia If requiere de la palabra Then

```
If x > 10 Then Hacer()

If x < 10 Then
    Hacer1()
    Hacer2()
End If

If x < 10 Then
    Hacer1()
Else
    Hacer2()
End If

If x < 10 Then
    Hacer1()
ElseIf x > 20 Then
    Hacer2()
Else
    Hacer3()
End If
```

Figura 2.16 Sentencia if en visual Basic.net

C#: sentencia case

```
int a = 0;
switch(a) {
        case 1:{//CODIGO 1
          break;
        }
        case 2: {//CODIGO 2
          break;
        }
        default:{//CODIGO DEFAULT
          break;
        }
}
```

Figura 2.17 Sentencia switch en c#

VB.NET: sentencia case

```
Dim a As Integer = 0
Select a
        Case 1
          'Código 1
        Case 2
          'Código 2
        Case Else
          'Código Default
End Select
```

Figura 2.18 Sentencia switch en visual Basic.net

2.6 ADO.NET

Para las conexiones a las bases de datos utilice ADO.NET

ADO.NET es un subconjunto de la .NET Framework Class Library, que contiene todas las funcionalidades necesarias para conectarse e interactuar con dos tipos de repositorios permamentes de información:

1) Bases de Datos, como Microsoft SQL Server (clases del namespace System.Data, que se encuentran compiladas en System.data.dll)
2) Archivos XML (clases del namespace System.XML, que se encuentran compiladas en System.Xml.dll)

En la actualidad se plantean dos tipos de escenarios de acceso a bases de datos relacionales. El primero de ellos es el que se conoce como “Escenario Conectado”, ya que en él se requiere una conexión física establecida con el servidor de datos durante todo momento para poder efectuar cualquier consulta o actualización sobre los datos.
Esto tiene algunas ventajas y también sus desventajas.
Algunas Ventajas:

- Al haber una única conexión a la base de datos por usuario, o incluso a veces por aplicación, establecida permanentemente, puede llegar a resultar más sencillo administrar la seguridad y el acceso al servidor de datos. Lo mismo ocurre con el control de concurrencia: en un escenario donde múltiples usuarios se estuvieran conectando y desconectando permanentemente para realizar distintas acciones, este control sería más difícil de llevar.
- Siempre la aplicación tiene acceso a los datos actualizados

Algunas Desventajas:

- Se requiere una conexión abierta todo el tiempo con el servidor de base de datos, lo cual consume recursos innecesariamente si no se la está utilizando.
- La escalabilidad del acceso a los datos se ve limitada por la cantidad de conexiones establecidas simultáneamente contra el servidor de base de datos.

El segundo escenario de acceso a bases de datos relacionales se conoce como “Escenario Desconectado”, ya que en él una parte de los datos del servidor central se copia localmente y puede luego ser consultada y actualizada sin contar con una conexión abierta. Luego si se desea puede

establecerse una conexión con el servidor de base de datos para sincronizar los cambios efectuados sobre la copia local y actualizar los datos. Este tipo de funcionalidad es particularmente útil para escenarios de usuarios móviles, que salen de su oficina con una laptop, un SmartPhone o una PocketPC y desean poder continuar trabajando por más que no tengan conectividad física con el servidor de base de datos ubicado en la red interna de la empresa.
Algunas ventajas que provee un escenario de acceso a datos desconectado son:

- La posibilidad de trabajar sobre los datos independientemente del resto de los usuarios de la aplicación
- Mayor escalabilidad en el acceso a datos y utlización más óptima de recursos del servidor, ya que se mantiene en un mínimo indispensable la cantidad y duración de conexiones abiertas.
- Mayor performance, al trabajar con una copia local de los datos.

Algunas Desventajas:

- Puede ocurrir que en un momento dado un usuario no esté accediendo a los datos más actualizados del repositorio central
- Al momento de sincronizar los cambios efectuados localmente contra el repositorio central pueden surgir conflictos, los cuales deben ser resueltos manualmente.

ADO.NET soporta el acceso a datos tanto en escenarios conectados como desconectados.

La arquitectura de ADO.NET está basada en el concepto de proveedores de acceso a datos, siendo un proveedor un conjunto de clases que permiten conectarse a una base de datos, ejecutar un comando sobre ella y tener acceso a los resultados de su ejecución, tanto de forma conectada como desconectada

Los proveedores de acceso a datos ADO.NET (conocidos como “Managed Data Providers”) representan conjuntos específicos de clases que permiten conectarse e interactuar con una base de datos, cada uno utilizando un protocolo particular. El .NET Framework incluye cuatro proveedores de acceso a datos, que en conjunto le permiten conectarse e interactuar virtualmente con cualquier base de datos existente en la actualidad:

- Data Provider For SQL Server: es el proveedor de acceso nativo a servidores de bases de datos Microsoft SQL Server 7.0 o superior, y Microsoft Access. Al conectarse via protocolos nativos de bajo nivel, povee la alternativa más performante para conexiones contra estos motores de bases de datos. Sus clases se encuentran en el namespace System.Data.SqlClient.
- Data Provider For OLE DB: es el proveedor de acceso a datos que permite interactuar via el protocolo estándar OLE DB con cualquier repositorio de datos que lo soporte. Sus clases se encuentran en el namespace System.Data.OleDb.
- Data Provider For ODBC: es el proveedor de acceso a datos que permite interactuar via el protocolo estándar ODBC con cualquier repositorio de datos que lo soporte. Sus clases se encuentran en el namespace System.Data.Odbc.
- Data Porvider For Oracle: es el proveedor de acceso nativo a bases de datos Oracle, desarrollado por Microsoft utilizando las herramientas de conectividad de Oracle. De esta forma puede lograrse un acceso más performante a bases de datos Oracle desde aplicaciones .NET que utilizando ODBC u OLE DB. Sus clases se encuentran en el

namespace System.Data.OracleClient, y están compiladas en un assembly diferente al resto: System.Data.OracleClient.dll.

ADO.NET provee una arquitectura extensible, posibilitando que terceras partes creen sus propios proveedores de acceso nativo para aplicaciones .NET. Algunos ejemplos de esto son:

- Data Provider For DB2, desarrollado por IBM
- Oracle Data Provider For .NET, desarrollado por Oracle
- Providers de acceso nativo a bases de datos OpenSource, como MySQL y PostgreSQL

Es importante volver a destacar que utilizando estos proveedores de acceso a datos cualquier aplicación .NET puede utilizar casi cualquier base de datos relacional existente en la actualidad como respositorio de información persistente (esto incluye, además de MS SQL Server, a IBM DB2, Oracle, Sybase, Informix, TeraData, MySQL y PostgreSQL, entre otras).

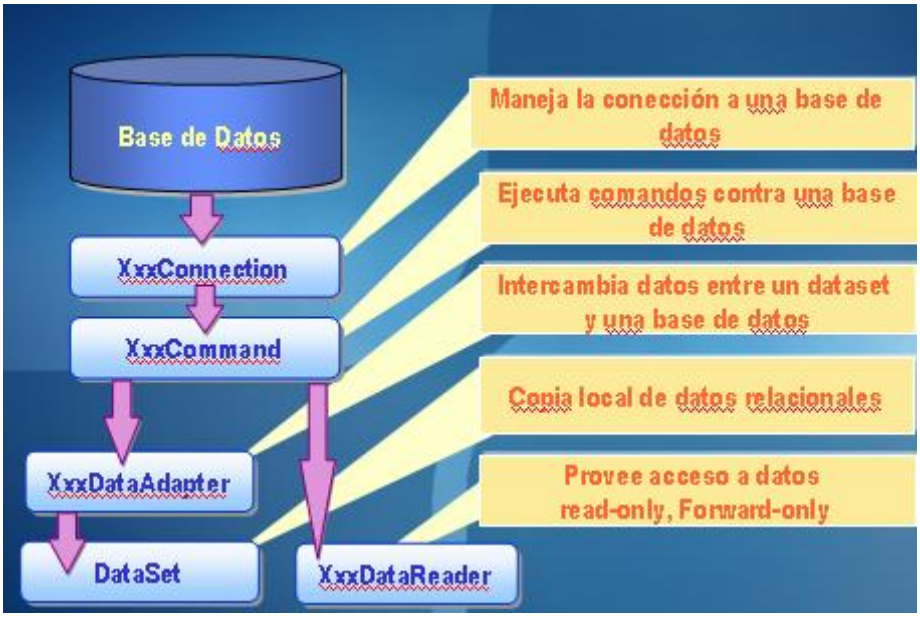


Figura 2.19 Clases de ADO.NET

En la figura 2.19 se pueden apreciar las clases más comunes que componen a todos los proveedores de acceso a datos de ADO.NET. Nótese que algunos nombres empiezan con las letras "Xxx": esto se debe a que los nombres de esas clases varían según el proveedor específico que se esté utilizando. Por ejemplo, la clase que representa una conexión con la base de datos usando el Data Provider For Sql Server es "SqlConnection", mientras que si usamos el Data Provider For Oracle podemos obtener la misma funcionalidad de la clase "OracleConnection". Mas allá del ejemplo, pasemos a describir cada una de estas clases y su funcionalidad:

- XxxConnection: representa una conexión. Almacena, entre otras cosas, el string de conexión (connection string), y permite conectarse y desconectarse con una base de datos.
- XxxCommand: permite almacenar y ejecutar una instrucción SQL contra una base de datos, enviando parámetros de entrada y recibiendo parámetros de salida.

