APOYO AL DISEÑO Y CONSTRUCCIÓN DE APLICACIONES DISTRIBUIDAS DESARROLLADAS EN VISUAL STUDIO .NET CON MULTIPLATAFORMAS DE BASE DE DATOS EN PROYECTOS SOFTWARE DE LA EMPRESA ISL INGENIEROS DE SISTEMAS LTDA.

YENNY ADRIANA GUEVARA SANDOVAL

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2007

APOYO AL DISEÑO Y CONSTRUCCIÓN DE APLICACIONES DISTRIBUIDAS DESARROLLADAS EN VISUAL STUDIO .NET CON MULTIPLATAFORMAS DE BASE DE DATOS EN PROYECTOS SOFTWARE DE LA EMPRESA ISL INGENIEROS DE SISTEMAS LTDA.

YENNY ADRIANA GUEVARA SANDOVAL

Trabajo de grado para obtener el titulo de
Ingeniera de Sistemas

Tutor
Ing. María Teresa Castro García
ISL. INGENIEROS DE SISTEMAS LTDA.

Director
Ing. Leonel Parra Pinilla
ESCUELA DE SISTEMAS - UIS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2007

Quiero dedicar este logro a mi madre ***Aura Stella*** *mi fuerza y motor, a mi padre* ***José Antonio*** *por que su fortaleza ante la adversidad me inspira, a mi novio* ***Luís Fernando*** *por sus palabras de aliento y amor incondicional a toda prueba, a mi familia a quien espero no llegar a decepcionar y sobretodo a Dios todo poderoso que hizo posible todo esto.*

P.S. *En recuerdo de mi abuelito Emilio Guevara Santos quién partió hace muchos años y al que le habría gustado verme recibir este título.*

Yenny Adriana Guevara Sandoval

AGRADECIMIENTOS

“La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.” Aristóteles

En este espacio me gustaría hacer un reconocimiento a todas y cada una de las personas que influyeron positivamente en este camino de formación profesional; en especial a:

Dios Padre Todopoderoso por permitirme alcanzar con éxito esta primera meta de mi vida y por rodearme de personas maravillosas que me apoyaron en el transcurso del proceso.

Mi madre Aura Sandoval por la comprensión, la paciencia y los sacrificios realizados que me permitieron llegar a ser quien soy.

Mi familia por el cariño, la compañía y el apoyo brindado durante todo este tiempo; el mismo que ayudo a sobrellevar la ausencia prolongada del hogar.

Ingeniera María Teresa Castro García, mi tutora y jefe inmediato durante la práctica, por la confianza depositada en mí y sus valiosas enseñanzas tanto en el plano personal como en el profesional.

Ingeniero Luís Antonio Flórez Flórez, Gerente General de ISL Ingenieros de Sistemas Ltda., por brindarme la oportunidad de realizar la practica en esta reconocida y experimentada empresa; me llevo infinidad de anécdotas y aprendizaje de mi tiempo en ella.

José Vicente Silva compañero de práctica, por el apoyo brindado en el ámbito laboral y por sus consejos, los mismos que lo llevaron a convertirse en un gran amigo.

Ingeniero Leonel Parra Pinilla por su orientación en el transcurso del proyecto y a la Escuela de Ingeniería de Sistemas de la UIS por la formación académica y personal que obtuve a mi paso por sus aulas y por apoyar la modalidad de práctica empresarial para de este modo culminar con éxito esta etapa de mi vida.

A todos los compañeros y amigos de la Universidad Industrial de Santander Sede Barrancabermeja y la División de Sistemas de Información UIS Bucaramanga por los momentos compartidos, la experiencia obtenida y los gratos recuerdos.

Por último y no por eso menos importante, a mi novio Luís Fernando: Mi vida sin tu amor y apoyo incondicional no hubiera logrado mantenerme a flote en este difícil trayecto.

CONTENIDO

Pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

RESUMEN

TÍTULO: APOYO AL DISEÑO Y CONSTRUCCIÓN DE APLICACIONES DISTRIBUIDAS DESARROLLADAS EN VISUAL STUDIO .NET CON MULTIPLATAFORMAS DE BASE DE DATOS EN PROYECTOS SOFTWARE DE LA EMPRESA ISL INGENIEROS DE SISTEMAS LTDA.[*]

AUTOR: GUEVARA SANDOVAL, Yenny Adriana[**]

PALABRAS CLAVES: Aplicación Distribuida, Visual Studio .NET, Sistema de Información Web, Arquitectura Tres Capas.

DESCRIPCIÓN

La práctica fue desarrollada en la empresa ISL Ingenieros de Sistemas Ltda. Organización dedicada al desarrollo de soluciones informáticas a la medida y a la prestación de servicios de administración y contratación de TI (*Tecnologías de la Información*). Las actividades realizadas durante la práctica giraron en torno a la construcción de un sistema de información en ambiente Web para el manejo de la venta y ocupación de servicios funerarios para un parque cementerio. La aplicación antes mencionada se encuentra desarrollada en ASP .NET y en el entorno de trabajo de Visual Studio 2005, con motor de base de datos SQL Server; uno de los objetivos más importantes del proyecto era el de apoyar la adaptación de la empresa a esta nueva herramienta que supone la instauración de una nueva ideología de trabajo.

Es de resaltar que la empresa en que se desarrolló la práctica se encuentra certificada en la norma ISO 9001:2000 por parte del ICONTEC y que todas las actividades realizadas al interior de la misma estaban regidas por los procedimientos consignados dentro de su sistema de Gestión de la Calidad.

Adicionalmente se desarrollaron funciones de atención a usuarios y soporte de aplicaciones existentes que complementaron las actividades de desarrollo software y contribuyeron al crecimiento profesional del estudiante en práctica.

---

[*] Proyecto de Grado en la modalidad de Práctica Empresarial.

[**] Escuela de Ingeniería de Sistemas e Informática UIS. Director: Ing. Leonel Parra Pinilla.

SUMMARY

TITLE: DESIGN SUPPORT AND CONSTRUCTION OF DISTRIBUTED APPLICATIONS DEVELOPED IN VISUAL STUDIO.NET WITH MULTI-PLATFORMS OF DATA BASE IN SOFTWARE PROJECTS FOR THE COMPANY "ISL INGENIEROS DE SISTEMAS LTDA."[*]

AUTHOR: GUEVARA SANDOVAL, Yenny Adriana[**]

KEY WORDS: Distributed application, Visual Studio .NET, System of Web information, three layers architecture.

DESCRIPTION

Engineering practice was developed at the company: ISL Ingenieros de Sistemas Ltda. Organization dedicated to the development of system solutions, measurement and provision of administration and contracting services of TI (*Information Technology*). Activities developed during the Engineering practice were based on the construction of an information system on web environment to handle sale and management of funeral services for a cemetery. The above mentioned application is developed on ASP .NET and in Visual Studio 2005 environment, with a SQL Server motor of data base. One of the major objectives of the project was to support the adaptation of the company to this new tool which is suppose to establish a new ideology of work.

It is important to emphasize that the company where the Engineering Practice was developed is certified by the ISO 9001:2000 regulation by the ICONTEC and all activities made were controlled by procedures consigned on its system of Quality Control.

Besides, support to users activities and application support were developed which complement activities for software development and contributed to increase professional knowledge of the student in practice.

---

[*] Degree Project in the modality of enterprise practice.

[**] Faculty of Physical – Mechanical Engineerings. Systems and Computer Engineering. Leonel Parra Pinilla.

INTRODUCCIÓN

Las aplicaciones Web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dando servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. El auge de este tipo de aplicaciones se debe principalmente a la creciente necesidad de tener información confiable, de disponibilidad inmediata y accesible.

Las empresas modernas han respondido a estas necesidades mediante la incursión progresiva en el ámbito de la Internet, primero utilizándola como una forma de dar a conocer sus productos. Y ahora buscando captar la atención de sus clientes mediante la prestación de una gama de servicios a través de este medio, convirtiendo el uso de estas tecnologías en una ventaja competitiva.

El reto ahora es encontrar los métodos de diseño e implementación de estas aplicaciones que permitan construir sistemas robustos, confiables y sobre todo seguros. Por ende es necesario un cambio en los paradigmas de diseño y desarrollo software principalmente en lo que a arquitectura se refiere, Microsoft ha liderado este cambio mediante la creación de un marco de trabajo que facilita la integración de diversas tecnologías que permiten cubrir las características antes mencionadas. Sin embargo las empresas desarrolladoras necesitan apoyo para adoptar este nuevo entorno de trabajo, convirtiéndose en escenario propicio para que se conjuguen la experiencia de este tipo de organizaciones con los conocimientos que sobre las nuevas plataformas tecnológicas y metodologías de desarrollo poseen los profesionales del área de Sistemas.

Es en este aspecto donde cobra importancia la realización de proyectos de culminación de carrera bajo la modalidad de Práctica Empresarial, puesto que es un espacio de mutua colaboración entre las empresas que brindan esta oportunidad a los estudiantes para que tengan una experiencia de trabajo real fuera del ámbito académico, permitiéndoles encarar situaciones propias de su quehacer profesional tales como manejo de clientes, labores de mantenimiento y soporte y toma de decisiones.

El presente documento contiene un compendio de los conceptos teóricos y actividades realizadas durante la ejecución de esta práctica empresarial, además de presentar algunas conclusiones y recomendaciones para la culminación exitosa de los proyectos en desarrollo y los futuros, obtenidas de este período que comprendió seis meses completos de trabajo y aprendizaje para su autora.

1. PRESENTACIÓN

En el presente capitulo se describen aspectos generales de la organización en la que se desarrolló la practica empresarial, tales como la razón social, reseña histórica y campo de acción, así como los principales lineamientos de su sistema de gestión de calidad, además se presenta una descripción de los objetivos y alcances del proyecto, a la vez que cuantifica su impacto y viabilidad

1.1 DESCRIPCIÓN DE LA EMPRESA

Esta sección presenta la información básica de la empresa en la que se desarrolló la práctica con el fin de ilustrar el entorno de trabajo de la misma. Se resalta el hecho de que esta empresa fue una de las primeras del nororiente colombiano en obtener el ***Certificado de Gestión de la Calidad*** otorgado por el ICONTEC el cual se basa en el conjunto de normas ISO 9001:2000, en el año 2002 y re-certificándose una vez más en 2006.

1.1.1 Nombre.

- **ISL INGENIEROS DE SISTEMAS LTDA.**
- **Tipo de Organización:** Sociedad Limitada.
- **Fecha de Fundación:** 3 de marzo de 1992.
- **Domicilio:** Calle 35 No. 17 –77 Of. 401 Edificio. Bancoquia.
- **Ciudad:** Bucaramanga

1.1.2 Misión. “Ofrecer servicios informáticos oportunos y de calidad, mediante el análisis, la creatividad, la innovación y la optimización de métodos y herramientas que contribuyan al crecimiento y desarrollo de nuestros clientes.”

1.1.3 Visión. “Ser reconocidos a nivel regional como la organización de base tecnológica líder en innovación y conocimiento de servicios informáticos integrales, con productos ampliamente posicionados y una sólida estructura organizacional, garantizando el desarrollo personal y profesional del recurso humano y estando a la vanguardia en el monitoreo de los cambios y transformaciones que se presenten para dar respuesta y valor agregado a nuestros clientes”

1.1.4 Valores.

- Trabajar en equipo para lograr los objetivos que se propongan en un ambiente de cordialidad, confianza y respeto mutuo.

- Facilitar la aplicación de las capacidades de cada individuo para afrontar los retos que se presenten en cumplimiento de sus objetivos.

- Buscar soluciones creativas e innovadoras para el cumplimiento de los objetivos planteados.

- Dar libertad para expresar sugerencias y reclamos. La administración de ISL evalúa y responde en forma justa.

- Promover el desarrollo personal permanente en forma integral, tanto profesional como humana.

- Proveer servicios oportunos de la más alta calidad que sean consistentes con las necesidades y preferencias de los clientes, logrando la satisfacción de sus necesidades.

1.1.5 Descripción y alcances del sistema de gestión de calidad.

- ***ISL Ingenieros de Sistemas Ltda.*** Se encuentra certificada desde hace cuatro años bajo la norma **ISO 9001:2000**, que certifica la calidad de sus procesos, recientemente le fue renovada su certificación por los próximos tres años, y le fue ampliado el alcance, lo cual refleja el posicionamiento y liderazgo de esta organización frente a otras empresas de su genero.

- **Alcance de la certificación.** "Diseño, desarrollo, implantación y mantenimiento de software para la gestión administrativa y del negocio. Prestación de servicios de: gerencia e interventoría de proyectos administrativos y técnicos de IT, administración de redes, soporte a usuarios, administración y soporte a infraestructura informática, suministro de hardware y software."

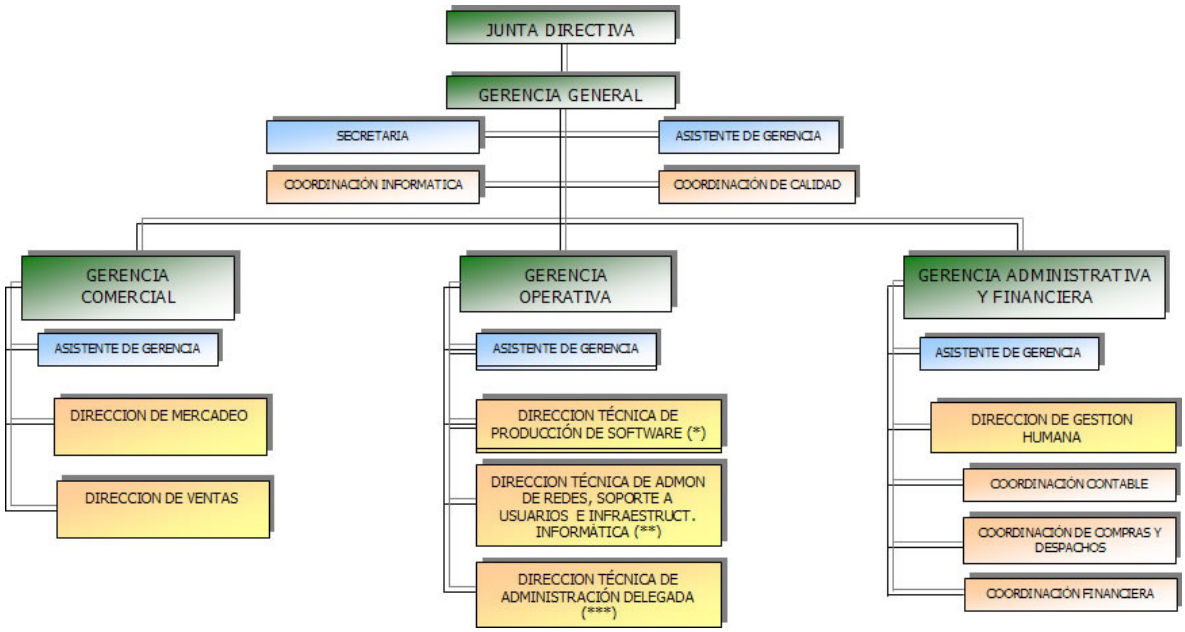
- **Política de calidad.** "Lograr la satisfacción de las necesidades informáticas del cliente identificando oportunidades que permitan mejorar la calidad de nuestros servicios, procesos y sistema de gestión en general."

- **Objetivos de calidad.**

  - Alcanzar un nivel del 80% de satisfacción del cliente
  - Disminuir en un 10% el número de Quejas y reclamos
  - Mejorar la eficiencia del sistema de gestión de calidad en un 5%
  - Alcanzar un nivel cumplimiento del 90% en el tiempo de producción de software

1.1.6 Estructura Organizacional

Figura 1. Organigrama ISL Ingenieros de Sistemas Ltda.