Estas dos clases se utilizan tanto en escenarios conectados como desconectados.

- XxxDataReader: permite acceder a los resultados de la ejecución de un comando contra la base de datos de manera read-only (sólo lectura), forward-only (sólo hacia adelante). Esta clase se utiliza en escenarios conectados, ya que no es posible operar sobre los registros de un DataReader estando desconectado de la fuente de datos.

- XxxDataAdapter y DataSet: en conjunto, estas clases constituyen el corazón del soporte a escenarios desconectados de ADO.NET. El DataSet es una representación en memoria de una base de datos relacional, que permite almacenar un conjunto de datos obtenidos mediante un DataAdapter. El DataAdapter actúa como intermediario entre la base de datos y el DataSet local desconectado. Una vez que el DataSet se encuentra lleno con los datos que se necesitan para trabajar, la conexión con la base de datos puede cerrarse sin problemas y los datos pueden ser modificados localmente. Por último, el DataAdapter provee un mecanismo para sincronizar los cambios locales contra el servidor de base de datos. Nótese que la clase System.Data.DataSet no tiene el prefijo Xxx, ya que es independiente del proveedor de acceso a datos utilizado.

Como ya se ha mencionado, el DataSet es una representación residente en memoria de datos relacionales, independiente de la base de datos y del protocolo utilizado para interactuar con la misma. Un DataSet, al igual que una base de datos, está compuesto por un conjunto de tablas (colección de clases “DataTable”), cada una de las cuales está compuesta a su vez por un conjunto de filas (colección de clases “DataRow”) y columnas (colección de clases “DataColumn”). Dentro de un DataSet pueden establecerse relaciones entre DataTables, y hasta restricciones de integridad referencial (Claves Primarias y Foráneas). Internamente, los DataSets representan toda su estructura y datos contenidos en formato XML.

En un escenario conectado, los recursos se mantienen en el servidor hasta que la conexión se cierra

1) Abrir Conexión

2) Ejecutar Comando

3) Procesar Filas en DataReader

4) Cerrar Reader

5) Cerrar Conexión

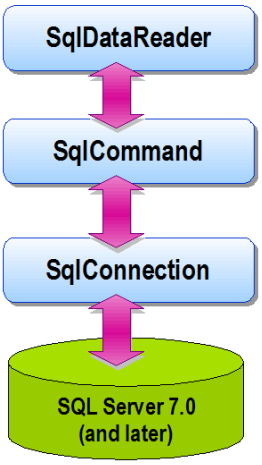


Figura 2.20 Diagrama de un escenario conectado

En un escenario desconectado, los recursos no se mantienen en el servidor mientras los datos se procesan

1) Abrir Conexión

2) Llenar DataSet mediante DataAdapter

3) Cerrar Conexión

4) Procesar DataSet

5) Abrir Conexión

6) Actualizar fuente de datos mediante DataAdapter

7) Cerrar Conexión

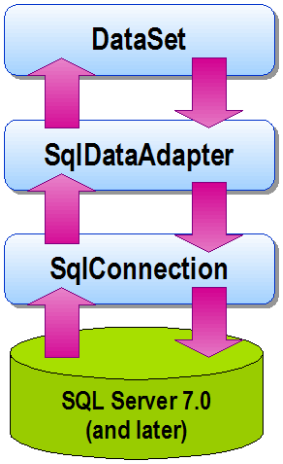


Figura 2.21 Diagrama de un escenario desconectado

Los datos para su consulta son presentados en GridView.

El nuevo control GridView es una mejora de la DataGrid de ASP.NET 1.1 que permite entre otras cosas el enlace de datos contra los controles **SQLDataSource** (como en el ejemplo) y **ObjectDataSource** (visto anteriormente).
El control SQLDataSource, hace referencia a la cadena de conexión configurada en el web.config mediante el signo $. Esta es una nueva forma de hacer referencia al contenido del archivo de configuración desde el código del cliente.

De este modo el ordenamiento, modificación de registros, Paginado, etc. se realizan automáticamente indicándolo como en el ejemplo (ver las propiedades resaltadas).

Otra de las grandes ventajas sobre su grilla predecesora es que permite la definición de múltiples campos como clave primaria

DetailsView
Es un control que visualiza un registro por vez, opcionalmente provee botones de navegación que permite navegar entre los distintos registros asociados. Además de visualizar, también permite modificar el registro actual.

FormView
Es similar al anterior a diferencia que necesita la definición de un template para representar los campos.

Visual Web Developer 2005 Express Edition provee mayor flexibilidad al momento de administrar los archivos de una aplicación Web.

File System: Permite seleccionar la carpeta del disco local donde se van a alojar las páginas Web del sitio, de esta forma no es necesario el uso de IIS.
Local IIS: Permite un manejo mucho mas simple al momento de trabajar con un servidor Web IIS. Cuando crea un proyecto o intenta abrir uno existente, Visual Studio 2005 le permite ver todos los sitios Web y aplicaciones configuradas en su máquina. (http://localhost/Site1). Las extensiones de servidor de Front Page no son necesarias para desarrollar aplicaciones Web en un IIS local.
FTP: Permite la edición y actualización de proyectos a través del protocolo estándar de transferencia de archivos FTP (File Transfer Protocol).

Sitio Remoto: Es posible publicar con un sitio Web a un servidor remoto, permitiendo mantener sincronizadas los archivos del proyecto local con los del servidor Web remoto (requiere las extensiones de servidor de Front Page).

Un servidor web es un sistema informático conectado a una red, donde se almacenan las páginas, imágenes, etc. (que forman una aplicación web) disponibles para ser visitadas por los usuarios de la red.

Internet Information Server (IIS), es el servidor Web de Microsoft que corre sobre plataformas Windows. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS

2.7 STRUCTURED QUERY LANGUAGE (SQL)

Para la creación de la base de datos se utilizo SQL Server 2005 Standard Edition que es la licencia que se adquirió en la ASF.

El lenguaje de gestión de bases de datos más conocido en la actualidad es el SQL, Structured Query Language, que es un lenguaje estandar internacional, comúnmente aceptado por los fabricantes de generadores de bases de datos.
El SQL trabaja con estructura cliente/servidor sobre una red de computadoras.
La PC Cliente es la que inicia la consulta; el Servidor atiende la consulta. De esta manera, el cliente no utiliza toda su capacidad de proceso para trabajar sino que se limita a solicitar datos al Servidor. Estas peticiones y las respuestas son transferencias de textos que cada ordenador cliente se encarga de obtener por pantalla, presentar en informes tabulados, imprimir, guardar, etc., dejando el Servidor libre.

El SQL permite:

- Definir una base de datos mediante tablas
- Almacenar información en tablas.
- Seleccionar la información que sea necesaria de la base de datos.
- Realizar cambios en la información y estructura de los datos.
- Combinar y calcular datos para conseguir la información necesaria.
- Concepto de Bases de Datos.

Existen tres ediciones principales de SQL Server:

SQLServer 2005 Enterprise Edition

SQL Server 2005 Standard Edition

SQL Server 2005 Workgroup Edition

Además de las ediciones Enterprise, Standard y Workgroup, SQL Server 2005 incluye:

SQL Server 2005 Developer Edition

SQL Server 2005 Express Edition.

La edición utilizada en la ASF es la SQLServer 2005 Enterprise Edition

Edicion SQL Server 2005	Descripción
Enterprise Edition (Disponible en versiones 32-bit y 64-bit)	Una versión comprensiva de SQL Server diseñada para altos niveles de performance. Esta edición se usa para grandes escalas, niveles empresarios, aplicaciones criticas. La Enterprise Edition contiene todos las ventajas de la edición Standard, como así también las de Enterprise, incluyendo: ✓ Failover clustering ✓ Database mirroring ✓ Snapshot databases ✓ Mirrored backups ✓ Online page and file restore ✓ Distributed partitioned views ✓ Heterogeneous replication ✓ Peer-to-peer replication
Standard Edition (Disponible en versiones 32-bit y 64-bit versions)	Diseñado para aplicaciones de trabajo de grupo y departamentales. Use esta edición si no necesita los niveles de performance y disponibilidad ofrecidos por la Enterprise Edition.
Express Edition (Disponible en 32-bit)	Una versión de SQL Server 2005 para clientes que no están conectados o aplicaciones que se ejecutan solas.
Mobile Edition	Una edición compacta que provee administración de datos para equipos inteligentes. Esta edición es capaz de replicar datos con SQL Server 2005 y SQL Server 2000, permitiendo a los usuarios mantener un mobile data store que puede ser sincronizado con enterprise data.
Developer Edition (Available in 32-bit and 64-bit versions)	Incluye todas las funcionalidades de la Enterprise Edition, pero esta licenciado para ser usado como un sistema de testeo y desarrollo, no como un servidor de producción. Use esta edición para desarrollar y testear soluciones para base de datos.