1.1.7 Responsabilidades a cargo.

- Brindar asesoría y soporte técnico a los usuarios de los equipos de micro computación para la resolución de dudas y problemas relacionados con la operación de los equipos, aplicaciones y programas instalados.

- Servir de soporte básico a los sistemas de información, entendiéndose por esto la solución a problemas de impresión, manipulación de archivos, estrategias de respaldo y recuperación, incorrecta instalación y demás actividades directamente ligadas con la informática.

- Instalar, configurar y, de ser necesario, reconfigurar equipos de computo, impresoras, graficadores, computadores, programas, unidades de equipo, etc.

- Diagnosticar e identificar errores de mal funcionamiento de software y hardware instalados en la infraestructura informática.

- Identificar tendencias y tomar las acciones conducentes a remediar globalmente aquellos problemas que repetitivamente son reportados y/o requeridos por su trabajo.

- Brindar apoyo en las labores relacionadas con instalación y traslado de equipos y periféricos.

- Realizar modificaciones menores a los archivos elaborados con las aplicaciones de software básico y realización de nuevos desarrollos.

- Coordinar las labores de mantenimiento correctivo y preventivo de los microcomputadores y sus periféricos.

HELP DESK.

- Actuar como punto de contacto para la recepción y registro de las solicitudes o problemas reportados por los usuarios.
- Diagnosticar las solicitudes, brindando asesoría y soporte técnico a aquellas que pueda resolver.
- Registrar toda la información necesaria para el seguimiento y optimización del servicio.

Calidad.

- Controlar los documentos del sistema de calidad que estén bajo su responsabilidad participando activamente en la revisión de los mismos.

- Generar y ejercer un control sobre los registros del sistema de gestión de la calidad que estén contemplados dentro de sus procesos y responsabilidades, asegurando su identificación, almacenamiento y protección.

- Comunicar y reportar cualquier eventualidad o situación que potencialmente pueda generar una no conformidad, siguiendo el procedimiento de acciones preventivas.

- Aplicar las acciones necesarias para corregir las no conformidades que se puedan presentar, generando las respectivas acciones correctivas y planes de mejora que eviten posteriores ocurrencias. [1]

1.2 DESCRIPCIÓN DEL PROYECTO

1.2.1 Situación Actual. La empresa hasta el momento se ha dedicado a desarrollar tanto aplicaciones de oficina (contabilidad, cartera, inventario y facturación) de carácter genérico como aplicaciones a la medida para diferentes clientes, estas aplicaciones están construidas en diferentes herramientas como son: CLIPPER (Las más antiguas), Visual Fox Pro, Visual Basic 6.0, entre otras. La incursión de las nuevas tecnologías y creciente interés de las empresas en globalizar sus productos y servicios ha llevado a que ISL decidiera adoptar estas nuevas tecnologías, sin embargo sus procesos deben adaptarse para que se

[1] Tomado del documento **GAGR01 MANUAL DE RESPONSABILIDADES** (Documento Interno)

acoplen a los nuevos paradigmas que trae consigo herramientas de desarrollo como Visual Studio .Net 2005; así mismo la facilidad de las mismas para acoplarse a diferentes motores de base de datos como SQL Server, ORACLE y MySQL.

1.2.2 Objetivos. ISL como empresa certificada define estructuras y procedimientos para garantizar la calidad de sus procesos, es por esto que conocerlos y aplicarlos debe ser una de las primeras actividades que se debe contemplar. Además para brindar apoyo a los proyectos que la empresa desarrolla es necesario conocer sus requerimientos y los documentos que los soportan.

Uno de los principales objetivos de la práctica es el apoyo a la transición de la empresa de las herramientas de desarrollo convencionales a las herramientas orientadas a la Web como lo es Visual Studio .NET que permiten que el acceso a una base de datos desde el Web, por ejemplo, sea un mero trámite, lo más importante es decidir entre el conjunto de posibilidades la correcta dadas la experiencia y recursos con que cuenta la empresa. Participar en las diferentes etapas de desarrollo software esta por ende implícito en esta transición.

- **General.** Apoyar el diseño y construcción de aplicaciones desarrolladas con herramientas Visual Studio .NET y diferentes plataformas de base de datos en proyectos de software de la empresa ISL Ingenieros de Sistemas Ltda.

- **Específicos.**

  - Apoyar la determinación de requerimientos finales de los módulos de Facturación y Cartera de la aplicación, haciendo uso de, entre otros: requerimientos iniciales del proyecto, documentos de la etapa de análisis y la interacción con el cliente.

- Participar en las etapas de diseño y construcción de los módulos de Facturación y Cartera de la aplicación STigia - Productos Tradicionales.

- Ejecutar y documentar las pruebas realizadas al Software de Facturación, Cartera, Inventarios y Comisiones Nomina de Productos Tradicionales – STigia para un Parque Cementerio.

- Realizar la documentación técnica y de usuario de la aplicación.

- Participar en la capacitación del personal de la empresa en el uso de las herramientas de desarrollo Visual Studio .NET para el software de Facturación, Cartera, Inventarios y Comisiones Nomina de Productos Tradicionales para un Parque Cementerio.

1.2.3 Justificación. ISL Ingenieros de Sistemas Ltda. dentro de su línea de servicios, contempla el "Diseño, desarrollo, implantación y mantenimiento de software para la gestión administrativa y del negocio" [2] la empresa cuenta con más de 14 años de experiencia a través de los cuales se ha caracterizado por brindarle a sus clientes aplicaciones a la medida con herramientas de última tecnología para lo cual debe mantenerse actualizados y a la vanguardia en lo que a técnicas de desarrollo se refiere.

Con la introducción de Internet y en concreto de las tecnologías Web, se nos presentan un sin fin de posibilidades en cuanto al acceso a la información desde casi cualquier lugar del planeta. Esto se convierte en un reto para los desarrolladores, ya que las empresas demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizar el Web como medio de globalización de sus productos y servicios.

[2] Tomado del alcance de la certificación ISO 9001:2000 de la empresa.

En vista de lo anterior se evidencia la necesidad de nuestro aporte que, como futuros profesionales formados en metodologías y herramientas modernas, podemos hacer a los nuevos desarrollos al interior de la empresa para el mejoramiento de los procesos y la actualización de las metodologías en virtud de lo que la política de calidad de la organización plantea: “Lograr la satisfacción de las necesidades informáticas del cliente identificando oportunidades que permitan mejorar la calidad de nuestros servicios, procesos y sistema de gestión en general”[3], estas oportunidades se presentan en la realización de la presente práctica.

- **Impacto y Viabilidad.** Este proyecto contribuirá al fin que la empresa desea lograr a través de la elaboración y documentación de componentes software que contribuyan crecimiento y desarrollo de sus clientes.

Para esto la empresa posee la experiencia y herramientas necesarias: Visual Studio .NET, SQL Server 2005, múltiples productos ampliamente comercializados y un equipo de desarrollo, conformado por personal de la empresa y un estudiante en práctica empresarial, que realizará tareas que comprenden desde el diseño del modelo de datos y los componentes software, hasta la realización de pruebas en ambiente real e implementación y capacitación del cliente en la aplicación desarrollada. La selección de las herramientas adecuadas para el desarrollo, la organización en grupos de trabajo y la elaboración e integración de módulos que compartan datos y funcionalidades, implicará la capacitación y orientación necesaria al estudiante en práctica en cuanto a la metodología que utilizará y la forma de trabajo dentro de la empresa.

1.2.4 Plan de Trabajo. Se determino dividir la práctica en tres fases: Planeación, Ejecución y Terminación.

[3] Tomado del manual de calidad de ISL Ingenieros de Sistemas Ltda.

- **Planeación.** Corresponde a una comprensión inicial de la organización y sus procesos así como la capacitación de las metodologías y herramientas a utilizar durante la etapa de ejecución. Las actividades a realizadas durantes esta etapa fueron:

  - Inducción a la organización.
  - Revisión de documentos y procesos internos.
  - Capacitación en sistema de gestión de calidad.
  - Revisión y ajuste de plan de trabajo.
  - Estudio de metodologías y herramientas de desarrollo.
  - Elaboración y entrega de informe de avance.

- **Ejecución.** En esta etapa la meta es poner en práctica el conocimiento adquirido en la documentación y apropiación de las herramientas de desarrollo efectuadas en la etapa anterior, así como entrar en forma a colaborar en los procesos de desarrollo de la empresa.

  Las actividades desarrolladas en esta etapa fueron:

  - Instalación y conocimiento de la herramienta de desarrollo Visual Studio .NET.

  - Instalación y configuración del servidor de BD.

  - Apoyo a la elaboración y revisión de informes de análisis y diseño de la aplicación STigia.

  - Apoyo al diseño y construcción de la base de datos de la aplicación.

  - Elaboración y entrega del primer informe de avance etapa de ejecución

- Construcción de los módulos de facturación y cartera.

- Elaboración y entrega del segundo informe de avance de la etapa de ejecución

- Planeación, ejecución y documentación de pruebas de la aplicación.

- Apoyo a la elaboración de la documentación técnica y de usuarios.

- Elaboración y entrega del tercer informe de avance de la etapa de ejecución

- **Terminación.** En esta fase se empieza a delegar las responsabilidades que durante la práctica se hayan tenido a cargo así como la elaboración y entrega del informe final.

- Transferencia de responsabilidades a cargo.
- Elaboración y entrega de informe final.

2. MARCO METODOLÓGICO

El presente capítulo recopila las bases teóricas y metodológicas que orientaron el trabajo realizado durante la duración de la práctica.

Esta primera parte describe las principales metodologías a ser tenidas en cuenta para el desarrollo de proyectos software.

2.1 FUNDAMENTOS

2.1.1 Metodologías de desarrollo software. Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. En el desarrollo de proyectos informáticos es necesario establecer una metodología y un plan de trabajo adecuados que permitan obtener los resultados enmarcados en un cronograma y un presupuesto previamente programados. En este punto radica la importancia de definir y aplicar una de ellas en la construcción de una aplicación Web o cualquier otra aplicación basada en el computador, ya que actúan como plano de apoyo y punto de referencia para hacer más efectivo el trabajo en equipo, cada vez más común dado que las aplicaciones son cada vez de mayor envergadura y la aparición de las tecnologías Web demandan además el empleo de una mayor cantidad de métodos y herramientas de seguridad.

Sin embargo la elección de la metodología adecuada debe tener en cuenta los siguientes factores:

- El tipo de sistema a desarrollar (de Gestión, Científico, de Tiempo real, a la medida ó genérico).
- Alcance y envergadura de la aplicación.
- Forma de trabajo del equipo de desarrollo.
- El tipo de cliente o usuarios.
- La volatilidad de los requisitos.

Cualquiera sea el modelo de ciclo de vida seleccionado, debe cumplir con los siguientes requisitos:

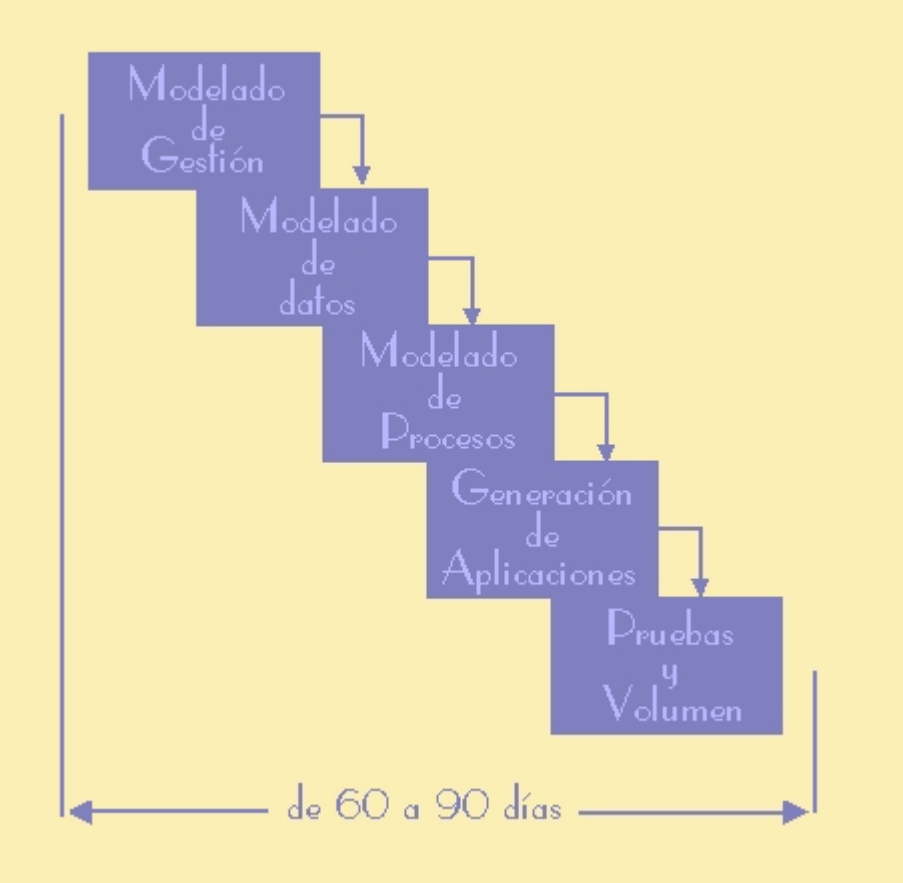
- Detallar el procedimiento a seguir y el orden de las etapas de desarrollo.
- Determinar los criterios que señalen la culminación de una etapa.

Actualmente, existen diferentes tipos de metodologías o procesos que buscan satisfacer el desarrollo de proyectos informáticos, llamados también modelos de ciclo de vida del producto software, entre los que se encuentran: *Modelos secuénciales: cascada pura, desarrollo rápido de aplicaciones (DRA); y Modelos evolutivos: prototipado evolutivo, el modelo en espiral, programación Extrema (XP: Xtreme Programming y Proceso Unificado (RUP)*; los cuales poseen características especificas que buscan adaptarse a distintos proyectos. A continuación se describen brevemente:

- **DRA: Desarrollo rápido de aplicaciones.** El Desarrollo Rápido de Aplicaciones (DRA) (Rapid Application Development RAD) es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. DRA es una adaptación a "Alta velocidad" en el

que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un "sistema completamente funcional" dentro de periodos cortos de tiempo.

Figura 2. Fases o Etapas del DRA.



Cuando se utiliza principalmente para aplicaciones de sistemas de información, el enfoque DRA comprende las siguientes fases:

- Modelado de gestión: Diseñar la interrelación entre las funciones y la información generada por ellas para apoyar el proceso de gestión.

- Modelado de datos: Descripción detallada de los objetos, atributos y relaciones de la información definida en el modelado de gestión.

- Modelado de proceso: los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.

- Generación de aplicaciones: El DRA se enfoca en la construcción de componentes reutilizables y la reutilización de módulos de otros programas.

- Pruebas de entrega: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfases a fondo.

Si el proceso de gestión puede modularse entonces la aplicación es candidato a utilizar la metodología DRA. Así pues cada una de las funciones puede ser afrontada por un equipo diferente y ser integradas en un solo conjunto.

Ventajas.

- DRA enfatiza el desarrollo de componentes de programas reutilizables. La reutilización es la piedra angular de las tecnologías de objetos, y se encuentra en el modelo de proceso de ensamblaje.