Figura 2.22 Ediciones de SQL SERVER 2005

2.8 BASE DE DATOS

Una base de datos se puede definir como un conjunto de información que pertenece al mismo contexto, que se encuentra agrupada ó almacenada para su uso posterior.

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

Desde el punto de vista más formal, podríamos definir una base de datos como un conjunto de datos estructurados, fiables y homogéneos, organizados independientemente en máquina, accesibles a tiempo real, compartibles por usuarios concurrentes que tienen necesidades de información diferente y no predecible en el tiempo.

De forma sencilla podemos indicar que una base de datos no es más que un conjunto de información relacionada que se encuentra agrupada o estructurada.

El archivo por sí mismo, no constituye una base de datos, sino más bien la forma en que está organizada la información es la que da origen a la base de datos. Las bases de datos manuales, pueden ser difíciles de gestionar y modificar. Por ejemplo, en una guía de teléfonos no es posible encontrar el número de una persona si no sabemos su apellido, aunque conozcamos su domicilio.

Del mismo modo, en un archivo de pacientes en el que la información esté ordenada por el nombre de los mismos, será una tarea bastante engorrosa encontrar todos los pacientes que viven en una zona determinada. Los problemas expuestos anteriormente se pueden resolver creando una base de datos informatizada.

Las aplicaciones más usuales de bases de datos, son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Tipos de bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:
Según la variabilidad de los datos almacenados

- Bases de datos estáticas. Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.
- Bases de datos dinámicas. Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

2.8.1 Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

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

Bases de datos jerárquicas. Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

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

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

Base de datos relacional. Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.
Base de Datos Relacional
Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas".
Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenan los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red).

Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos.

La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información.
El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Un RDBMS es un Sistema Administrador de Bases de Datos Relacionales. RDBMS viene del acrónimo en inglés Relational Data Base Management System.

Los RDBMS proporcionan el ambiente adecuado para gestionar una base de datos.

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

Bases de datos documentales. Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes. Tesaurus es un sistema de índices optimizado para este tipo de bases de datos.

Base de datos deductivas. Un sistema de base de datos deductivas, es un sistema de base de datos pero con la diferencia que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos.
Bases de datos distribuidas. La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así, por ejemplo a distintas universidades, sucursales de tiendas, etcétera.

2.8.2 Sistema de gestión de base de datos

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa.

Propósito
El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos.

Objetivos

Existen distintos objetivos que deben cumplir los DBMS:

Abstracción de la información. Los DBMS ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. Lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los DBMS deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los DBMS disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación. Los DBMS deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un DBMS debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el DBMS tarda en darnos la información solicitada y en almacenar los cambios realizados.

Ventajas

1. Facilidad de manejo de grandes volumen de información.
2. Gran velocidad en muy poco tiempo.
3. Independencia del tratamiento de información.
4. Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
5. No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
6. Integridad referencial el terminar los registros

2.8.3 Diseño de Base de Datos

El diseño de una base de datos requiere un conocimiento profundo de las funciones empresariales que se desean modelar, así como de las características y conceptos de base de datos que se desea utilizar para representar esas funciones empresariales. Asegúrese de diseñar con precisión la base de datos que utilizará para modelar el negocio porque el cambio del diseño de la base de datos una vez implementada puede requerir mucho tiempo. Por otra parte, una base de datos bien diseñada ofrece un mejor rendimiento.

¿Qué es un buen diseño de base de datos?
El proceso de diseño de una base de datos se guía por algunos principios. El primero de ellos es que se debe evitar la información duplicada o, lo que es lo mismo, los datos redundantes, porque malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias. El segundo principio es que es importante que la información sea correcta y completa. Si la base de datos contiene información incorrecta, los informes que recogen información de la base de datos contendrán también información incorrecta y, por lo tanto, las decisiones que se tome a partir de esos informes estarán mal fundamentadas.

Un buen diseño de base de datos es, por lo tanto, aquél que:

Divide la información en tablas basadas en temas para reducir los datos redundantes.
Proporciona a RDBMS la información necesaria para reunir la información de las tablas cuando así se precise.
Ayuda a garantizar la exactitud e integridad de la información.
Satisface las necesidades de procesamiento de los datos y de generación de informes.

2.8.4 Creación de claves principales

Cada tabla debe incluir una columna o conjunto de columnas que identifiquen inequívocamente cada fila almacenada en la tabla. Ésta suele ser un número de identificación exclusivo, como un número de identificador de empleado o un número de serie. En la terminología de bases de datos, esta información recibe el nombre de clave principal de la tabla. El RDBMS utiliza los campos de clave principal para asociar rápidamente datos de varias tablas y reunir automáticamente esos datos.

Si ya tiene un identificador exclusivo para una tabla, como un número de producto que identifica inequívocamente cada producto del catálogo, puede utilizar ese identificador como clave principal de la tabla, pero sólo si los valores de esa columna son siempre diferentes para cada registro. No puede tener valores duplicados en una clave principal. Por ejemplo, no utilice los nombres de las personas como clave principal, ya que los nombres no son exclusivos. Es muy fácil que dos personas tengan el mismo nombre en la misma tabla.

Una clave principal siempre debe tener un valor. Si el valor de una columna puede quedar sin asignar o vacío (porque no se conoce) en algún momento, no puede utilizarlo como componente de una clave principal.

Debe elegir siempre una clave principal cuyo valor no cambie. En una base de datos con varias tablas, la clave principal de una tabla se puede utilizar como referencia en las demás tablas (clave externa o secundaria). Si la clave principal cambia, el cambio debe aplicarse también a todos los lugares donde se haga referencia a la clave.

Usar una clave principal que no cambie reduce la posibilidad de que se pierda su sincronización con las otras tablas en las que se hace referencia a ella.

A menudo, se utiliza como clave principal un número único arbitrario. Por ejemplo, puede asignar a cada pedido un número de pedido distinto. La única finalidad de este número de pedido es identificar el pedido. Una vez asignado, nunca cambia.

Si piensa que no hay ninguna columna o conjunto de columnas que pueda constituir una buena clave principal, considere la posibilidad de utilizar una columna que tenga el tipo de datos Autonumérico. Cuando se utiliza el tipo de datos Autonumérico, El RDBMS asigna automáticamente un valor. Este tipo de identificador no es "fáctico", es decir, no contiene información objetiva sobre la fila que representa. Los identificadores de este tipo son perfectos para usarlos como claves principales, ya que no cambian. Una clave principal que contiene datos sobre una fila, como un número de teléfono o el nombre de un cliente, es más probable que cambie, ya que la propia información "fáctica" podría cambiar.

Una columna establecida en el tipo de datos Autonumérico suele constituir una buena clave principal. No hay dos identificadores de producto iguales.

En algunos casos, tal vez considere conveniente utilizar dos o más campos juntos como clave principal de una tabla. Por ejemplo, una tabla Detalles de pedidos que contenga artículos de línea de pedidos tendría dos columnas en su clave principal: Id. de pedido e Id. de producto. Cuando una clave principal está formada por más de una columna se denomina clave compuesta.

2.8.5 Crear relaciones entre las tablas

Ahora que ha dividido la información en tablas necesita un modo de reunir de nuevo la información de forma provechosa. Por ejemplo, el siguiente formulario incluye información de varias tablas.

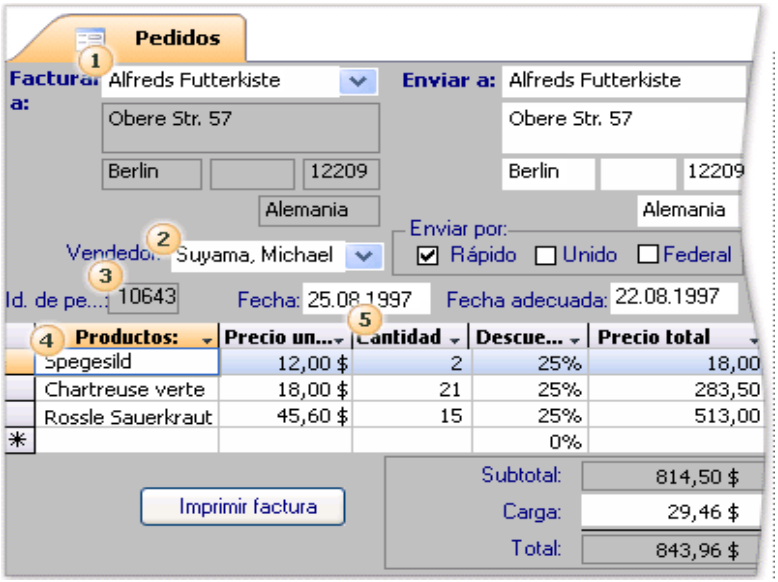


Figura 2.23 Relaciones entre tablas

1. La información de este formulario procede de la tabla Clientes...
2. ...la tabla Empleados...
3. ...la tabla Pedidos...
4. ...la tabla Productos...
5. ...y la tabla Detalles de pedidos.

2.8.6 Crear una relación de uno a varios

Considere este ejemplo: las tablas Proveedores y Productos de la base de datos de pedidos de productos. Un proveedor puede suministrar cualquier número de productos y, por consiguiente, para cada proveedor representado en la tabla Proveedores, puede haber muchos productos representados en la tabla Productos. La relación entre la tabla Proveedores y la tabla Productos es, por tanto, una relación de uno a varios.

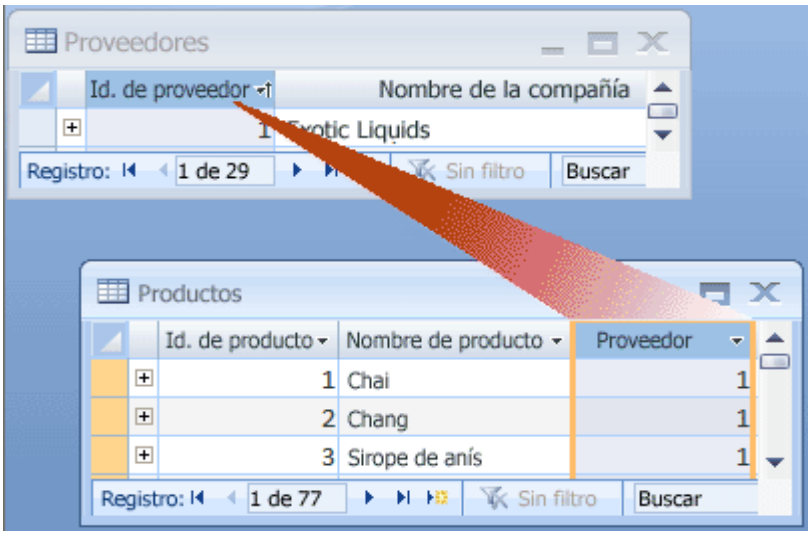


Figura 2.24 Relación de uno a varios

Para representar una relación de uno a varios en el diseño de la base de datos, tome la clave principal del lado "uno" de la relación y agréguela como columna o columnas adicionales a la tabla en el lado "varios" de la relación. En este caso, por ejemplo, agregaría la columna Id. de proveedor de la tabla Proveedores a la tabla Productos. SQL Server utiliza entonces el número de identificador de proveedor de la tabla Productos para localizar el proveedor correcto de cada producto.

La columna Id. de proveedor de la tabla Productos se denomina clave externa.
Una clave externa es la clave principal de otra tabla.

La columna Id. de proveedor de la tabla Productos es una clave externa porque también es la clave principal en la tabla Proveedores.

2.8.7 Crear una relación de varios a varios

Un solo pedido puede incluir varios productos. Por otro lado, un único producto puede aparecer en muchos pedidos. Por tanto, para cada registro de la tabla Pedidos puede haber varios registros en la tabla Productos. Y para cada registro de la tabla Productos puede haber varios registros en la tabla Pedidos. Este tipo de relación se denomina relación de varios a varios porque para un producto puede haber varios pedidos, y para un pedido puede haber varios productos. Tenga en cuenta que para detectar las relaciones de varios a varios entre las tablas, es importante que considere ambas partes de la relación.

Los temas de las dos tablas (pedidos y productos) tienen una relación de varios a varios. Esto presenta un problema. Para comprender el problema, imagine qué sucedería si intenta crear la relación entre las dos tablas agregando el campo Id. de producto a la tabla Pedidos. Para que haya más de un producto por pedido, necesita más de un registro en la tabla Pedidos para cada pedido y, en ese caso, tendría que repetir la información de pedido para cada fila relacionada con un único pedido, lo que daría lugar a un diseño ineficaz que podría producir datos inexactos. El mismo problema aparece si coloca el campo Id. de pedido en la tabla Productos: tendría varios registros en la tabla Productos para cada producto. ¿Cómo se soluciona este problema?

La solución a este problema consiste en crear una tercera tabla que descomponga la relación de varios a varios en dos relaciones de uno a varios. Inserte la clave principal de cada una de las dos tablas en la tercera tabla y, por consiguiente, la tercera tabla va a registrar todas las apariciones o instancias de la relación.

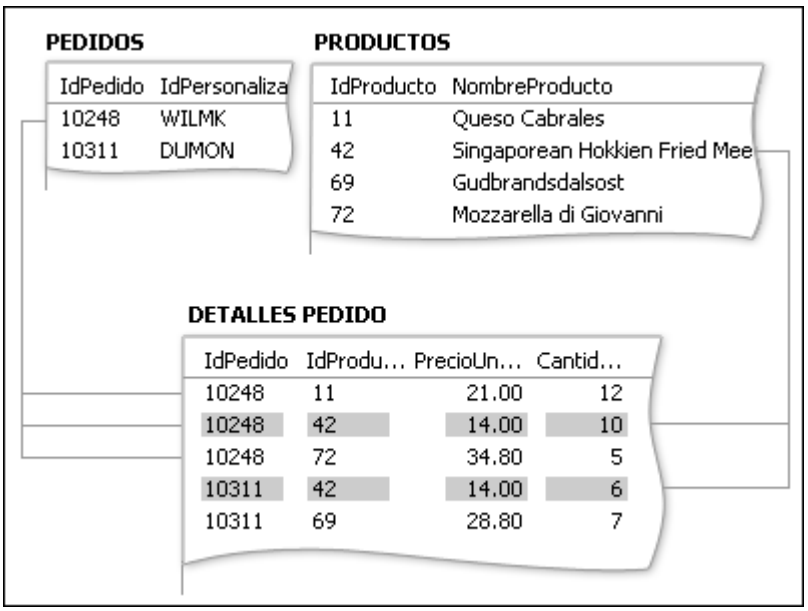


Figura 2.25 Relación de varios a varios

Cada registro de la tabla Detalles de pedidos representa un artículo de línea de un pedido. La clave principal de la tabla Detalles de pedidos consta de dos campos: las claves externas de las tablas Pedidos y Productos. El campo Id. de pedido no se puede utilizar en solitario como clave principal, ya que un pedido puede tener varios artículos de línea. El identificador de pedido se repite para cada artículo de línea del pedido, por lo que el campo no contiene valores únicos.

Tampoco serviría utilizar solamente el campo Id. de producto, porque un producto puede aparecer en varios pedidos. Pero los dos campos juntos producen un valor exclusivo para cada registro.

En la base de datos de ventas de productos, la tabla Pedidos y la tabla Productos no se relacionan directamente entre sí, sino indirectamente a través de la tabla Detalles de pedidos. La relación de varios a varios entre los pedidos y los productos se representa en la base de datos mediante dos relaciones de uno a varios:

La tabla Pedidos y la tabla Detalles de pedidos tienen una relación de uno a varios. Cada pedido tiene varios artículos de línea, pero cada artículo está asociado a un único pedido.
La tabla Productos y la tabla Detalles de pedidos tienen una relación de uno a varios. Cada producto puede tener varios artículos asociados, pero cada artículo de línea hace referencia únicamente a un producto.

Desde la tabla Detalles de pedidos se puede determinar todos los productos de un determinado pedido, así como todos los pedidos de un determinado producto.

Después de incorporar la tabla Detalles de pedidos, la lista de tablas y campos sería similar a la siguiente:

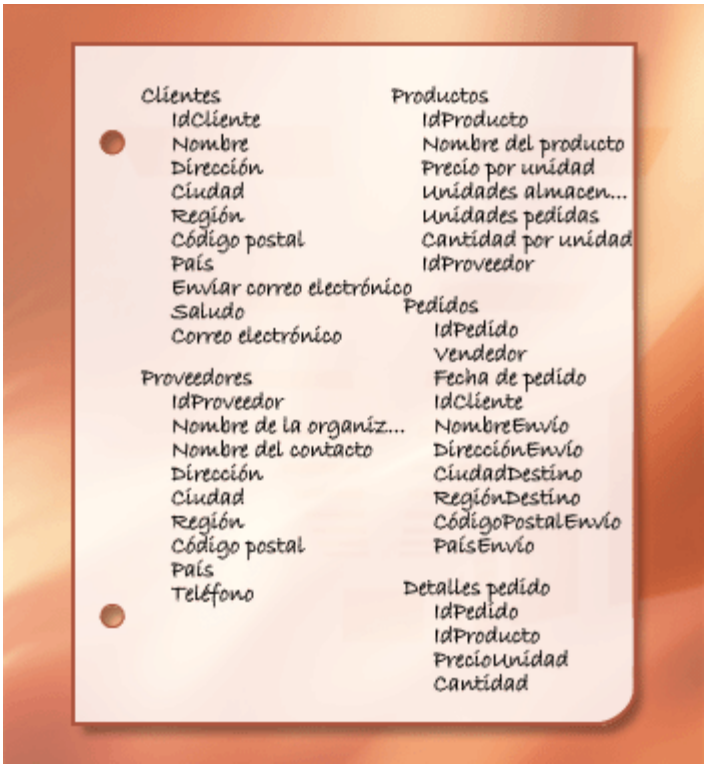


Figura 2.26 Tabla de detalles de pedidos

2.8.8 Crear una relación de uno a uno

Otro tipo de relación es la relación de uno a uno. Suponga, por ejemplo, que necesita registrar información complementaria sobre productos que apenas va a necesitar o que sólo se aplica a unos pocos productos. Como no necesita la información con frecuencia, y como almacenar la información en la tabla Productos crearía un espacio vacío para todos los productos que no necesitan esa información, la coloca en una tabla distinta. Al igual que en la tabla Productos, utiliza el identificador de producto como clave principal. La relación entre esta tabla complementaria y la tabla Productos es una relación de uno a uno. Para cada registro de la tabla Productos hay un único registro coincidente en la tabla complementaria. Cuando identifique esta relación, ambas tablas deben compartir un campo común.