- El tiempo de desarrollo que se puede obtener con esta metodología va de los 60 a los 90 días convirtiéndose en una alternativa adecuada para pequeños proyectos al interior de las organizaciones que requieran aplicaciones a la medida.

Desventajas.

- Para aplicaciones a gran escala aunque por módulos el enfoque DRA tiene el inconveniente de necesitar tantos grupos de desarrollo como módulos se definan.

- Esta metodología demanda ingenieros y clientes comprometidos con el ritmo de trabajo de esta metodología, de lo contrario el proyecto fracasará.

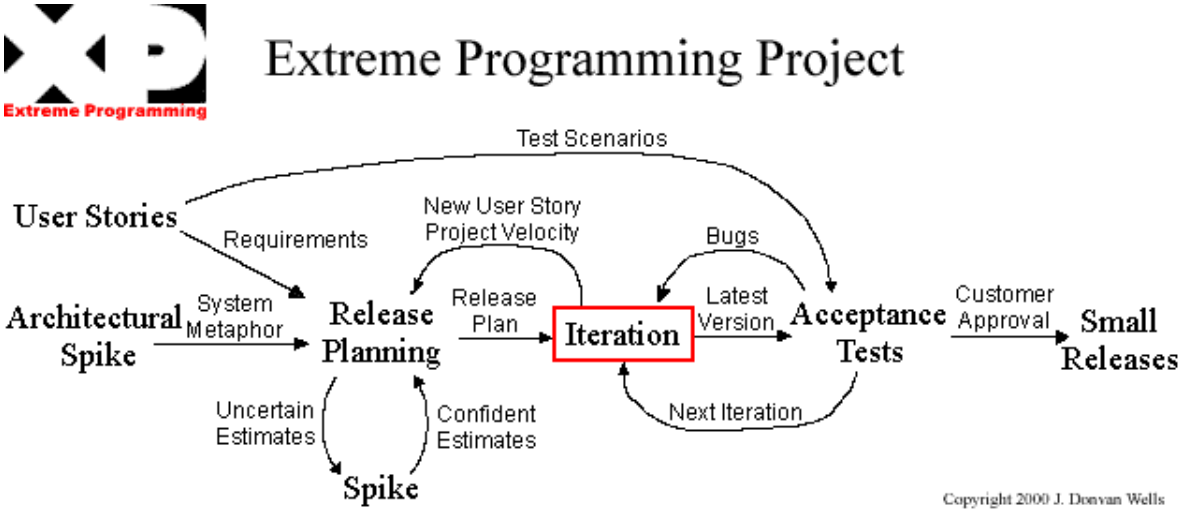
- No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modulizar adecuadamente. La construcción de los componentes necesarios para DRA será problemático.

- DRA no es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el nuevo software requiere un alto grado de interoperabilidad con programas de computadora ya existentes.

- **XP: Extreme Programming.** Metodología de desarrollo software formulada por Kent Beck en 1999, hace parte de las llamadas “Metodologías Ágiles” se caracteriza por enfocarse a la adaptabilidad (aceptar los cambios) y en ser altamente utilizada en proyectos a corto plazo con poca disponibilidad de equipos y principalmente cuyo cronograma se encuentra atrasado.

Una de las características más notorias de esta metodología es contar con el cliente como parte del grupo de desarrollo.

Figura 3. Fases de la Metodología XP4



El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

- Fase I. Exploración: en esta fase los clientes ilustran a grandes rasgos las funciones y procesos del usuario que son relevantes para la primera entrega del producto, paralelamente el equipo de desarrollo se familiariza con las tecnologías y herramientas a utilizar y se construye un primer prototipo. Esta fase puede llevar desde unas semanas hasta un par de meses.

- Fase II. Planificación de la Entrega: en esta fase el cliente prioriza las diferentes funciones y procesos esbozados en la etapa anterior de tal manera que los programadores realizan una estimación del esfuerzo y tiempo necesario. Se toman acuerdos sobre el contenido y alcance. Conjuntamente con el cliente se establece el cronograma de la primera entrega.

[4] http://oness.sourceforge.net/proyecto/html/ch05s02.html

- Fase III. Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

- Fase IV. Producción: Esta fase se requieren pruebas adicionales, revisiones de rendimiento y reducción del tiempo de las iteraciones de tres a una semana. Se debe reevaluar el contenido del proyecto para incorporar nuevos elementos obtenidos de la refinación de esta fase. Se documentan las sugerencias para su posterior implementación (Por ejemplo en la fase de mantenimiento).

- Fase V. Mantenimiento: Se realizan tareas de soporte al cliente mientras la primera versión se encuentra en producción manteniendo el sistema en funcionamiento mientras se desarrollan nuevas iteraciones. Deben incluirse nuevos elementos al equipo de trabajo en esta etapa.

- Fase VI. Muerte del Proyecto: Se requiere que las necesidades y expectativas del cliente sean satisfechas, ampliando estas necesidades a los ámbitos de rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios a la arquitectura. La muerte del proyecto también se presenta cuando la aplicación no presenta los resultados esperados por el cliente o cuando no se cuenta con presupuesto para mantenerla.

Ventajas.

- Se cuenta con el cliente en cada paso del desarrollo permitiendo así que su cumplan sus expectativas a la par de que se puede realimentar de las ideas del equipo y presentar alternativas que repercutan en pos de la mejora del sistema.

- La planificación es iterativa permitiendo que la interacción con el cliente permita adaptar el proyecto a los cambios que van surgiendo dentro de la organización y que ineludiblemente repercuten en el diseño inicial.

- Una de las características más revolucionarias y controversiales es la de el trabajo en pares, el cual consiste en que dos programadores trabajen en un mismo equipo lo cual, según esta metodología, reduce los tiempos de ocio, reduce las omisiones y disminuye la negligencia.

- Otra de las características de resaltar es el concepto de refactorización, Este es básicamente el proceso de mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez, disminuyendo notoriamente los costos desorbitados del mantenimiento, modificación y ampliación de aplicaciones ya existentes. La refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar nueva funcionalidad.

Desventajas.

- Esta metodología funciona en su máxima expresión en proyectos que demanden un grupo de no más de 20 personas, para proyectos de mayor envergadura las líneas de comunicación se vuelven demasiado complejas.

- Trabaja mejor con interacción cara a cara. Difícil con equipos grandes.

- **Modelo en Cascada.** Presentado por primera vez en 1970 por Royce. Es el paradigma más antiguo y ampliamente difundido en la Ingeniería del Software. Desde su aparición muchos equipos de desarrollo han seguido este modelo el cual se define como una secuencia ordenada de transiciones, de fase en fase, que evoluciona en forma lineal, exige un enfoque sistemático y secuencial del desarrollo del producto software y contempla las siguientes fases:

  - **Ingeniería y Análisis del Sistema Global:** Al finalizar esta etapa se han definido y documentado los requisitos globales del sistema global, el plan de tiempos del sistema, el alcance, el estudio de viabilidad y la arquitectura del sistema.

  - **Análisis de Requisitos de Software:** Se redacta el documento formal de requerimientos (Flujo de información, interfaces, funcionalidad y rendimiento del sistema, restricciones de información y seguridad); el plan de pruebas y la aceptación de los requerimientos finales por parte del cliente.

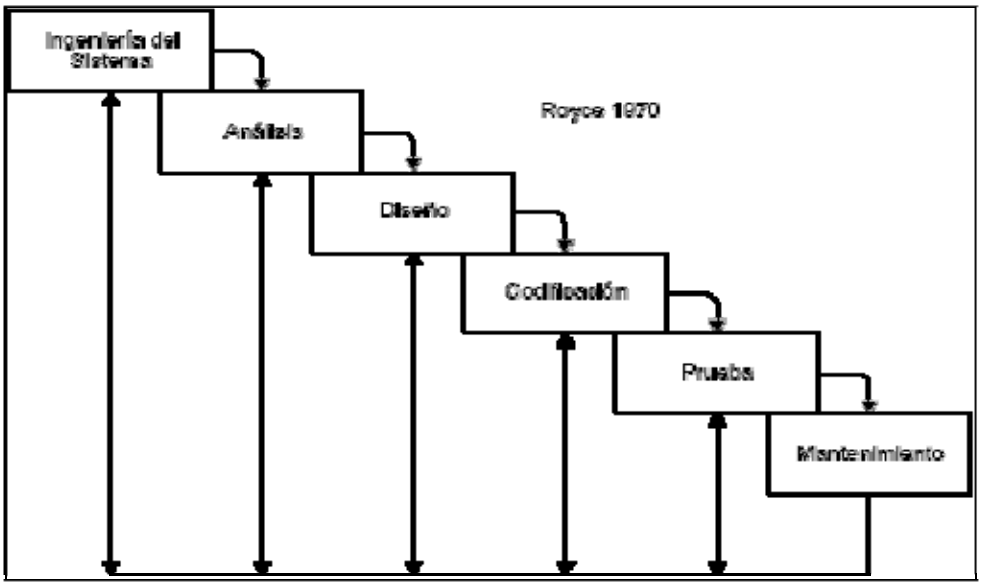
  - **Diseño:** Comprende la documentación detallada de la arquitectura, de los flujos de información, del modelo de datos, interfaces, plan de pruebas y el manual de usuario básico.

  - **Codificación:** Esta fase tiene como productos el código fuente, las librerías necesarias, datos para casos de pruebas, documentación relacionada con las herramientas utilizadas, plan de pruebas. Debe conservarse copia magnética de respaldo para todos estos productos ya que conforman la columna vertebral del sistema.

  - **Prueba:** Ejecutar el plan de pruebas y documentar los resultados.

- **Mantenimiento:** Se establece al finalizar la fase de Instalación o Implantación, comprende la documentación relacionada con las tareas de operación y mantenimiento, plan de formación, recomendaciones de mantenimiento y Plan de Retiro.

Figura 4. Modelo presentado en 1970 por Royce para el modelo de cascada



Ventajas.

- Una fase no comienza hasta que no hayan terminado las anteriores. Esto supone que no se pasa a la siguiente etapa hasta verificar que los resultados sean acordes a lo planificado.

- Esta metodología fue la base para las subsecuentes y fue aplicada en forma exitosa por grandes compañías por casi dos décadas.

- Ayuda a prevenir que sobrepase fechas de entrega y controla los sobrecostos.

Desventajas.

- La principal desventaja es la dificultad de tener en cuenta los cambios cuando el proceso ya está en marcha.

- Inflexibilidad al dividir el proyecto en estas etapas.

- Es difícil responder a los cambios en los requisitos del cliente.

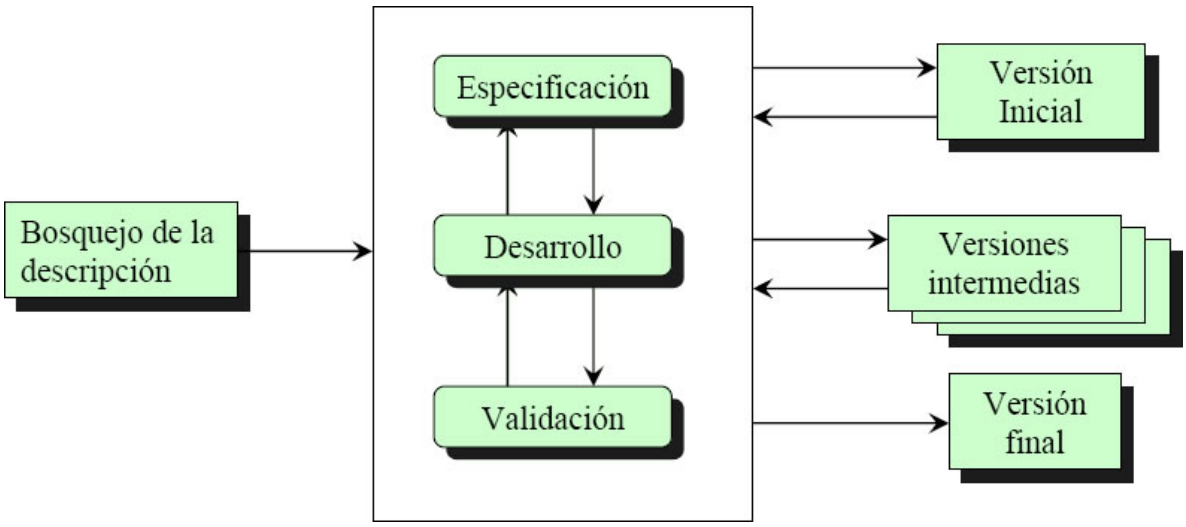
- Es restrictivo y rígido.

- **Desarrollo Evolutivo.** El modelo de desarrollo evolutivo (algunas veces denominado como prototipado evolutivo) construye una serie de grandes versiones sucesivas del producto. Este modelo asume que los requerimientos no son completamente conocidos al inicio del proyecto.

El objetivo es desarrollar una implementación inicial e ir refinándola hasta conseguir el sistema adecuado incluyendo en ésta solo los requerimientos que se encuentren bien comprendidos. Esta primera implementación se les entrega a los usuarios, los cuales lo usan y proveen retroalimentación al equipo de trabajo. El sistema evoluciona según las nuevas propuestas del cliente.
Este tipo de metodología es adecuado para:

- Para sistemas interactivos pequeños o de tamaño mediano
- Para partes de sistemas grandes (Ej. el interfaz de usuario)
- Para sistemas con vida corta

Figura 5. Desarrollo Evolutivo.



Las fases de esta metodología son:

- Análisis de requisitos del sistema.
- Análisis de requisitos del software.
- Diseño, desarrollo e implementación del prototipo
- Prueba del prototipo.
- Refinamiento iterativo del prototipo.
- Refinamiento de las especificaciones del prototipo.
- Diseño e implementación del sistema final.
- Explotación (u operación) y mantenimiento.

Ventajas

- La especificación se desarrolla de forma creciente.

- El diseño y construcción del prototipo permite evaluar hasta que punto se han comprendido realmente las necesidades y requisitos del cliente.

- Si al evaluar el primer prototipo se encuentra que no se escogieron bien las herramientas o arquitectura del proyecto se puede tomar correcciones tempranamente.

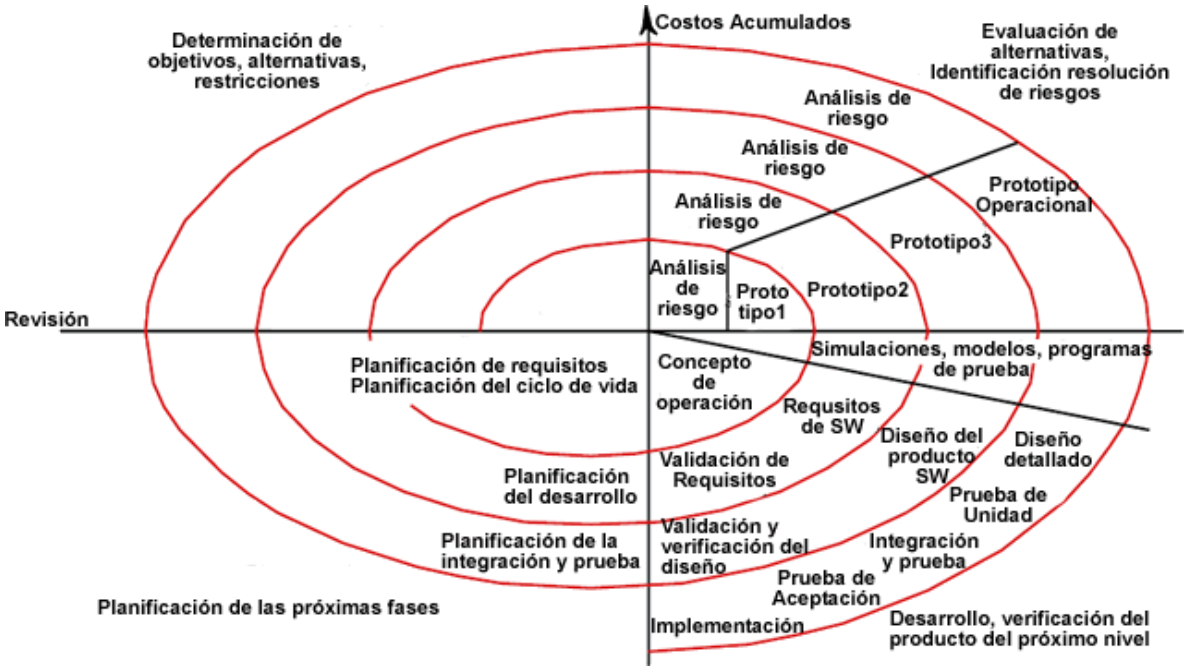
Desventajas.