Cuando necesite crear una relación de uno a uno en la base de datos, considere si puede incluir la información de las dos tablas en una tabla. Si no desea hacer eso por algún motivo (quizás porque se crearía una gran cantidad de espacio vacío), puede representar esa relación en su diseño guiándose por las pautas siguientes:

Si las dos tablas tienen el mismo tema, probablemente podrá definir la relación utilizando la misma clave principal en ambas tablas.

Si las dos tablas tienen temas diferentes con claves principales distintas, elija una de las tablas (cualquiera de ellas) e inserte su clave principal en la otra tabla como clave externa.

Determinar las relaciones entre las tablas le ayudará a asegurarse de que tiene las tablas y columnas correctas. Cuando existe una relación de uno a uno o de uno a varios, las tablas implicadas deben compartir una o varias columnas comunes. Cuando la relación es de varios a varios, se necesita una tercera tabla para representar la relación.

2.9 OBJETOS DE LA BASE DE DATOS

2.9.1 Tablas

En una base de datos la información se organiza en tablas, que son filas y columnas similares a las de los libros contables o a las de las hojas de cálculo.

Una base de datos simple puede que sólo contenga una tabla, pero generalmente las bases de datos necesitan varias tablas. Por ejemplo, podría existir una base de datos con las siguientes tablas: una tabla con información sobre productos, otra con información sobre pedidos y una tercera con información sobre clientes.

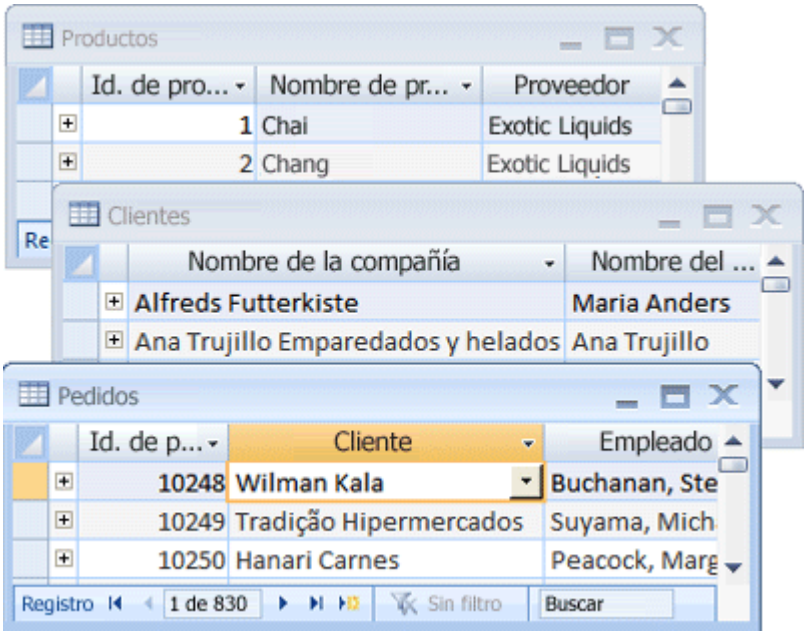


Figura 2.27 Tablas

Cada fila de la tabla recibe también el nombre de registro (o tupla) y cada columna se denomina también campo.

Un registro es una forma lógica y coherente de combinar información sobre algún tema.
Un campo es un elemento único de información: un tipo de elemento que aparece en cada registro.

En la tabla Productos, por ejemplo, cada fila o registro contendría información sobre un producto, y cada columna contendría algún dato sobre ese producto, como su nombre o el precio.

2.9.2 Vistas

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla real, una vista consta de un conjunto de columnas y filas de datos con un nombre. Sin embargo, a menos que esté indizada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Una vista actúa como filtro de las tablas subyacentes a las que se hace referencia en ella. La consulta que define la vista puede provenir de una o de varias tablas, o bien de otras vistas de la base de datos actual u otras bases de datos. Asimismo, es posible utilizar las consultas distribuidas para definir vistas que utilicen datos de orígenes heterogéneos. Esto puede resultar de utilidad, por ejemplo, si se desea combinar datos de estructura similar que proceden de distintos servidores, cada uno de los cuales almacena los datos para una región distinta de la organización.

En esta ilustración se muestra una vista basada en dos tablas:

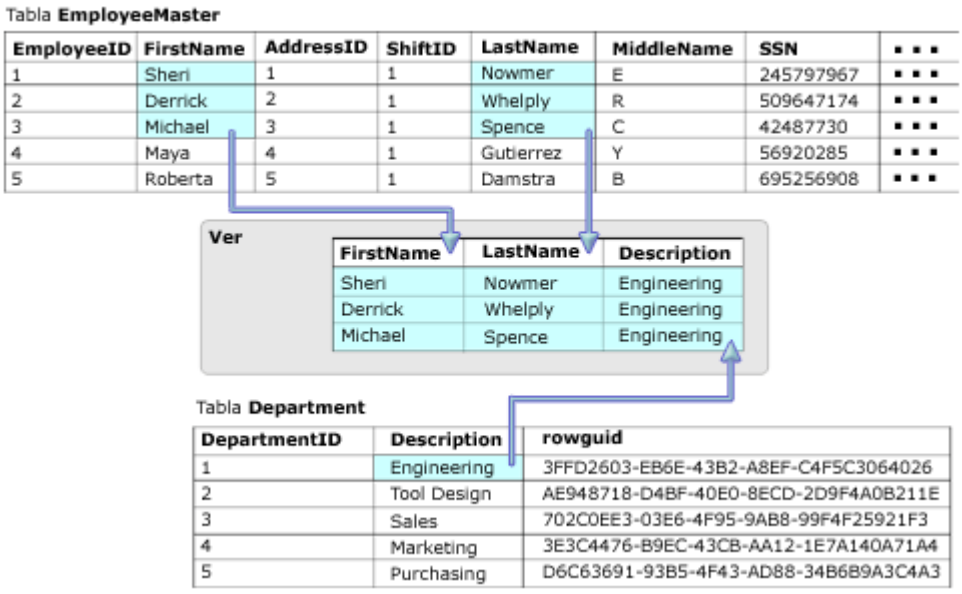


Figura 2.28 Vistas

2.9.3 Procedimientos Almacenados

Cuando crea una aplicación con Microsoft SQL Server 2005, el lenguaje de programación Transact-SQL es la principal interfaz de programación entre las aplicaciones y la base de datos de Microsoft SQL Server. Cuando utiliza programas Transact-SQL, dispone de dos métodos para almacenar y ejecutar los programas.
Puede almacenar los programas localmente y crear aplicaciones que envían los comandos a SQL Server y procesan los resultados.
Puede almacenar los programas como procedimientos almacenados en SQL Server y crear aplicaciones que ejecutan los procedimientos almacenados y procesan los resultados.

Los procedimientos almacenados de Microsoft SQL Server son similares a los procedimientos de otros lenguajes de programación en el sentido en que pueden:
Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.
Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.
Devolver un valor de estado a un lote o a un procedimiento que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores (y el motivo de éstos).
Puede utilizar la instrucción EXECUTE de Transact-SQL para ejecutar un procedimiento almacenado. Los procedimientos almacenados difieren de las funciones en que no devuelven valores en lugar de sus nombres ni pueden utilizarse directamente en una expresión.

Utilizar procedimientos almacenados en SQL Server en vez de programas Transact-SQL almacenados localmente en equipos cliente presenta las siguientes ventajas:

Se registran en el servidor.

Pueden incluir atributos de seguridad (como permisos) y cadenas de propiedad; además se les pueden asociar certificados. Los usuarios pueden disponer de permiso para ejecutar un procedimiento almacenado sin necesidad de contar con permisos directos en los objetos a los que se hace referencia en el procedimiento.
Mejoran la seguridad de la aplicación. Los procedimientos almacenados con parámetros pueden ayudar a proteger la aplicación ante ataques por inyección de código SQL.
Permiten una programación modular. Puede crear el procedimiento una vez y llamarlo desde el programa tantas veces como desee. Así, puede mejorar el mantenimiento de la aplicación y permitir que las aplicaciones tengan acceso a la base de datos de manera uniforme.
Constituyen código con nombre que permite el enlace diferido. Esto proporciona un nivel de direccionamiento indirecto que facilita la evolución del código.
Pueden reducir el tráfico de red. Una operación que necesite centenares de líneas de código Transact-SQL puede realizarse mediante una sola instrucción que ejecute el código en un procedimiento, en vez de enviar cientos de líneas de código por la red.

2.10 CONSULTAS

2.10.1 Creación de Tablas Nuevas

CREATE TABLE tabla (
campo1 tipo (tamaño) índice1,
campo2 tipo (tamaño) índice2,... ,
índice multicampo , ...)

2.10.2 Select

La sentencia SELECT sirve para extraer un conjunto de registros (filas) de una o más tablas de una base de datos. En su forma simple debe incluir una cláusula SELECT (especifica las columnas que se van a retornar), FROM (especifica la tabla de la base de datos) y WHERE (especifica una condición que deben cumplir las filas retornadas).
Es decir, su sintaxis básica es:
SELECT [DISTINCT] * | lista_de_columnas
FROM nombre_tabla
[WHERE condición]

Es posible filtrar las filas de una consulta cuando no se requiera obtener todas las filas de la tabla, sino aquellas que cumplan una determinada condición. Para ello se utiliza la cláusula WHERE. Esta cláusula contiene una condición que se debe cumplir. Si la condición es verdadera se retorna la fila que cumple dicha condición.

2.10.3 Asignación de nombres significativos a las columnas

Para cambiar los rótulos de las columnas que se obtienen como resultado de una consulta se utiliza la palabra clave AS seguido del rótulo que desea que se muestre. Algunas bases de datos no exigen el uso de AS (Oracle, por ejemplo), sin embargo lo permiten.
Alternativamente podemos hacer uso del operador BETWEEN que sirve para comprobar si un valor se encuentra entre dos valores dados. En ambos casos el resultado obtenido es el mismo.

Mediante el operador IN es posible realizar consultas que comprueban si un valor dado coincide con uno o más de una lista de valores especificada.

Las consultas de correspondencia con un patrón recuperan filas para las que el contenido de una columna de tipo cadena de caracteres se corresponde con una cadena dada, es decir, comprueban si el valor de la columna se ajusta a un patrón especificado. En estos casos hacemos uso del operador LIKE.
Por ejemplo, para obtener un listado de los empleados cuyo nombre empiece con la letra "S" escribimos:

```
SELECT *
FROM emp
WHERE ename LIKE 'S%'
```

```
EMPNO  ENAME  JOB       MGR   HIREDATE   SAL   COMM   DEPTNO
------------------------------------------------------------------------
7369   SMITH  CLERK     7902  1980-12-17 800   NULL   20
7788   SCOTT  ANALYST   7566  1987-04-19 3000  NULL   20
```

Para listar todas las filas que contengan valores nulos en una columna específica usamos la condición IS NULL. Por ejemplo:

```
SELECT *
FROM emp
WHERE comm IS NULL
```

```
EMPNO  ENAME  JOB       MGR   HIREDATE   SAL   COMM   DEPTNO
------------------------------------------------------------------------
7369   SMITH  CLERK     7902  1980-12-17 800   NULL   20
7566   JONES  MANAGER   7839  1981-04-02 2975  NULL   20
7698   BLAKE  MANAGER   7839  1981-05-01 2850  NULL   30
7782   CLARK  MANAGER   7839  1981-06-09 2450  NULL   10
7788   SCOTT  ANALYST   7566  1987-04-19 3000  NULL   20
7839   KING   PRESIDENT NULL  1981-11-17 5000  NULL   10
7876   ADAMS  CLERK     7788  1987-05-23 1100  NULL   20
7900   JAMES  CLERK     7698  1981-12-03 950   NULL   30
7902   FORD   ANALYST   7566  1981-12-03 3000  NULL   20
7934   MILLER CLERK     7782  1982-01-23 1300  NULL   10
```

Para mostrar filas que no contengan valores nulos usamos la condición IS NOT NULL, por ejemplo:

```
SELECT *
FROM emp
WHERE comm IS NOT NULL
```

```
EMPNO  ENAME  JOB       MGR   HIREDATE   SAL   COMM   DEPTNO
--------------------------------------------------------------------
7499   ALLEN  SALESMAN   7698  1981-02-20 1600  300    30
7521   WARD   SALESMAN   7698  1981-02-22 1250  500    30
7654   MARTIN SALESMAN   7698  1981-09-28 1250  1400   30
7844   TURNER SALESMAN   7698  1981-09-08 1500  0      30
```

2.10,4 Consultas de Actualización

Las consultas de actualización son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir y borrar y modificar registros.

Delete

Crea una consulta de eliminación
que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es:

```
DELETE Tabla.* FROM Tabla WHERE criterio
```

Insert Into

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: Insertar un único registro ó Insertar en una tabla los registros contenidos en otra tabla.

Para insertar un único Registro: En este caso la sintaxis es la siguiente:

```
INSERT INTO Tabla (campo1, campo2, ..,
campoN)
VALUES (valor1, valor2, ..., valorN)
```

Esta consulta graba en el campo1 el valor1, en el campo2 y valor2 y así sucesivamente. Hay que prestar especial atención a acotar entre comillas simples (') los valores literales (cadenas de caracteres) y las fechas indicarlas en formato mm-dd-aa y entre caracteres de almohadillas (#).
Para insertar Registros de otra Tabla:
En este caso la sintaxis es:

```
INSERT INTO Tabla [IN base_externa]
(campo1, campo2, ..., campoN)
SELECT TablaOrigen.campo1, TablaOrigen.campo2, ..., TablaOrigen.campoN
FROM TablaOrigen
```

Update

Update cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2,
... CampoN=ValorN
WHERE Criterio;

2.10.5 Consultas de Unión

Las vinculaciones entre tablas se realizan mediante la cláusula INNER que combina registros de dos tablas siempre que haya concordancia de valores en un campo común. Su sintaxis es:

SELECT campos FROM tb1 INNER JOIN tb2
ON tb1.campo1 comp tb2.campo2
En donde:

tb1, tb2:
Son los nombres de las tablas desde las que se combinan los registros.

campo1, campo2: Son los nombres de los campos que se combinan. Si no son numéricos, los campos deben ser del mismo tipo de datos y contener el mismo tipo de datos, pero no tienen que tener el mismo nombre.

Se puede utilizar una operación INNER JOIN en cualquier cláusula FROM. Esto crea una combinación por equivalencia, conocida también como unión interna. Se combinan los registros de dos tablas siempre que haya concordancia de valores en un campo común a ambas tablas. Se puede utilizar INNER JOIN con las tablas Departamentos y Empleados para seleccionar todos los empleados de cada departamento. Por el contrario, para seleccionar todos los departamentos (incluso si alguno de ellos no tiene ningún empleado asignado) se emplea LEFT JOIN o todos los empleados (incluso si alguno no está asignado a ningún departamento), en este caso RIGHT JOIN.

DESARROLLO DE ACTIVIDADES Y PROYECTOS EN LA AUDITORÌA SUPERIOR DE LA FEDERACIÒN.

En el sistema CheckUp se siguió el siguiente trayecto de actividades:

Análisis de requerimientos para un nuevo sistema llamado CheckUp
Inicio de programación de la aplicación CheckUp en asp.net
Creación de páginas maestras para la aplicación
Creación de la base de datos
Creación de la autentificación de usuarios por medio de su alias de red
Creación de un calendario dinámico que presenta días disponibles para hacer reservaciones por medio de consulta a base de datos
Creación de un calendario dinámico que presenta días disponibles para hacer reservaciones por medio de consulta a base de datos
Creación de consultas de horarios por día y mes para hacer reservación.
Consultas de horarios por día y mes para hacer reservaciones.
Actualización a la base de datos para insertar fecha y horario de reservación
Creación de actualización a la base de datos para insertar fecha y horario de reservación
Creación de modulo para cancelar reservaciones y comprobar las que ya están hechas
Creación de un modulo para administración en el cual se hace consulta por mes de todas las reservaciones hechas
Creación de un modulo para exportar consultas a excel
Creación de un modulo para administración en el cual se puedan cancelar reservaciones aun pasado el periodo que se tiene para cancelarlas
Creación de modulo para quitar dias del calendario ya sea porque no es dia laboral o porque ya esta lleno ese dia
Pruebas a los sistemas en un ambiente local
Instalación en el servidor

En el sistema AIB se siguió el siguiente trayecto de actividades:

Análisis de requerimientos para un nuevo sistema llamado AIB
Inicio de programación de la aplicación AIB en asp.net
Creación de la base de datos con sql
Creación de un web service.
Creación de modulo para consultas por auditorias
Creación de modulo de consultas por recensiones
Creación de modulo de consultas por extractos
Creación de la presentación de datos en GridView
Creación de modulo para exportar consultas a excel
Creación de un modulo para petición de información por correo
Configuración de detalles del SMTP
Pruebas al sistema en un ambiente local
Instalación en el servidor

III.- CONCLUSIÒN
3.1 CONCLUSIONES DE SERVICIO SOCIAL EN LA ASF

Mi experiencia en el desarrollo del servicio social en la ASF fue muy importante para adquirir experiencia dentro de la programación de sistemas, ya que me encontré en todas las etapas del desarrollo de sistemas desde el análisis hasta el mantenimiento que deben tener los sistemas informáticos con el paso del tiempo y el encontrar que es lo que en realidad requería el usuario final.