- Hay que documentar cada versión del sistema
- Los sistemas tienen una estructura deficiente
- Se requieren herramientas y técnicas especiales (p.e. conocimientos en lenguajes para el prototipado rápido)

- **Modelo en Espiral.** Presentado por Böem en 1986, esta metodología tiene por objetivo reducir los riesgos mediante el análisis y planificación en cada etapa de desarrollo dirigida a su disminución.

Representada por ciclos de desarrollo iterativo e incremental de forma espiral, su avance angular ilustra el progreso del desarrollo, a su vez el desplazamiento radial desde el centro hacia fuera define el incremento de los costos acumulados del desarrollo.

Figura 6. El modelo Espiral



- **Fases.**

▪ Determinación de Objetivos y Alternativas: Involucra la determinación de objetivos del proyecto, alternativas y restricciones, y, en etapas posteriores, la valoración de los resultados de las tareas de ingeniería previas.

▪ Análisis de riesgo: Estas actividades permiten gestionar los riesgos asociados al proceso de desarrollo. Se materializan en el análisis de alternativas e identificación y resolución de los riesgos que puedan hacer fracasar el proyecto o sobrepasar el presupuesto o plazo fijado. Producido el análisis de riesgo, se toma la decisión de continuar o no con el desarrollo.

- Ingeniería: Abarca las actividades inherentes al desarrollo del producto y comprende la construcción de los prototipos de nivel cada vez mas refinados, a medida que se produce la evolución del sistema, hasta llegar al producto final.

- Planificación: Involucra todo lo atinente a las tareas de planificación y estimaciones del proyecto en las distintas etapas. [5]

Ventajas:

- Esta metodología puede ser utilizada en cualquier tipo de proyecto de desarrollo, en especial en aquellos donde no se conocen todos los requerimientos del sistema.

- Si bien los costos son altos en la etapa de desarrollo, la implementación de esta metodología disminuye los riesgos a mediano y largo plazo.

- Permite un alto grado de control.

- Reducción continúa del nivel de riesgo en el proyecto.

- Permite visualizar progreso con mayor facilidad.

- Se pueden incorporar modelos como el incremental y el prototipado evolutivo en las iteraciones.

- Este modelo es ideal para manejar proyectos que requieran la incorporación de nuevas tecnologías, o para desarrollar productos completamente nuevos o con un nivel alto de inestabilidad de los requerimientos.

[5] **RANCÁN,** Claudio Jorge. TRABAJO FINAL ESPECIALIDAD EN CONTROL Y GESTIÓN DE SOFTWARE, Julio 2003

Desventajas.

- Requiere mayor planificación

- Se necesita mayor grado de seguimiento y verificación.

- Es un modelo complicado.

- Puede ser difícil definir hitos[6] y objetivos de comprobación que indiquen si esta preparado para pasar al siguiente nivel.

- Debe tenerse en cuenta que el cliente no siempre esta inclinado a ser parte activa del desarrollo.

- **Proceso Unificado (Rational Process Unified).** El proceso unificado está basado en componentes y se sostiene sobre tres ideas básicas: casos de uso, arquitectura y desarrollo iterativo e incremental. Para hacer que estas ideas funcionen, se necesita un proceso, que tenga en cuenta ciclos, fases, flujos de trabajo, gestión del riesgo, control de calidad, gestión de proyecto y control de la configuración.

Un desarrollo iterativo, guiado por los casos de uso y centrado en la arquitectura, construye un software mediante pequeños incrementos, y añade cada incremento a la acumulación previa de incrementos de tal forma que siempre se tenga una construcción ejecutable. La arquitectura proporciona la estructura sobre la cual guiar las iteraciones mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración. De esta manera el proceso reduce el riesgo de

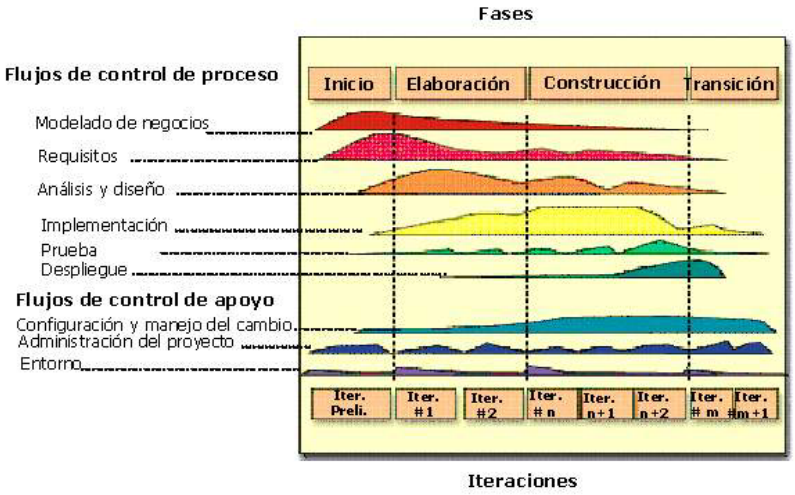
[6] Hito es un punto de referencia que marca acontecimientos importantes en un proyecto y que se utiliza para controlar el progreso del mismo.

grandes retrasos en la entrega de un producto, se fijan metas más inmediatas por lo cual se puede controlar mejor el avance del proyecto.

El Proceso Unificado divide el proceso de desarrollo en ciclos, donde se obtiene una nueva versión del producto al final de cada ciclo. Cada ciclo se divide en cuatro Fases: Inicio, Elaboración, Construcción, y Transición. Cada una de estas fases concluye con un hito bien definido donde deben tomarse decisiones respecto al proyecto como la reestructuración del cronograma de trabajo. Cada una de estas fases se divide a su vez en iteraciones.

Cada iteración sigue la estructura de un pequeño ciclo de vida en cascada, pasando a través de los cinco flujos de trabajo fundamentales: requisitos, análisis, diseño, implementación y prueba. En la iteración también incluye la planificación que precede a los flujos de trabajo y la evaluación que va detrás de ellos.

Figura 7. Fases e Iteraciones de la Metodología RUP



- **Ventajas.**

- El proceso unificado es un marco de trabajo genérico.
- Busca eliminar los riesgos críticos primero.
- Reduce el costo de riesgos.
- Presenta signos visibles de progreso rápidamente.
- Proporciona funcionalidad al cliente antes de entregar todo el proyecto.
- Se entregan prestaciones importantes al inicio
- Proporciona suficiente control de gestión

- **Desventajas.**

- Requiere mayor planificación
- Requiere mayor control de gestión.
- Se necesita definir su arquitectura antes de dar inicio a las iteraciones de construcción.

2.1.2 Metodología Utilizada: Prototipado Evolutivo. Al no tener un conocimiento total o detallado de las herramientas y de los alcances que estas puedan tener, se deben controlar los posibles riesgos de la aplicación, buscando una solución adecuada para cumplir con los objetivos propuestos. Una alternativa para manejar este tipo de situaciones, es el de construir sólo una parte del sistema, reservando otros aspectos para niveles posteriores; y esto lo ofrece el prototipado evolutivo. Por estas y otras razones ya expuestas y, en conformidad con los lineamientos del sistema de gestión de calidad específicamente en lo que corresponde a la realización del producto, se ha adoptado la metodología de Prototipado Evolutivo para el desarrollo de los proyectos. A continuación se hace una descripción de los aspectos relevantes en esta metodología.

- **Prototipado Evolutivo.** El uso de prototipos se centra en la idea de ayudar a comprender los requisitos que plantea el usuario, sobre todo si este no tiene una idea muy acabada de lo que desea. También pueden utilizarse cuando el ingeniero de software tiene dudas acerca de la viabilidad de la solución pensada. Esta versión temprana de lo que será el producto, con una funcionalidad reducida, en principio, podrá incrementarse paulatinamente a través de refinamientos sucesivos de las especificaciones del sistema, evolucionando hasta llegar al sistema final.

Al usar prototipos, las etapas del ciclo de vida clásico quedan modificadas de la siguiente manera:

- Análisis de requisitos del sistema.
- Análisis de requisitos del software.
- Diseño, desarrollo e implementación del prototipo
- Prueba del prototipo.
- Refinamiento iterativo del prototipo.
- Refinamiento de las especificaciones del prototipo.
- Diseño e implementación del sistema final.
- Explotación (u operación) y mantenimiento.

Si bien el modelo de prototipos evolutivos, fácilmente modificables y ampliables es muy usado, en muchos casos pueden usarse prototipos descartables para esclarecer aquellos aspectos del sistema que no se comprenden bien.

- **Ciclo de vida del Prototipado Evolutivo.** En el ciclo de vida de prototipo incremental se definen las siguientes etapas:

1. Factibilidad (FAC)
2. Definición de requisitos del sistema (RES)

3. Especificación de los requisitos del prototipo (REP)
4. Diseño del prototipo (DPR)
5. Diseño detallado el prototipo (DDP)
6. Desarrollo del prototipo (codificación) (DEP)
7. Implementación y prueba del prototipo (IPP)
8. Refinamiento iterativo de las especificaciones del prototipo (aumentando el objetivo y/o el alcance).

Luego, se puede volver a la etapa 2 o continuar si se logró el objetivo y alcance deseados. (RIT)

9. Diseño del sistema final (DSF)
10. Implementación del sistema final (ISF)
11. Operación y mantenimiento (OPM)
12. Retiro (si corresponde) (RET)

A continuación se describen cada una de las etapas del ciclo de vida elegido:

1. Factibilidad (FAC): En esta etapa se define el producto software y se determina su factibilidad en el ciclo de vida desde la perspectiva de la relación costo – beneficio, como así las ventajas y desventajas respecto de otros productos.

2. Requisitos del sistema (RES): En esta etapa se deben definir las funcionalidades requeridas para el desarrollo del sistema (o programa), las interfaces y el tipo de diseño.

3. Especificación de requisitos del prototipo (REP): Consiste en especificar las funciones requeridas, las interfaces y el rendimiento para el prototipo. Aquí se considerarán incrementos en porcentajes de la funcionalidad total del sistema.

4. Diseño del prototipo (DPR): Es poner en ejecución del plan del prototipo, ya que una vez fijadas las restricciones con el usuario, hay que mostrar el mismo funcionando, aunque sean sólo algunas funcionalidades restringidas. Aquí, hay que hacer un análisis de cómo se va a trabajar, qué módulos se van a hacer, con qué lógica y qué funciones se van a usar.

5. Diseño detallado del prototipo (DDP): Esta etapa es una especificación verificada de la estructura de control, la estructura de los datos, las relaciones de interfaces, el tamaño, los algoritmos básicos y las suposiciones de cada componente del programa. En esta etapa no sólo se definen y sino que se documentan los algoritmos que llevarán a cabo la función a realizar por cada uno de los módulos.

El diseño de software, es un proceso que se centra en cuatro atributos distintos del programa: la estructura de datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfase. En este proceso deben traducirse los requisitos a una representación del software que pueda ser establecida de forma que se obtenga la calidad requerida antes de que comience la codificación.

6. Desarrollo del prototipo (codificación) (DEP): Consiste en realizar la codificación o diseño detallado, en forma legible para la máquina.

7. Implementación y prueba del prototipo (IPP): Consiste en lograr un funcionamiento adecuado del producto software en el sistema informático, funcionando operacionalmente, incluyendo objetivos tales como la conversión del programa y datos (si la hubiere), la instalación y el entrenamiento. La prueba debe asegurar que se han probado toas las sentencias del mismo, y que en las funciones externas se han realizado pruebas que aseguren que la entrada definida produce los resultados que se esperan realmente.

8. Refinamiento iterativo de las especificaciones del prototipo (RIT): Es un aumento de la funcionalidad del sistema, para luego volver REP a fin de aumentar la funcionalidad del prototipo o continuar, si se logró el objetivo y alcance deseados.

9. Diseño del sistema final (DSF): Consiste en ajustar las restricciones o condiciones finales e integrar los últimos módulos.

10. Implementación del sistema final (ISF): Es el sistema de informático funcionando operativamente, incluyendo tales objetivos como conversión del programa y datos, (si la hubiere), la instalación y La capacitación del personal.

11. Operación y mantenimiento (OPM): Es la puesta en funcionamiento del sistema informático, objetivo que se repite para cada actualización.

12. Retiro (RET): Es una transición adecuada de las funciones realizadas para el producto y sus sucesores. [7]

[7] **CATALDI**, Zulma. **Metodología de diseño, desarrollo y evaluación de software educativo.** Tesis de Magíster en Informática. (versión resumida) Facultad de Informática. UNLP. 2000. *cataldi-tesisdemagistereninformatica.pdf*

3. MARCO TEÓRICO

3.1 APLICACIONES WEB

La aparición de Internet y el rápido avance de nuevas tecnologías asociadas al Web han facilitado que los sistemas de información se generalicen, permitiendo su uso a cualquier usuario potencial con acceso a la red.

En este contexto encuentra sentido el término de “Aplicación Web”, referido a la nueva familia de desarrollos software especialmente modelados y diseñados para ser ejecutas en la Web.

En los próximos años una de las mayores preocupaciones latentes es el desarrollo de métodos de producción de software que permitan construir aplicaciones Web complejas con comportamiento dinámico, que sean compatibles con los estándares metodológicos más extendidos en la actualidad (especialmente RUP) y que establezcan como diseñar y desarrollar aplicaciones Web basadas en arquitecturas sólidas que permitan integrar aplicaciones existentes, realizar transferencias electrónicas seguras y ofrecer funcionalidad y rendimiento a través de la Web.

El creciente cambio de las aplicaciones tradicionales de escritorio hacia las aplicaciones Web se debe principalmente a dos grandes razones:

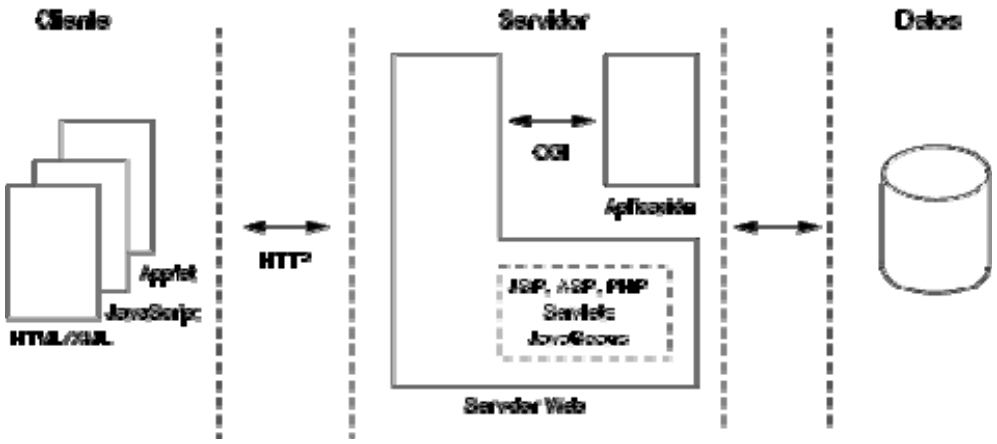
1. La información sea accesible desde cualquier lugar dentro de la organización e incluso desde el exterior.