Además el saber como es el campo laboral, el cual es exigente y se necesita de mucha preparación, lo mas importante y sobretodo experiencia ya que sin esta es complicado que alguna empresa te contrate.

La formación académica dentro de la facultad a veces puede no ser suficiente, por lo cual debes ser autodidacta para ampliar tus habilidades y conocimientos, la tecnología de .net no se enseña en la facultad, sin embargo gracias a esa inquietud de seguir aprendiendo programación estudie y me fui muy útil para entrar y realizar el servicio social dentro de la ASF.

En la entrevista se me realizo un examen un poco sencillo pero sin los conocimientos básicos de la tecnología seria imposible haberlo aprobado. No obstante la formación dada en la Facultad es muy útil porque a mi parecer son conceptos básicos y con ellos tienes las bases para desarrollar tu formación y aptitudes hasta donde tu desees.

Así mismo .NET es una herramienta de desarrollo que tiene posibilidad de poder usar cualquier lenguaje que compile a MSIL es realmente una ventaja. El poder desarrollar libremente sin tener que tomar decisiones técnicas a un lenguaje es una verdadera "innovación", las aplicaciones tienen un rendimiento muy bueno en comparación con competidores como lo es java. A pesar de que se tienen que pagar licencias para las versiones completas, las versiones gratuitas también son suficientes como la llamada Visual Studio Express y se pueden bajar los lenguajes independientemente. Según la página de Microsoft son versiones 100% gratuitas, tanto para uso como para desarrollo de aplicaciones comerciales; la diferencia está en las características y/o herramientas que trae.

El Servicio Social es un primer acercamiento al campo de desarrollo profesional por lo cual es muy importante realizarlo y encontrar un buen lugar donde hacerlo ya que puede ayudarte mucho para el futuro profesional.

GLOSARIO

ADO.NET [ADO.NET]

Conjunto de tecnologías de acceso a datos incluidas en las bibliotecas de clases de .NET Framework que proporcionan acceso a datos relacionales y a XML. ADO.NET está formado por clases que conforman DataSet (como tablas, filas, columnas, relaciones, etc.), proveedores de datos de .NET Framework y definiciones de tipos personalizados (como SqlTypes para SQL Server).

API

(Interfaz para programas de aplicación) Una serie de reglamentos y acuerdos que nos definen la manera en cómo llamar determinado servicio desde cierto programa.

aplicación Web ASP.NET [ASP.NET Web application]

Aplicación que procesa solicitudes HTTP (solicitudes Web) y se ejecuta en ASP.NET. Una aplicación Web ASP.NET puede incluir páginas ASP.NET, servicios Web XML, controladores HTTP y módulos HTTP.

archivo de código subyacente [code-behind file]

Archivo de código que contiene la clase de página que implementa la lógica de programa de una aplicación de formularios Web Forms o de formularios Web Forms móviles de ASP.NET.

archivo de configuración [configuration file]

Archivo XML con la extensión .config que contiene la configuración de las opciones para una aplicación o sitio Web. Los archivos de configuración comunes incluyen Machine.config y Web.config.

Array(matriz)

Los arrays (o matrices) son un tipo de variable que permiten tener más de un elemento, (o valor en su interior), a los que se pueden acceder mediante un índice.Un array también es el tipo en el que se basan todas las matrices o arrays.

ASP.NET [ASP.NET]

Conjunto de tecnologías de Microsoft .NET Framework para crear aplicaciones Web y servicios Web XML. Las páginas ASP.NET se ejecutan en el servidor y generan lenguaje de marcado (como HTML, WML o XML) que se envía a un explorador móvil o de escritorio. Las páginas ASP.NET utilizan un modelo de programación compilado y basado en eventos que mejora el rendimiento y permite la separación de la lógica de aplicación y de la interfaz de usuario. Las páginas ASP.NET y los archivos de servicios Web XML creados con ASP.NET contienen lógica de servidor (en vez de lógica de cliente) escrita en Visual Basic, C# o cualquier lenguaje compatible con .NET. Las aplicaciones Web y los servicios Web XML aprovechan las funciones de Common Language Runtime, como la seguridad de tipos, la herencia, la interoperabilidad entre lenguajes, el control de versiones y la seguridad integrada.

autenticación [authentication]

En la seguridad de .NET Framework, proceso de descubrir y comprobar la identidad de un principal mediante el examen de las credenciales del usuario y su consulta a una autoridad determinada.

autopostback [autopostback]

En controles de servidor ASP.NET, valor de configuración que hace que el control envíe la página cuando el usuario interactúa con el control. (De forma predeterminada, sólo los controles de botón producen una devolución de datos.)

Base de datos

(DataBase). Conjunto de datos relacionados que se almacenan de forma que se pueda acceder a ellos de manera sencilla, con la posibilidad de relacionarlos, ordenarlos en base a diferentes criterios, etc. Las bases de datos son uno de los grupos de aplicaciones de productividad personal más extendidos. Entre las más conocidas pueden citarse dBase, Paradox, Access y Aproach, para entornos PC, y Oracle, ADABAS, DB/2, Informix o Ingres, para sistemas medios y grandes.

biblioteca de clases de .NET Framework [.NET Framework class library]

Biblioteca de clases, interfaces y tipos de valor incluidos en Microsoft .NET Framework SDK. Esta biblioteca brinda acceso a la funcionalidad del sistema y es la base sobre la que se crean las aplicaciones, los componentes y los controles de .NET Framework.

C# [C#]

Lenguaje de programación diseñado para crear aplicaciones empresariales que se ejecutan en .NET Framework. C#, que es una evolución de C y C++, garantiza la seguridad de tipos y está orientado a objetos. Puesto que se compila como código administrado, aprovecha los servicios de Common Language Runtime, como interoperabilidad de lenguaje, seguridad y recolección de elementos no utilizados.

caché de ensamblados global (GAC) [global assembly cache (GAC)]

Caché de código para todo el equipo que almacena los ensamblados instalados específicamente para ser compartidos por varias aplicaciones del equipo. Las aplicaciones implementadas en la caché de ensamblados global deben tener nombres seguros.

caché del ensamblado [assembly cache]

Caché de código utilizada para el almacenamiento simultáneo de ensamblados. La caché consta de dos partes: la caché de ensamblados global contiene ensamblados que se instalan explícitamente para compartirse entre varias aplicaciones del equipo y la caché de descarga almacena código descargado desde Internet o desde sitios de la intranet, aislado de la aplicación que causó la descarga, de forma que el código descargado en nombre de una aplicación o de una página no afecte a otras aplicaciones.

clase [class]

Tipo de referencia que encapsula datos (constantes y campos) y el comportamiento (métodos, propiedades, indizadores, eventos, operadores, constructores de instancia, constructores estáticos y destructores), y puede contener tipos anidados. Los tipos de clase admiten la herencia, un mecanismo mediante el cual una clase derivada puede extender y especializar una clase base.

clave externa [foreign key]

Clave en una tabla de base de datos que procede de otra tabla. Esta clave hace referencia a una clave concreta, normalmente la clave principal, de la tabla que se está utilizando.

código nativo [native code]

Código que se ha compilado en código máquina específico del procesador.

Compilador

Programa traductor que genera lenguaje máquina (Ver: Código Máquina) a partir de un lenguaje de programación de alto nivel basado en el lenguaje.

Programa capaz de traducir un codigo fuente, escrito en el lenguaje de alto nivel que sea, a un codigo objeto escrito en lenguaje de maquina.

Common Language Runtime [common language runtime]

Motor que es el núcleo de la ejecución de código administrado. El motor en tiempo de ejecución proporciona al código administrado servicios como integración entre varios lenguajes, seguridad de acceso a código, administración de la duración de los objetos, y compatibilidad con la depuración y la generación de perfiles.

Common Language Specification (CLS) [Common Language Specification (CLS)]

Subconjunto de funciones del lenguaje admitidas por Common Language Runtime, incluyendo funciones comunes de varios lenguajes de programación orientados a objetos. Se garantiza que las herramientas y los componentes compatibles con CLS pueden interoperar con otras herramientas y componentes compatibles con CLS.

Component Object Model

COM es una manera de implementar objetos neutral con respecto al lenguaje, de manera que pueden ser usados en entornos distintos de aquel en que fueron creados, a través de fronteras entre máquinas. Para componentes bien creados, COM permite la reutilización de objetos sin conocimiento de su implementación interna, porque fuerza a los implementadores de componentes a proveer interfaces bien definidos que están separados de la implementación. Las diferentes semánticas de reserva de memoria están acomodadas haciendo a los objetos responsables de su propia creación y destrucción por medio del contador de referencias. Se puede hacer casting entre distintos interfaces de un objeto por medio de la función QueryInterface().

Casting

Un casting (o cast) sirve para cambiar el tipo de dato del valor resultante de una expresión.