2. Esta información sea compartida entre todas las partes interesadas, de manera que todas tengan acceso a la información completa (o a aquella parte que les corresponda según su función) en cada momento.

Ya que las aplicaciones Web, por sus características, cubren a la perfección las necesidades mencionadas los sitios Web tradicionales (Los cuales se limitaban a mostrar información) se han convertido en aplicaciones capaces de un mayor nivel de interacción con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar nuevas opciones de diseño que permitan definir la arquitectura óptima que facilite la construcción de los mismos.

3.1.1 Funcionamiento. El usuario interacciona con las aplicaciones web a través del navegador. Como consecuencia de la actividad del usuario, se envían peticiones al servidor, donde se aloja la aplicación y que normalmente hace uso de una base de datos que almacena toda la información relacionada con la misma. El servidor procesa la petición y devuelve la respuesta al navegador que la presenta al usuario. Por tanto, el sistema se distribuye en tres componentes: el navegador, que presenta la interfaz al usuario; la aplicación, que se encarga de realizar las operaciones necesarias según las acciones llevadas a cabo por éste y la base de datos, donde la información relacionada con la aplicación se hace persistente. Esta distribución se conoce como el modelo o arquitectura de tres capas (ó niveles).

Figura 8. Funcionamiento de una aplicación Web

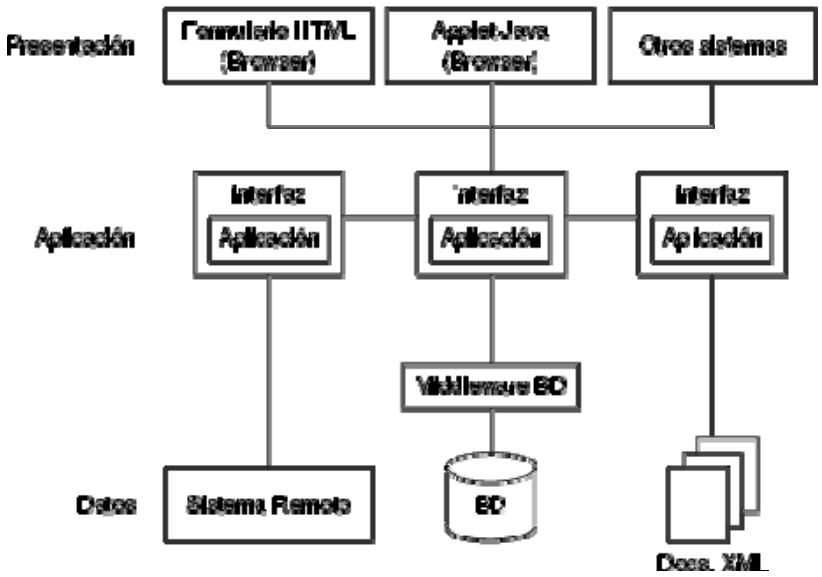


3.1.2 Arquitectura. Este tipo de aplicaciones se cuentan dentro de las llamadas aplicaciones de 3 capas. Los sistemas tradicionales de arquitectura cliente/servidor presenta varios inconvenientes entre los cuales se cuenta el hecho de que todo el peso del procesamiento recae sobre los clientes (aunque generalmente estos suelen ser maquinas menos potentes que los servidores) dejando al servidor la parte menos pesada de gestión de la base de datos. Además esta el problema del mantenimiento y actualización de las aplicaciones, ya que las modificaciones deberán hacerse en todos los equipos cliente.

Para solucionar estos problemas se ha desarrollado el concepto de arquitectura de tres niveles (o modelo de tres capas): interfaz de presentación, lógica de la aplicación y los datos. La capa intermedia es el código que el usuario invoca para recuperar información de la base de datos requerida. La capa de presentación recibe los datos y los prepara para ser mostrados adecuadamente. Esta división entre la capa de presentación y la de la lógica permite un alto grado de flexibilidad e integración con otras aplicaciones ya que se pueden tener múltiples interfaces

sin cambiar la lógica de programación. La tercera capa consiste en los datos que gestiona la aplicación. Estos datos pueden ser cualquier fuente de información como una base de datos o archivos XML.

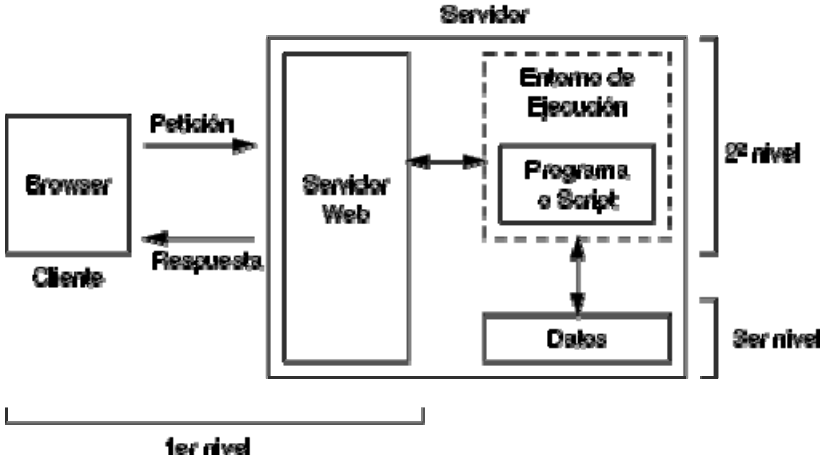
Figura 9. Arquitectura de tres niveles



La arquitectura de las aplicaciones Web suelen presentar un esquema de tres niveles (véase la figura). El primer nivel consiste en la capa de presentación el cual incluye tanto el navegador como el servidor Web el cual es el responsable de darle el formato adecuado a los datos. El segundo nivel esta compuesto por algún tipo de programa o *Script*. Finalmente, el tercer nivel proporciona al segundo los datos necesarios para su ejecución.

Una aplicación Web típica recogerá datos del usuario (primer nivel), los enviará al servidor, que ejecutará un programa (segundo y tercer nivel) y cuyo resultado será formateado y presentado al usuario en el navegador (primer nivel otra vez).

Figura 10. Arquitectura Web de tres niveles.



3.2 APLICACIONES DISTRIBUIDAS

3.2.1 Evolución de las Aplicaciones. La evolución de las aplicaciones software en los últimos años ha venido de la mano de la integración de la informática con el mundo de las telecomunicaciones, debido en gran parte al desarrollo de las redes locales y especialmente al auge de Internet.

- **Arquitectura Monolítica.** Desde sus inicios con las aplicaciones de arquitectura monolítica en las cuales la interfaz de usuario, la lógica de procesos y el manejo de la información almacenada estaba todo ligado o mezclado entre sí; de manera que en el caso de que el equipo fallará el sistema entero quedaba inservible, sin contar con riesgos tales como la pérdida absoluta de los datos. Las primeras soluciones vinieron con los denominados sistemas centralizados, que incluían servicios de red, en los que un solo computador con una o varias unidades centrales de procesamiento recibía las peticiones de los usuarios que se conectaban a él mediante terminales tontas; sin embargo, debido a razones tales

como falta de escalabilidad, altos costos, entre otros hacían de este un modelo poco atractivo.

- **Arquitectura Cliente / Servido.** Caracterizado por dividir la funcionalidad de las aplicaciones en dos roles perfectamente definidos: cliente (las maquinas desde las que se conectarían los usuarios, estaciones de trabajo donde se ejecutaría al menos la parte de la interfaz de usuario) y servidor (En ellos normalmente residirían las bases de datos).

La aparición de esta arquitectura fue facilitada entre otras por las siguientes razones:

- La depreciación del Hardware.
- La aparición de los ordenadores o computadores personales.
- El desarrollo de redes locales.

De modo abstracto, el servidor ofrece una serie de servicios que pueden ser utilizados por los clientes para completar la funcionalidad de la aplicación. Una interacción básica implica a un cliente que inicia una petición de algún servicio a un servidor. El servidor entonces realiza la función especificada por el cliente, devolviendo los posibles resultados que el servicio genera.

En la práctica, los clientes y servidores se implementan como procesos que se están ejecutando en máquinas conectadas a una red. La arquitectura C/S[8] se organiza en niveles o capas, existiendo arquitecturas de dos ó tres o en general n capas.

- **Arquitectura de 3 Capas.** La arquitectura C/S más popular es la de tres capas. El cliente se encarga de proporcionar la interfaz de usuario. El nivel

[8] C/S: Arquitectura Cliente/Servidor

intermedio se encarga de implementar la lógica de la aplicación, en el último nivel se encuentra la lógica y gestión de datos. Las funciones que realizan los clientes son muy sencillas de manera que varios clientes puedan reutilizar servicios estándares definidos en alguno de los niveles intermedios (estos servicios se encuentran encapsulados en los procesos que constituyen la lógica de la aplicación). Al estar dividida en partes más pequeñas, facilita el proceso de distribución de funcionalidad de la aplicación en los procesadores más adecuados. Un ejemplo de este tipo de sistemas C/S multicapa son las aplicaciones Web.

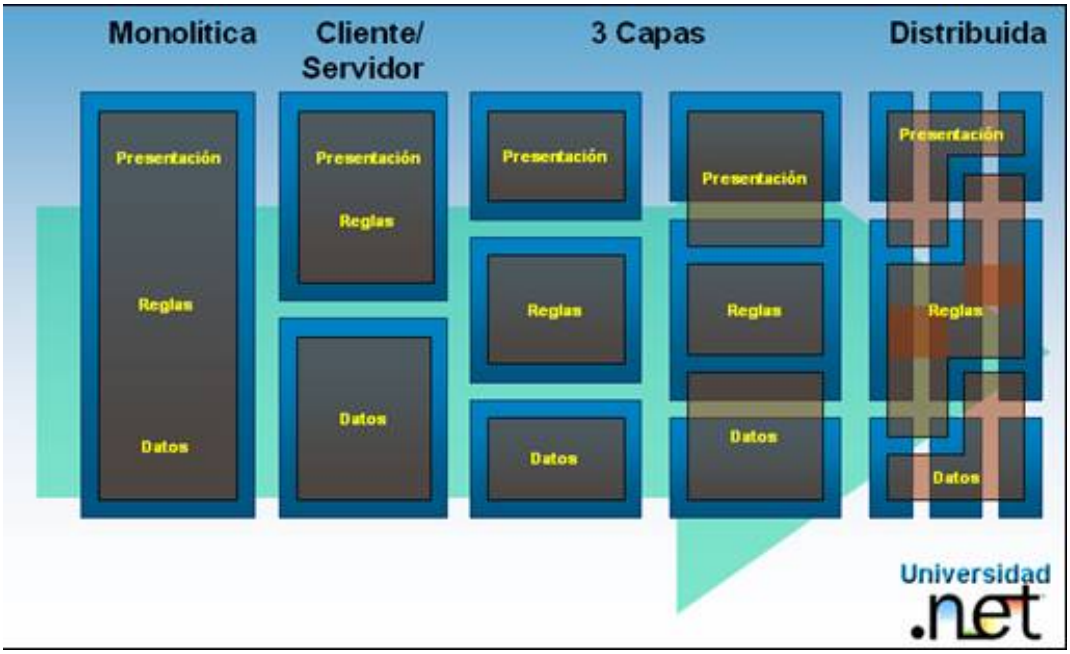
- **Sistemas Distribuidos.** Los sistemas distribuidos son el nivel más alto de la filosofía del modelo cliente servidor. Basados en la creación de componentes reutilizables u “Objetos” (con un significado similar al término “Objeto” de la programación Orientada a Objetos. POO); presentan la particularidad de no hacer explícitos los papeles de cliente y servidor ya que estos roles pueden cambiar en el tiempo.

Por tanto, un sistema de programación distribuida basada en componentes (Objetos) presenta las características de un sistema de programación orientada a objetos y de un sistema descentralizado o distribuido. Estas características pueden resumirse en:

- Distribución
- Transparencia
- Integridad de datos
- Tolerancia a fallos
- Disponibilidad
- Capacidad de recuperación
- Autonomía de los objetos
- Concurrencia de programas
- Concurrencia de objetos

- Mejora en el rendimiento.

Figura 11. Evolución de las Aplicaciones



3.2.2 Filosofía y ventajas de los sistemas distribuidos. El principio fundamental de las aplicaciones distribuidas es la división lógica de una aplicación en tres capas fundamentales:

- Presentación
- Lógica empresarial
- Acceso a datos y almacenamiento

Los programadores pueden generar aplicaciones altamente escalables y flexibles, organizando las aplicaciones según estos criterios. Para ello, deberán usar técnicas de programación basadas en componentes:

- Representativos de las reglas de negocio y el acceso a datos
- Que pueden distribuirse entre servidores

Un modelo simple de aplicación distribuida consiste en un cliente que comunica con la capa intermedia, que a su vez es el servidor de aplicaciones y una aplicación que contiene la lógica empresarial. La aplicación, a su vez, comunica con una base de datos que suministra y almacena datos.

Dentro de las ventajas que este modelo ofrece podemos contar:

- Proporciona una escalabilidad, capacidad de administración y utilización de recursos mejorados.
- Cada capa es un grupo de componentes que realiza una función específica.
- Se puede actualizar una capa sin recompilar otras capas.

3.2.3 Diseño de Aplicaciones Distribuidas. El diseño de aplicaciones modernas involucra la división de una aplicación en múltiples capas; la interfase de usuario, la capa media de objetos de negocios, y la capa de acceso a datos.

- Interfaz de usuario (Capa de Presentación): Interactuar con otros usuarios, Interactuar con aplicaciones externas o servicios.

- Procesos de negocios (Capa de Negocios): Cálculos u otros procesos de negocios, Ejecución de reglas de negocios, Validación de datos relacionados al negocio.

- Procesos de datos (Capa de Servicios de Datos): Manipulación de datos, Ejecución de las reglas de datos relacional.

- **Capa de Presentación.** La capa de Presentación provee su aplicación con una interfase de usuario (IU). Aquí es donde su aplicación presenta información a los usuarios y acepta entradas o respuestas del usuario para usar por su programa. Idealmente, la IU no desarrolla ningún procesamiento de negocios o reglas de validación de negocios. Por el contrario, la IU debería relegar sobre la capa de negocios para manipular estos asuntos. Esto es importante, especialmente hoy en día, debido a que es muy común para una aplicación tener múltiples IU, o para sus clientes o usuarios, que le solicitan que elimine una IU y la remplace con otra. Por ejemplo, usted puede desarrollar una aplicación Win32 (un programa en Visual Basic) y entonces solicitársele remplazarla con una página HTML, quizás usando tecnología ASP.

Una de las mayores dificultades y factores importantes cuando desarrollamos aplicaciones cliente/servidor es mantener una separación completa entre la presentación, la lógica de negocios y los servicios de datos. Es muy tentador para los desarrolladores mezclar una o más capas; poniendo alguna validación u otro proceso de negocios dentro de la capa de presentación en vez de en la capa de negocios.

- **Capa de Negocios.** Toda aplicación tiene código para implementar reglas de negocios, procesos relacionados a los datos o cálculos y otras actividades relativas a los negocios. Colectivamente este código es considerado para formar la capa de negocios. Otra vez, uno de los principios del diseño lógico cliente/servidor, la lógica de negocios debe mantenerse separada de la capa de presentación y de los servicios de datos. Esto no significa necesariamente que la lógica de negocios está en cualquier parte, por el contrario, esta separación es en un sentido lógico.

Hay muchas formas de separar la lógica de negocios. En términos orientados a objetos, usted debería encapsular la lógica de negocios en un conjunto de objetos

o componentes que no contienen presentación o código de servicios de datos. Teniendo separada lógicamente su lógica de negocios de ambas, la capa de presentación y servicios de datos, usted ganará en flexibilidad en término de donde usted puede almacenar físicamente la lógica de negocios. Por ejemplo, usted puede seleccionar almacenar la lógica de negocios sobre cada estación de cliente, u optar por ejecutar la lógica de negocios sobre un servidor de aplicaciones, permitiendo a todos los clientes acceder a un recurso centralizado.

Los objetos de negocios son diseñados para reflejar o representar sus negocios. Ellos se convierten en un modelo de sus entidades de negocios e interrelaciones. Esto incluye tanto objetos físicos como conceptos abstractos. Estos son algunos ejemplos de objetos del mundo real: un empleado, un cliente, un producto, una orden de compra.

Todos estos son objetos en el mundo físico, y la idea en su totalidad detrás de usar objetos de negocios de software, es crear una representación de los mismos objetos dentro de su aplicación. Sus aplicaciones pueden hacer que estos objetos interactúen unos con otros como ellos lo hacen en el mundo real. Por ejemplo, un empleado puede crear una orden de compra a un cliente que contiene una lista de productos. Siguiendo esta lógica usted puede crear objetos de negocios de una orden conteniendo el código necesario para administrarse a si mismo, así usted nunca necesitará replicar código para crear ordenes, usted solo usará el objeto.

Similarmente, un objeto cliente contiene y administra sus propios datos. Un buen diseño de un objeto cliente contiene todos los datos y rutinas necesitadas para representarlo a través del negocio completo, y puede ser usado a través de toda la aplicación de ese negocio.

No toda la lógica de negocio es la misma. Alguna lógica de negocio es un proceso intensivo de datos, requiriendo un eficiente y rápido acceso a la base de datos.

Otras no requieren un frecuente acceso a los datos, pero es de uso frecuente por una interfase de usuario robusta para la validación en la entrada de campos u otras interacciones de usuarios. Si nosotros necesitamos una validación al nivel de pantallas y quizás cálculos en tiempo real u otra lógica de negocios, pudiéramos considerar este tipo de lógica de negocios para ser parte de la IU, ya que en su mayor parte es usada por la interfase de usuario.

Una alternativa de solución es dividir la capa de lógica de negocios en dos:

- Objetos de negocios de la IU.
- Objetos de negocios de datos.

Un ejemplo del objeto Empleado de la capa objetos de negocios de la IU proveerá propiedades y métodos para usar por el diseñador de la interfase de usuario. Ejemplo de propiedades y métodos pudieran ser: IDEmpleado, Nombre, Dirección, etc., y como métodos crear una de compra, etc. El objeto Empleado de la capa de objetos de negocios de datos será responsable de los mecanismos de persistencias, interactuar con la base de datos. Los objetos de esta capa son considerados sin estado, solo poseen métodos.

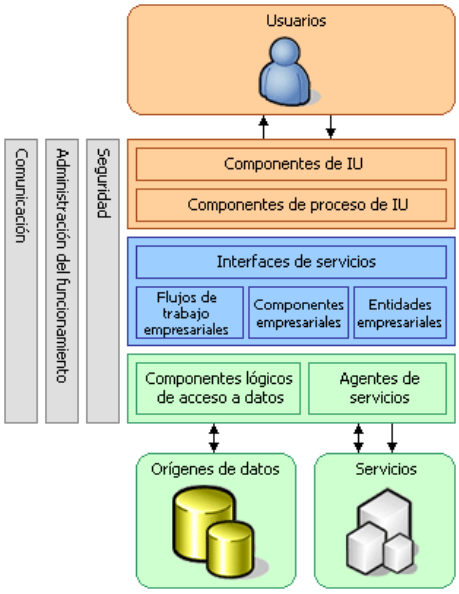
- **Capa de Servicios de Datos.** Muchas aplicaciones interactúan con datos, los almacenan en alguna forma de bases de datos. Hay algunas funciones básicas que son comunes a todos los procesos. Estas incluyen:

- Crear datos,
- leer datos,
- actualizar datos y
- eliminar datos.

Adicionalmente, nosotros tenemos servicios más avanzados disponibles, tales como: búsquedas, ordenamientos, filtrados, etc.[9]

3.2.4 Arquitectura de aplicaciones distribuidas. El diseño de aplicaciones distribuidas no es una tarea sencilla. Es necesario tomar un gran número de decisiones a nivel de arquitectura, diseño e implementación. Estas decisiones tendrán un impacto en las "capacidades" de la aplicación (seguridad, escalabilidad, disponibilidad y mantenimiento, entre otras), así como en la arquitectura, el diseño y la implementación de la infraestructura de destino. La guía le ayudará a comprender las distintas opciones que se presentan a la hora de diseñar las capas de una aplicación distribuida; estas opciones se presentan como un conjunto de capas de componentes que se podrán utilizar para modelar la aplicación. En la figura 3.5 se muestran las capas de los componentes lógicos que este documento utiliza para estructurar sus instrucciones.

Figura 12. Capas de componentes de servicios y aplicaciones distribuidas



[9] http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml

- **Componentes Software.** Hasta ahora se ha establecido que una aplicación distribuida es una aplicación de arquitectura de 3 capas basada en componentes reutilizables que permiten la descentralización del procesamiento brindando escalabilidad, confiabilidad y rendimiento. En las próximas secciones se definirán conceptos como el de “Componente”, sus tipos y la función de cada uno en las diferentes capas de una aplicación distribuida.

El desarrollo de software basado componentes es la evolución natural de la ingeniería software para mejorar la calidad, disminuir los tiempos de desarrollo y gestionar la creciente complejidad de los sistemas.

El objetivo de la tecnología de componentes software es construir aplicaciones complejas mediante ensamblado de módulos (componentes) que han sido previamente diseñados por otras personas a fin de ser re-usados en múltiples aplicaciones. La arquitectura software de una aplicación basada en componentes consiste en uno o un número pequeño de componentes específicos de la aplicación (que se diseñan específicamente para ella), que hacen uso de otros muchos componentes prefabricados que se ensamblan entre sí para proporcionar los servicios que se necesitan en la aplicación.

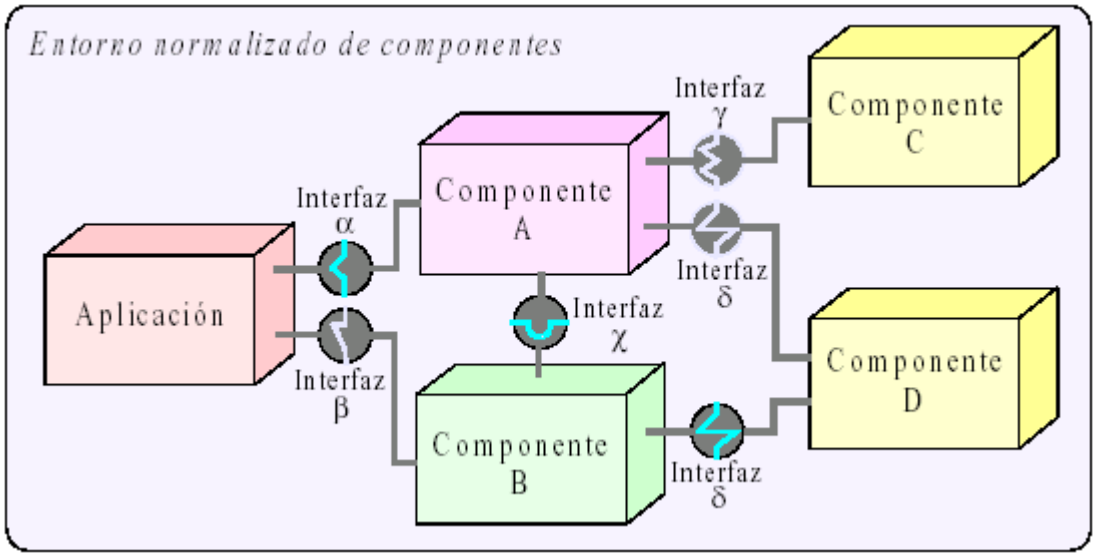
En la tecnología de componentes la interfaz constituye el elemento básico de interconectividad. Cada componente debe describir de forma completa las interfaces que ofrece, así como las interfaces que requiere para su operación. Y debe operar correctamente con independencia de los mecanismos internos que utilice para soportar la funcionalidad de la interfaz.

Características muy relevantes de la tecnología de programación basada en componentes son la modularidad, la reusabilidad y componibilidad y en todos ellos coincide con la tecnología orientada a objetos de la que se puede considerar una evolución. Sin embargo, en la tecnología basada en componentes también se

requiere robustez ya que los componentes han de operar en entornos mucho más heterogéneos y diversos.

- **Entornos normalizados de desarrollo de componentes software.**[10]

Figura 13. Entorno normalizado de componentes



Para que una arquitectura de componentes pueda operar es necesario disponer de un entorno normalizado que proporcione soporte a los mecanismos con que se comunican las interfaces:

- **COM (Component Object Model).** Los lenguajes de programación clásicos fueron diseñados para desarrollar aplicaciones secuenciales compuestas de módulos, todos ellos codificados con un solo lenguaje. Sin embargo, hay situaciones en las que no es práctico restringirse al uso de un único lenguaje. La tecnología COM aborda la solución a este problema proporcionando un sencillo,

[10] http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml

pero a la vez potente modelo para construir sistemas software a partir de la interacción de objetos (componentes).

COM define un estándar binario (esto implica que es independiente del lenguaje de programación) para objetos y la intercomunicación entre ellos. Toda comunicación se realiza a través de operaciones que son proporcionadas dentro de interfaces. El diseñador invoca las operaciones que necesita directamente, incluso si el objeto destinatario está localizado en otro proceso o en otra máquina.
El modelo de programación COM esta basado en la distribución de código de clases en componentes binarios. Esto significa que el software (componentes) que se adhiere a COM, puede ser rehusado sin ninguna dependencia de código fuente. Los desarrolladores pueden exponer sus trabajos como ficheros binarios sin dar a conocer sus algoritmos.

El desarrollo basado en componentes resuelve muchos de los problemas asociados con las aplicaciones monolíticas. Permite al grupo de desarrollo exponer ficheros binarios en vez de código fuente. Los componentes binarios pueden ser actualizados independientemente y reemplazados, lo que se hace mucho más fácil mantener y extender una aplicación después de que esta ha sido puesta en explotación.

- **MTS (Microsoft Transaction Server).** MTS es una pieza de software que fue creada para Windows NT Server. Como su nombre implica, MTS permite a los objetos de la capa media correr sobre Windows NT Server y controlar las transacciones distribuidas, es decir, permite a los componentes ser esparcidos por la red y que se ejecuten en otras computadoras con sistema operativo Windows NT Server. MTS provee un entorno de ejecución para objetos COM, adicionando soporte para la seguridad, soporte para administración y configuración. Es posible administrar varios servidores desde una simple computadora.

- ***COM+.*** COM+, no es más que la integración de la arquitectura COM y MTS (Microsoft Transation Server). A diferencia de MTS, esta nueva capa de ejecución no es opcional, COM+ es parte por defecto de la instalación del sistema operativo Windows 2000.

Como COM, COM+, es basado sobre componentes binarios y programación basada en interfaces. Los Componentes COM+ pueden ser actualizados y extendidos una vez que estén en explotación sin afectar a las aplicaciones clientes que los usan en la producción.

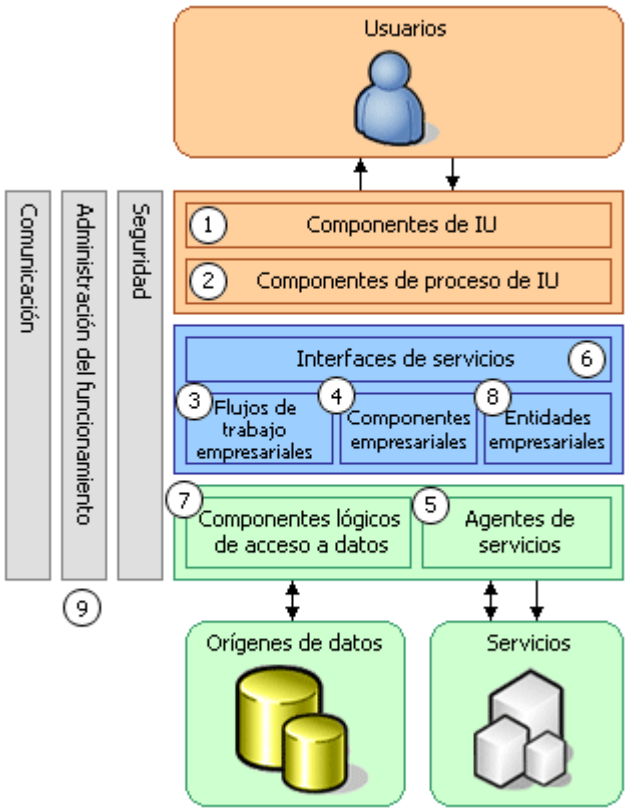
De este modo, la combinación de la tecnología COM+ junto con las técnicas de programación orientada a objetos, nos ofrece una importante simplificación en el proceso de desarrollo de aplicaciones informáticas.

- **Tipos de componentes[11].** El análisis de la mayoría de las soluciones empresariales basadas en modelos de componentes por capas muestra que existen varios tipos de componentes habituales. En la figura se muestra una ilustración completa en la que se indican estos tipos de componentes.

Aunque la lista que se muestra en la figura no es completa, representa los tipos de componentes de software más comunes encontrados en la mayoría de las soluciones distribuidas.

[11] http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh2.asp

Figura 14. Tipos de componentes utilizados en un escenario comercial típico.



Los tipos de componentes identificados un escenario de diseño normal son:

1. Componentes de interfaz de usuario (IU). La mayor parte de las soluciones necesitan ofrecer al usuario un modo de interactuar con la aplicación. En el ejemplo de aplicación comercial, un sitio Web permite al cliente ver productos y realizar pedidos, y una aplicación basada en el entorno operativo Microsoft Windows® permite a los representantes de ventas escribir los datos de los pedidos de los clientes que han telefoneado a la empresa. Las interfaces de usuario se implementan utilizando formularios de Windows Forms, páginas Microsoft ASP.NET, controles u otro tipo de tecnología que permita procesar y dar

formato a los datos de los usuarios, así como adquirir y validar los datos entrantes procedentes de éstos.

2. **Componentes de proceso de usuario.** En un gran número de casos, la interactuación del usuario con el sistema se realiza de acuerdo a un proceso predecible. Por ejemplo, en la aplicación comercial, podríamos implementar un procedimiento que permita ver los datos del producto. De este modo, el usuario puede seleccionar una categoría de una lista de categorías de productos disponibles y, a continuación, elegir uno de los productos de la categoría seleccionada para ver los detalles correspondientes. Del mismo modo, cuando el usuario realiza una compra, la interactuación sigue un proceso predecible de recolección de datos por parte del usuario, por el cual éste en primer lugar proporciona los detalles de los productos que desea adquirir, a continuación los detalles de pago y, por último, la información para el envío. Para facilitar la sincronización y organización de las interactuaciones con el usuario, resulta útil utilizar componentes de proceso de usuario individuales. De este modo, el flujo del proceso y la lógica de administración de estado no se incluye en el código de los elementos de la interfaz de usuario, por lo que varias interfaces podrán utilizar el mismo "motor" de interactuación básica.