Constante

Valores numéricos o de cadena que permanecen constantes, sin posibilidad de cambiar el valor que tienen.En caso de que necesitemos cambiar el valor, se usan las variables.

delegado [delegate]

En .NET Framework, referencia a una función. Un delegado es equivalente a un puntero a función.

En WMI, nivel de suplantación de seguridad utilizado para permitir el acceso remoto que implica a más de un salto de la red.

Descripción, descubrimiento e integración universales (UDDI) [Universal Description, Discovery, and Integration (UDDI)]

Marco de trabajo independiente de la plataforma que funciona como directorio (similar a una agenda de teléfonos) y proporciona un modo de encontrar y registrar servicios Web en Internet. La especificación UDDI llama a tres elementos: las páginas blancas, que proporcionan información de contacto comercial; las páginas amarillas, que organizan los servicios Web en categorías (por ejemplo, servicios de autorización de tarjetas de crédito) y las páginas verdes, que proporcionan información técnica detallada acerca de servicios individuales. La especificación UDDI también contiene un Registro operativo, ya disponible

dominio de aplicación (AppDomain) [application domain (AppDomain)]

Límite que Common Language Runtime establece alrededor de los objetos creados dentro del mismo ámbito de aplicación (es decir, cualquier lugar de la secuencia de activaciones de objetos que empieza en el punto de entrada de la aplicación). Los dominios de aplicación ayudan a aislar los objetos creados en una aplicación de los creados en otras aplicaciones, de forma que se pueda predecir el comportamiento en tiempo de ejecución. En un único proceso pueden existir varios dominios de aplicación.

encapsulación [encapsulation]

Posibilidad de que un objeto oculte sus datos y métodos internos, haciendo que sólo sean accesibles mediante programación las partes deseadas del objeto.

ensamblado [assembly]

Conjunto de uno o varios archivos que pertenecen a una versión y se implementan como unidad. Un ensamblado es la unidad de creación principal de una aplicación .NET Framework. Todos los tipos y recursos administrados se incluyen en un ensamblado y se marcan como accesibles únicamente dentro del ensamblado o bien como accesibles desde código de otros ensamblados. Los ensamblados también juegan un papel clave en la seguridad

espacio de nombres [namespace]

Esquema de nombres lógico para agrupar los tipos relacionados. .NET Framework utiliza un esquema de nombres jerárquico para agrupar los tipos en categorías lógicas de funcionalidad relacionada, como la tecnología ASP.NET o la funcionalidad de interacción remota. Las herramientas de diseño pueden utilizar espacios de nombres para que los programadores puedan examinar y hacer referencia más fácilmente a los tipos en el código. Un ensamblado individual puede contener tipos cuyos nombres jerárquicos tienen distintas raíces de espacio de nombres y una raíz de espacio de nombres lógico puede abarcar varios ensamblados. En .NET Framework, un espacio de nombres es una comodidad para la nomenclatura lógica en tiempo de diseño, mientras que un ensamblado establece el ámbito de nombres para los tipos en tiempo de ejecución.

Estado de sesión [Session state]

En ASP.NET, almacén de variables creado en el servidor para el usuario actual; cada usuario mantiene un estado de sesión independiente en el servidor. El estado de sesión se utiliza normalmente para almacenar información específica del usuario entre las devoluciones de datos.

estado de vista [view state]

Campo en una página Web ASP.NET donde puede almacenar la configuración que es necesaria conservar entre devoluciones de datos. También suele significar estado de control.

Formularios Web Forms [Web Forms]

Marco de trabajo de página ASP.NET, compuesto por páginas Web programables (denominadas páginas de formularios Web Forms) que contienen controles de servidor reutilizables.

Formularios Windows Forms [Windows Forms]

Biblioteca de clientes de Windows.

interfaz [interface]

Tipo de referencia que define un contrato. Otros tipos implementan una interfaz para garantizar que admiten ciertas operaciones. La interfaz especifica los miembros que las clases u otras interfaces que los implementan deben suministrar. Al igual que las clases, las interfaces pueden contener como miembros métodos, propiedades, indizadores y eventos.

Lenguaje de descripción de servicios Web (WSDL) [Web Services Description Language (WSDL)]

Lenguaje de contrato basado en XML para describir los servicios de red ofrecidos por un servidor

Lenguaje de marcado extensible (XML) [Extensible Markup Language (XML)]

Subconjunto del Lenguaje de marcado generalizado estándar (SGML) optimizado para su uso a través del Web. XML proporciona un método uniforme para describir e intercambiar datos estructurados que es independiente de las aplicaciones o los proveedores.

método [method]

En WMI, función que describe el comportamiento de una clase. La inclusión de un método en una clase no garantiza una implementación del método. El calificador **Implemented** se asocia al método para indicar que una implementación está disponible para la clase. Un método también es una función incluida en una interfaz WMI.

.NET Compact Framework [.NET Compact Framework]

Entorno independiente del hardware para ejecutar programas en dispositivos informáticos con recursos insuficientes. Hereda la arquitectura completa de .NET Framework de Common Language Runtime, es compatible con un subconjunto de la biblioteca de clases de .NET Framework y contiene clases diseñadas exclusivamente para .NET Compact Framework. Entre los dispositivos que admite se encuentran asistentes de datos personales (PDA) (como Pocket PC), teléfonos móviles, dispositivos de conexión para televisión digital, dispositivos informáticos de la industria del automóvil y dispositivos incrustados de diseño personalizado generados con el sistema operativo Microsoft Windows CE.NET.

.NET Framework [.NET Framework]

Componente integral de Windows que admite la creación, implementación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. Proporciona un entorno de múltiples lenguajes basado en estándares y muy productivo para integrar las inversiones existentes con aplicaciones y servicios de la próxima generación, así como la agilidad necesaria para resolver los desafíos que suponen la implementación y el funcionamiento de las aplicaciones para Internet. .NET Framework consta de tres partes principales: Common Language Runtime, un conjunto jerárquico de bibliotecas de clases unificadas y una versión dividida en componentes de ASP denominada ASP.NET.

página ASP.NET [ASP.NET page]

Componente de una aplicación ASP.NET

página principal [master page]

En ASP.NET, página que define el diseño de un conjunto de páginas. Una página principal puede contener texto estático y controles que deben aparecer en todas las páginas. Las páginas principales se combinan en tiempo de ejecución con las páginas de contenido que definen el contenido específico de la página.

Programación Orientada a Objetos (OOP / POO)

Una forma de programar basada en la reutilización de código mediante herencia, encapsulación y polimorfismo.

recolección de elementos no utilizados (GC) [garbage collection (GC)]

Proceso de hacer un seguimiento transitivamente por todos los punteros hasta los objetos utilizados activamente con el fin de encontrar todos los objetos a los que se puede hacer referencia y, después, conseguir la reutilización de toda la memoria de montón que no se haya encontrado durante este seguimiento. El recolector de elementos no utilizados de Common Language Runtime también compacta la memoria en uso para reducir el espacio de trabajo necesario para el montón

servicios Web XML [XML Web services]

Unidades de lógica de aplicaciones que proporcionan datos y servicios a otras aplicaciones. Las aplicaciones obtienen acceso a los servicios Web XML mediante protocolos Web estándar y formatos de datos como HTTP, XML y SOAP, con independencia de cómo se implementa cada servicio Web XML. Los servicios Web XML combinan los mejores aspectos del desarrollo basado en componentes y el Web, por lo que son una base fundamental del modelo de programación de Microsoft .NET.

SOAP [SOAP]

Protocolo simple basado en XML para intercambiar información estructurada y de tipos en el Web. El protocolo no contiene semántica de aplicación ni de transporte, por lo que resulta muy modular y extensible.

Variable

Son "espacios" de memoria en la que se almacena un valor. Se usarán para guardar en memoria los valores numéricos o de cadena de caracteres que nuestro programa necesite.

Windows Management Instrumentation (WMI) [Windows Management Instrumentation (WMI)]

WMI es la extensión de Microsoft de la iniciativa Administración empresarial basada en Web (WBEM) del Grupo de trabajo de administración distribuida (DMTF), y proporciona un conjunto de interfaces para el acceso a componentes que proporcionan capacidades de administración para una empresa.

BIBLIOGRAFÍA

Dino Esposito. Introducing Microsoft ASP.NET 2.0. Italia 2005. Microsoft Press

Unidad General de Administración. Inducción Institucional. México 2006
Instituto de capacitación y desarrollo en la Fiscalización Superior

Nick Randolph, David Gardner. Visual Studio 2005, Indianapolis. Indiana 2006. Wiley Publishing, Inc.

Jesse Liberty and Alex Horovitz. Programming .NET 2.0. O'Reilly

Francisco Charte. Microsoft SQL Server 2005. Mexico 2006. Anaya.

Francisco Javier Ceballos Sierra. Microsoft c# Curso de Programacion. Madrid 2006. Ra-Ma

PÉREZ, C. Microsoft Sql Server 2005. Administración Y Análisis De Bases De Datos. México 2006. Ra-Ma

Sitio Oficial de la Auditoria Superior de la Federación
www.**asf**.gob.mx

Sitio Oficial de ASP.NET
http://www.asp.net