3. **Flujos de trabajo empresariales.** Una vez que el proceso de usuario ha recopilado los datos necesarios, éstos se pueden utilizar para realizar un proceso empresarial. Por ejemplo, tras enviar los detalles del producto, el pago y el envío a la aplicación comercial, puede comenzar el proceso de cobro del pago y preparación del envío. Gran parte de los procesos empresariales conllevan la realización de varios pasos, los cuales se deben organizar y llevar a acabo en un orden determinado. Por ejemplo, el sistema empresarial necesita calcular el valor total del pedido, validar la información de la tarjeta de crédito, procesar el pago de la misma y preparar el envío del producto. El tiempo que este proceso puede tardar en completarse es indeterminado, por lo que sería preciso administrar las

tareas necesarias, así como los datos requeridos para llevarlas a cabo. Los flujos de trabajo empresariales definen y coordinan los procesos empresariales de varios pasos de ejecución.

4. Componentes empresariales. Independientemente de si el proceso empresarial consta de un único paso o de un flujo de trabajo organizado, la aplicación requerirá probablemente el uso de componentes que implementen reglas empresariales y realicen tareas empresariales. Por ejemplo, en la aplicación comercial, deberá implementar una funcionalidad que calcule el precio total del pedido y agregue el costo adicional correspondiente por el envío del mismo. Los componentes empresariales implementan la lógica empresarial de la aplicación.

5. *Agentes de servicios:* Cuando un componente empresarial requiere el uso de la funcionalidad proporcionada por un servicio externo, tal vez sea necesario hacer uso de código para administrar la semántica de la comunicación con dicho servicio. Por ejemplo, los componentes empresariales de la aplicación comercial descrita anteriormente podría utilizar un agente de servicios para administrar la comunicación con el servicio de autorización de tarjetas de crédito y utilizar un segundo agente de servicios para controlar las conversaciones con el servicio de mensajería. Los agentes de servicios permiten aislar las idiosincrasias de las llamadas a varios servicios desde la aplicación y pueden proporcionar servicios adicionales, como la asignación básica del formato de los datos que expone el servicio al formato que requiere la aplicación.

6. Interfaces de servicios Para exponer lógica empresarial como un servicio, es necesario crear interfaces de servicios que admitan los contratos de comunicación (comunicación basada en mensajes, formatos, protocolos, seguridad y excepciones, entre otros) que requieren los clientes. Por ejemplo, el servicio de autorización de tarjetas de crédito debe exponer una interfaz de servicios que describa la funcionalidad que ofrece el servicio, así como la semántica de

comunicación requerida para llamar al mismo. La interfase de servicios también se denominan fachadas empresariales.

7. **Componentes lógicos de acceso a datos.** La mayoría de las aplicaciones y servicios necesitan obtener acceso a un almacén de datos en un momento determinado del proceso empresarial. Por ejemplo, la aplicación empresarial necesita recuperar los datos de los productos de una base de datos para mostrar al usuario los detalles de los mismos, así como insertar dicha información en la base de datos cuando un usuario realiza un pedido. Por tanto, es razonable abstraer la lógica necesaria para obtener acceso a los datos en un capa independiente de componentes lógicos de acceso a datos, ya que de este modo se centraliza la funcionalidad de acceso a datos y se facilita la configuración y el mantenimiento de la misma.

8. **Componentes de entidad empresarial.** La mayoría de la aplicaciones requieren el paso de datos entre distintos componentes. Por ejemplo, en la aplicación comercial es necesario pasar una lista de productos de los componentes lógicos de acceso a datos a los componentes de la interfaz de usuario para que éste pueda visualizar dicha lista. Los datos se utilizan para representar entidades empresariales del mundo real, como productos o pedidos. Las entidades empresariales que se utilizan de forma interna en la aplicación suelen ser estructuras de datos, como conjuntos de datos, DataReader o secuencias de lenguaje de marcado extensible (XML), aunque también se pueden implementar utilizando clases orientadas a objetos personalizadas que representan entidades del mundo real necesarias para la aplicación, como productos o pedidos.

9. **Componentes de seguridad, administración operativa y comunicación.** La aplicación probablemente utilice también componentes para realizar la

administración de excepciones, autorizar a los usuarios a que realicen tareas determinadas y comunicarse con otros servicios y aplicaciones.

3.3 TECNOLOGÍA .NET

3.3.1 Que es .Net. .Net está diseñada para brindar servicios informáticos a través de redes TCP/IP y Web. Para lograr esto, debían generarse estándares que permitieran una fluida comunicación en Internet. Recordemos que ésta es una red heterogénea donde no se puede saber a ciencia cierta en cuál plataforma tecnológica se apoya el servidor, y en cuál el cliente.

Tales estándares se han ido generando poco a poco a través del Consorcio de la WWW (o W3C). Allí se han estado estructurando estándares para el intercambio fluido de datos, tales como XML, WSDL, UDDI, SOAP, entre otros. Estos estándares, así, sirven como base de toda la plataforma tecnológica que Microsoft ha presentado como .NET, y que otras empresas han presentado con sus propios nombres. En realidad, .NET es una propuesta funcional para hacer lo que también se puede hacer con otras plataformas: comunicarse fluidamente a través de Internet en la forma de Servicios Web XML, pero la intención de Microsoft es la de presentar la mejor opción para lograrlo.

Con esta finalidad, la empresa de las ventanas se ha ido deshaciendo de muchas tecnologías propietarias y las ha transformado paulatinamente de acuerdo a estándares. Windows XP, Office XP, .NET Framework, la nueva gama de servidores, y el VS.NET son parte de ese esfuerzo para ceñirse en lo posible a los estándares, y competir con calidad de servicios, y no con la desgastante invención continua de tecnologías o en la reinvención del hilo negro.

Es por ello que Microsoft prácticamente ha rediseñado toda su infraestructura tecnológica para ceñirla a los estándares internacionales. Por esta razón, en últimas fechas han aparecido tantas nuevas aplicaciones -hasta el propio Windows XP-, y otras que pronto estarán en el mercado, como la nueva familia de servidores Windows .NET.

Visual Studio .NET es parte del rompecabezas de esta tecnología en lo que respecta al desarrollo de aplicaciones y servicios que puedan aprovechar las bondades de Internet y redes Web para poder funcionar de mejor forma. Así, lo que sólo podía usarse por quienes tuvieran Windows (como los componentes COM, DCOM o COM+, así como los ActiveX), ahora mediante SOAP, XML, WSDL, UDDI, y otros, podrá ser utilizado por quien se base en estas tecnologías (sin importar su plataforma). Esto permitirá que se ofrezcan, efectivamente, servicios por Internet, que podrían coadyuvar a generar mejores oportunidades de negocio, ya sea al arrendar tales servicios, o al facilitar el desarrollo de sitios Web que mejoren las capacidades funcionales ofrecidas a los usuarios.

Por ejemplo: por medio de los Servicios Web XML ya podría diseñarse un sitio que reciba tarjetas de crédito, que sean validadas por alguna entidad confiable, y tal servicio se agregaría a la aplicación como si fuera un procedimiento u objeto integrado a la aplicación que lo solicite. Es como los objetos de un programa, pero que expondrán su funcionalidad en redes basadas en protocolos Web pueden inundar a la red de redes, o a la intranet, y que puedan usarse tanto en aplicaciones Web como en aplicaciones de escritorio.

De esta forma, un proveedor de servicios (una empresa de cualquier tamaño) puede valerse de la tecnología .NET para agilizar sus procesos productivos relacionados con Internet, redes Web o aplicaciones de escritorio. Si es usuario, podrá encontrarse con una mayor gama de servicios y más ricos en facultades conforme estas tecnologías tomen mayor fuerza en Internet. Su competitividad

como prestador de Servicios Web XML deberá basarse en su calidad y, por ende, confiabilidad. Cierto es que habrá más de un servicio que proporcionará una funcionalidad determinada, pero en su riqueza, calidad y confiabilidad estará basado su éxito. Es por ello que era tan importante comprender los conceptos de Servicio, Calidad y Confiabilidad que sirvieron como preámbulo a los conceptos que he vertido respecto a la tecnología Microsoft .NET.

Así pues, todo lleva a concluir que Internet y las redes Web ahora sí serán utilizados como un medio para apalancar a las empresas que ya estén establecidas físicamente en algún lugar. Se facilitará el intercambio de información y la agilidad para realizar una aplicación. Se reducirán los tiempos de desarrollo, y se obtendrían mejores resultados cuando se realice alguna tarea en este entorno. Así pues, Windows XP, Visual Studio .NET, Office XP y toda la gama de tecnologías que Microsoft recién ha presentado y que próximamente presentará se ciñe al nuevo esquema de la tecnología .NET. Si este tema es de su interés, puede encontrar mayor información en www.microsoft.com/net o en www.microsoft.com/latam/msdn.

3.3.2 Orígenes e Ideología. Según la descripción del ingeniero David Garza Marín en su artículo titulado “Microsoft .NET y su ideología: ¿Por qué nació?”, la tecnología .NET esta orientada a brindar “servicios” con calidad basados en principios de especialización que establecen que delegar actividades en personas o empresas que cuentan con la experiencia y los métodos para hacerlo cumpliendo con las metas o expectativas que se tienen con respecto al producto resulta en una reducción de tiempo y esfuerzo.

Esto se evidencia en el manejo de las herramientas .NET que le ofrecen al desarrollador rutinas completas de construcción y manejo de bases de datos y aplicaciones Web embebidas en objetos programables con Etiquetas Inteligentes ó “Smart Tags” que permiten que se definan aspectos como el origen de datos y la

forma como se mostrará la información sin tener que escribir una sola línea de código, definiendo dichas características a través de Ayudantes o “Wizards”; así el Ingeniero de desarrollo se preocupa de aspectos más importantes como la integridad y escalabilidad del sistema, al igual que el asegurar que se cumplan las expectativas del cliente o usuario final de la aplicación. El uso de estas herramientas permite hacer mayor énfasis en la fase de diseño sin que esto signifique que no se tenga apoyo en este aspecto por parte de esta tecnología, por ejemplo el “Diseñador de Clases de Visual Studio .NET” permite visualizar la estructura de un conjunto determinado de clases y las relaciones existentes entre las mismas, crear clases nuevas a través de un entorno de diseño visual y modificar las clases existentes de manera simple.

3.3.3 Visual Studio .NET12. Una de las piezas más importantes en el esquema de Microsoft .NET es el propio Visual Studio .NET. Éste es un entorno de trabajo tan distinto a lo hecho anteriormente por la empresa de las ventanas (o por cualquier otra) que Microsoft tuvo que darle otro nombre y prácticamente salirse de la secuencia de versiones de la plataforma de Visual Studio, aunque, al final, también lo mencionó como la séptima versión de la saga.

Para comprender a Visual Studio .NET hay que pensar como desarrollador y hacer una ligera historia de la tecnología de programación. Los primeros días de la programación se distinguieron por el código lineal, poco útil en aplicaciones grandes dado que arrojaba engendros difíciles de mantener. El código lineal evolucionó al código estructurado, que mejoró la disposición del código y que, por mucho tiempo, se estableció como el rey de la codificación.

Sin embargo, ya desde 1969, se avizoró otro esquema de programación que lograría una revolución real hasta mediados de la década de los años ochenta,

[12] Tomado de http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art29.asp

con la presentación de C++, y más específicamente en la de los noventa con la proliferación de los entornos gráficos que le dieron mayor razón de ser.

La máxima de la programación orientada a objetos es la reutilización; es decir, aprovechar el código ya hecho para evitar la duplicación de trabajo. Sin embargo, una de las mayores limitantes de esto es, precisamente, la diversidad de lenguajes de programación: Un objeto generado en C++ no puede usarse directamente en lenguajes como Visual Basic, Object Pascal o SmallTalk. En realidad, un objeto generado en C++ sólo podrá usarse en C++, así como uno de Visual Basic, sólo podrá usarse en Visual Basic (claro está que podría compilar el objeto y hacerlo binario para, así, poder integrarlo en un entorno de programación, pero ello traería la necesidad de usar protocolos propietarios como COM, DCOM, COM+, CORBA, SOM, etcétera, lo cual rompe con ese sueño del reuso integral).

Con esto en mente, Microsoft quiso aprovechar la oportunidad de la presentación de Microsoft .NET para generar un marco de trabajo que fuera aprovechado por cualquier lenguaje de programación que se ciñera a sus estándares. Ese marco de trabajo es el .NET Framework, que es el meollo de la tecnología .NET. En pocas palabras, el .NET Framework es un cúmulo de clases expuestas para que, quien quiera, haga uso de su funcionalidad. A su vez, este cúmulo de clases conforma un estándar abierto que puede integrarse a cualquier plataforma o lenguaje.

Esta apertura permitió el diseño de un entorno de desarrollo tan amplio como lo es el Visual Studio .NET, que no sólo incluye los lenguajes de C# .NET, Visual Basic .NET, Visual C++ .NET y, próximamente, J# .NET, sino que hay más de 20 lenguajes de otros fabricantes que pueden funcionar en él, como Pascal .NET, Cobol .NET, y muchos otros.

El que tantos lenguajes distintos puedan funcionar en un mismo entorno, tiene un beneficio adicional: puede incluirse un objeto hecho en cualquiera de estos

lenguajes en un proyecto generado en otro lenguaje. Por ejemplo, pueden incluirse clases generadas con C# .NET en un proyecto de Visual Basic .NET. Las clases de C# .NET no tendrán que compilarse para que esto sea posible, dado que el entorno interpretará adecuadamente las instrucciones que tenga para poder aprovechar su funcionalidad sin problemas.

De esta forma, Visual Studio .NET es lo más cercano a la máxima de la programación orientada a objetos: la reutilización. Claro está que todos los lenguajes tendrán que cumplir con las prerrogativas del .NET Framework, pero con la cantidad de lenguajes que están apareciendo para este entorno, esto podría ya no ser un problema.

Es importante mencionar que la tecnología Microsoft .NET es mucho más que lo poco que puede mencionarse en un espacio como el ocupado hasta ahora. Por ello mismo, puede hacer uso de los recursos que ponemos a su disposición en este medio (y en muchos otros) para ampliar su información al respecto. No obstante, espero haberle ayudado para comprender mejor el porqué de esta tecnología y empezar a aprovecharla aún mejor.

3.3.4 Desarrollo de Aplicaciones Web con Visual Studio .NET. .NET Framework y el sistema operativo Windows aceptan en su totalidad los elementos que conforman cada segmento de este modelo. Entre sus muchos servicios se encuentran los de directorio, seguridad, administración y comunicaciones, que operan en las tres capas. Las herramientas de programación, que incluyen el sistema de desarrollo Visual Studio .NET, permiten a los programadores generar componentes de aplicaciones entre las diferentes capas.

- **Introducción a aplicaciones Web ASP.NET en Visual Studio.** Visual Studio .NET permite la creación de aplicaciones basadas en tecnología Web, desde un sitio tradicional basado en HTML puro hasta complejas aplicaciones empresariales

que se ejecuten en Intranet o Internet, sofisticadas aplicaciones distribuidas basadas en componentes y servicios de intercambio de datos mediante XML.

- **Aplicaciones Web ASP.NET de Visual Studio[13].** Las aplicaciones Web de Visual Studio se generan en torno a ASP.NET. ASP.NET es una plataforma, que incluye objetos y controles de tiempo de diseño y un contexto de tiempo de ejecución, para desarrollar y ejecutar aplicaciones en un servidor Web.

ASP.NET, a su vez, forma parte de .NET Framework, de modo que proporciona acceso a todas las funciones de este marco de trabajo. Por ejemplo, puede crear aplicaciones Web ASP.NET mediante cualquier lenguaje de programación .NET (Visual Basic, C#, Extensiones administradas para C++ y muchas otras) y las utilidades de depuración .NET. Puede tener acceso a los datos utilizando ADO.NET. De manera similar, puede tener acceso a los servicios del sistema operativo utilizando las clases de .NET Framework.

Las aplicaciones Web ASP.NET se ejecutan en un servidor Web configurado con Microsoft Internet Information Services (IIS). Sin embargo, no necesitará trabajar directamente con IIS. Puede programar utilidades de IIS mediante clases ASP.NET; Visual Studio controla las tareas de administración de archivos, tales como la creación de aplicaciones IIS, cuando es necesario y proporciona los medios para implementar las aplicaciones Web en IIS.

- **Dónde es adecuado Visual Studio.** Como con cualquier aplicación .NET, si dispone de .NET Framework, puede crear aplicaciones ASP.NET mediante editores de texto, un compilador de línea de comandos y otras herramientas sencillas. Puede copiar archivos manualmente en IIS para implementar la aplicación.

[13]http://msdn.microsoft.com/library/SPA/vbcon/html/vbconIntroductionToWebApplicationsInVisualStudio.asp

También se puede utilizar Visual Studio. Cuando utilice Visual Studio para crear aplicaciones Web, estará creando esencialmente la misma aplicación que podría crear a mano. Es decir, Visual Studio no crea un tipo diferente de aplicación Web; el resultado final continúa siendo una aplicación Web ASP.NET.

La ventaja de utilizar Visual Studio reside en que proporciona herramientas que facilitan el desarrollo de aplicaciones y lo hacen más rápido y confiable. Entre estas herramientas se incluyen:

- Diseñadores visuales para páginas Web con controles para arrastrar y colocar, y vistas de código (HTML) con comprobación de sintaxis.

- Editores de código inteligentes que incluyen finalización de instrucciones, comprobación de sintaxis y otras características de IntelliSense.

- Compilación y depuración integradas.

- Utilidades de administración de proyectos para la creación y administración de archivos de aplicación, incluida la implementación en servidores locales o remotos.

Si ha utilizando antes Visual Studio, este tipo de características le resultarán familiares, ya que son similares a las que se utilizaban para crear aplicaciones en versiones anteriores de Visual Basic y Visual C++. Con Visual Studio .NET puede utilizar estas funciones para crear aplicaciones Web ASP.NET.

3.3.5 Información general: ASP.NET[14]. ASP.NET es más que una nueva versión de las páginas Active Server (ASP); proporciona un modelo de desarrollo Web unificado que incluye los servicios necesarios para que los programadores

[14] http://msdn.microsoft.com/library/SPA/vbcon/html/vbconIntroductionToWebApplicationsInVisualStudio.asp

creen aplicaciones Web para la empresa. Si bien ASP.NET es en gran medida compatible con la sintaxis de ASP, proporciona también un modelo de programación y una estructura nuevos para crear aplicaciones más escalables y estables que ayuden a proporcionar mayor protección. Las aplicaciones ASP se pueden ampliar agregándoles funcionalidad de ASP.NET.

ASP.NET es un entorno compilado basado en .NET. Se pueden crear aplicaciones en cualquier lenguaje compatible con .NET, como Visual Basic .NET, C# y JScript .NET. Además, .NET Framework está disponible en su totalidad para cualquier aplicación ASP.NET. Los programadores pueden aprovechar fácilmente las ventajas de estas tecnologías, que incluyen el entorno Common Language Runtime administrado, seguridad de tipos, herencia, etc.

ASP.NET se ha diseñado para funcionar sin problemas con editores HTML y otras herramientas de programación como Microsoft Visual Studio .NET. Todo esto, además de hacer más fácil la programación Web, ofrece todas las ventajas de estas herramientas, con una GUI (Interfaz Gráfica de Usuario, GUI por sus siglas en Inglés) que los programadores puede utilizar para ubicar controles de servidor en una página Web e integrar completamente la compatibilidad con la depuración.

A la hora de crear una aplicación ASP.NET, los programadores pueden utilizar formularios Web Forms o servicios Web XML o combinarlas de la manera que más les convenga. Las dos características son compatibles con la misma infraestructura, que permite utilizar esquemas de autenticación, almacenar en caché datos que se utilizan con frecuencia y personalizar la configuración de la aplicación, entre otras muchas cosas.

- Los formularios Web Forms permiten crear páginas Web basadas en formularios muy eficaces. Al crear estas páginas, se pueden usar controles de servidor ASP.NET para crear elementos comunes de la interfaz de usuario y programarlos

para que realicen las tareas comunes. Estos controles permiten crear con rapidez un formulario Web Forms a partir de componentes integrados reutilizables o personalizados, con un código de página simplificado.

▪ Un servicio Web XML proporciona los medios para obtener acceso a la funcionalidad del servidor de manera remota. Con los servicios Web XML, las empresas pueden exponer interfaces de programación a sus datos o lógica empresarial, que, a su vez, pueden obtener y manipular las aplicaciones de cliente y servidor. Los servicios Web XML permiten el intercambio de datos en escenarios cliente-servidor o servidor-servidor, utilizando estándares como los servicios de mensajería HTTP y XML para que los datos pasen los servidores de seguridad. Los servicios Web XML no están ligados a ninguna tecnología de componentes ni a ninguna convención de llamada a objetos concretas. En consecuencia, pueden tener acceso a los servicios Web XML los programas escritos en cualquier lenguaje, utilizando cualquier modelo de componentes y que se ejecuten en cualquier sistema operativo.

Cada uno de estos modelos puede aprovechar al máximo todas las características de ASP.NET, así como la eficacia de .NET Framework y Common Language Runtime de .NET Framework. Estas características y su utilización se describen así:

▪ Si tiene conocimientos de desarrollo de ASP, el nuevo modelo de programación de ASP.NET le resultará muy familiar. Sin embargo, el modelo de objetos ASP.NET ha cambiado de manera significativa con respecto a ASP, siendo más estructurado y orientado a objetos. Desafortunadamente, esto significa que ASP.NET no es totalmente compatible con las versiones anteriores; casi todas las páginas ASP existentes deberán modificarse en alguna medida para que puedan ejecutarse bajo ASP.NET. Además, los importantes cambios de Visual Basic .NET también implican que las páginas ASP escritas con Visual Basic Scripting Edition

por lo general no se portarán directamente a ASP.NET. Aún así, en la mayoría de los casos, los cambios necesarios sólo implicarán algunas líneas de código.

▪ El acceso a bases de datos desde aplicaciones ASP.NET es una técnica utilizada con frecuencia para mostrar datos a los usuarios que visitan un sitio Web. ASP.NET hace que tener acceso a bases de datos con esta finalidad sea más fácil que nunca. También permite administrar la base de datos desde el código.

▪ ASP.NET proporciona un modelo sencillo que permite que los programadores Web escriban que el código se ejecuta en el nivel de la aplicación. Los programadores pueden escribir dicho código en el archivo de texto Global.asax o en una clase compilada implementada como ensamblado. Este lógica puede incluir eventos del nivel de la aplicación, pero los programadores tienen la posibilidad de extender este modelo para que se ajuste a las necesidades de la aplicación Web.

▪ ASP.NET proporciona facilidades de aplicación y de estado de la sesión de fácil manejo, conocidas por los programadores ASP y compatibles con todas las demás API de .NET Framework.

▪ ASP.NET aprovecha las mejoras de rendimiento de .NET Framework y Common Language Runtime. Además, se ha diseñado para que ofrezca un rendimiento sensiblemente mejor que ASP y otras plataformas de programación Web. Todo el código de ASP.NET se compila, en lugar de interpretarse, lo que permite realizar enlaces en tiempo de diseño, establecer tipos inflexiblemente, compilar código nativo en modo Just-In-Time (JIT), entre otras muchas ventajas. En ASP.NET es muy fácil para los programadores eliminar módulos (por ejemplo, un modulo de sesión) que la aplicación que están programando no necesita. ASP.NET ofrece unos completos servicios de almacenamiento en caché (tanto servicios integrados como API de almacenamiento en caché). ASP.NET se

suministra con contadores de rendimiento que los programadores y los administradores del sistema pueden supervisar para probar nuevas aplicaciones y recopilar estadísticas de aplicaciones existentes.

▪ .NET Framework y ASP.NET proporcionan esquemas de autorización y autenticación predeterminados para las aplicaciones Web. Es muy sencillo quitar, modificar o reemplazar estos esquemas, dependiendo de las necesidades de la aplicación.

▪ Los valores de configuración de ASP.NET se guardan en archivos basados en XML, que los usuarios pueden leer y escribir. Cada una de las aplicaciones puede tener un archivo de configuración distinto y el esquema de configuración se puede extender como sea necesario.

3.3.6 Elementos de las aplicaciones Web ASP.NET. La creación de aplicaciones Web ASP.NET implica trabajar con muchos de los elementos que se utilizan en cualquier aplicación de escritorio o cliente-servidor. Éstos incluyen:

- **Funciones de administración de proyectos.** Al crear una aplicación Web ASP.NET, necesitará hacer un seguimiento de los archivos necesarios, cuáles se debe compilar y cuáles necesitan implementarse.

- **Interfaz de usuario.** Una aplicación suele presentar información a los usuarios; en una aplicación Web ASP.NET, la interfaz de usuario se presenta en páginas de formularios Windows Forms, que envían los resultados a un explorador. Opcionalmente, puede crear resultado adaptado para dispositivos móviles u otros aparatos Web.

- **Componentes.** Muchas aplicaciones incluyen elementos reutilizables que contienen código para ejecutar tareas específicas. En las aplicaciones Web, puede

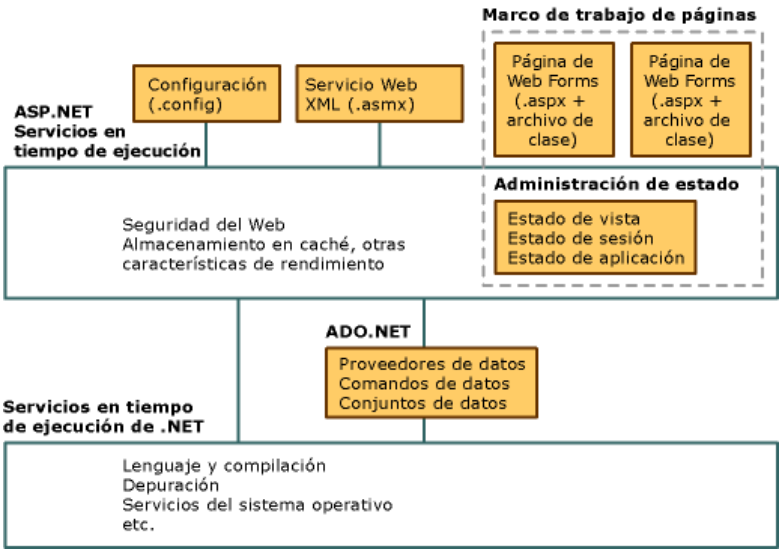
crear estos componentes como servicios Web XML, lo que les permite aceptar llamadas de todo el Web desde una aplicación Web, otro servicio Web XML o un formulario Windows Forms, por ejemplo.

- **Datos.** La mayoría de las aplicaciones necesitan algún mecanismo de acceso a datos. En las aplicaciones Web ASP.NET, puede utilizar ADO.NET, los servicios de datos que forman parte de .NET Framework.

- **Seguridad, rendimiento y otras características de infraestructura.** Como en cualquier aplicación, deberá implementar seguridad para evitar el uso no autorizado, probar y depurar la aplicación, ajustar su rendimiento y ejecutar otras tareas no relacionadas con la función principal de la aplicación.

El diagrama siguiente proporciona información general sobre el modo en que encajan entre sí las diferentes piezas de las aplicaciones Web ASP.NET y sobre cómo encajan en el contexto más amplio de .NET Framework.

Figura 15. Estructura de las aplicaciones Web ASP .NET



3.3.7 Componentes basados en Web: servicios Web XML. Un servicio Web XML es un componente al que otras aplicaciones pueden llamar a través de una red TCP/IP. Ejecuta una función específica (cualquier cosa desde cálculos y validación de tarjetas de crédito hasta procesamiento de pedidos complejos) y devuelve valores a la aplicación que hace la llamada.

Lo que hace únicos a los servicios Web XML es que se les puede llamar a través del Web. Los servicios Web XML se invocan mediante peticiones HTTP o SOAP e intercambian datos con otros componentes mediante XML. Como tales, pueden convertirse en parte integral de aplicaciones Web ASP.NET y proporcionar servicios no sólo a las aplicaciones, sino también a cualquier aplicación que disponga de acceso Web, por lo que resultan ideales para las transacciones de negocio a negocio.

3.3.8 Acceso a datos en aplicaciones Web. La mayoría de las aplicaciones Web ASP.NET implican al menos un cierto nivel de acceso a datos. ASP.NET no incluye directamente posibilidades de acceso a datos. En su lugar, las aplicaciones Web utilizan servicios de datos ADO.NET.

ADO.NET proporciona un marco de trabajo completo para obtener acceso a datos de distintos orígenes y administrarlos, incluidas las bases de datos y los archivos XML o las secuencias. ADO.NET incluye proveedores: clases que permiten conectar con orígenes de datos, ejecutar comandos y leer resultados. De modo opcional, puede conservar datos en un conjunto de datos, que es una caché en memoria desconectada.

El acceso a datos en las aplicaciones Web, ya sea en una página de formularios Web Forms o en un servicio Web XML, supone desafíos especiales:

- **Dinamismo.** Los componentes de las aplicaciones Web no suelen conservar un estado, lo que hace que no sea práctico mantener conexiones activas con un origen de datos (u otros recursos).

- **Escalabilidad.** Dado que las aplicaciones Web pueden tener cargas de usuario que varían sustancialmente en cortos periodos de tiempo, el acceso a los datos debe diseñarse teniendo en cuenta la escalabilidad.

Visual Studio proporciona diversas herramientas para trabajar con datos, incluida la compatibilidad del Cuadro de herramientas con los elementos de datos, diferentes asistentes para configuración. Los temas siguientes proporcionan información acerca de datos en general (es decir, de ADO.NET), así como del uso de datos en aplicaciones Web.

3.3.9 Infraestructura de las aplicaciones Web: seguridad, rendimiento y otros aspectos. Además de facilitar el medio para crear elementos de interfaz de usuario y componentes a los que se puede llamar, ASP.NET proporciona un contexto para ejecutar esos elementos. Por ejemplo, ASP.NET se comunica con IIS para controlar solicitudes de páginas de formularios Web Forms y servicios Web XML, analizar los archivos y llamar a componentes relacionados.

Gran parte de este trabajo tiene lugar bajo la superficie, en un nivel en el que habitualmente no es necesario programar durante la creación de aplicaciones de escritorio.

Visual Studio proporciona una compatibilidad limitada para trabajar con las características de infraestructura de ASP.NET. Por ejemplo, puede editar el archivo de configuración de la aplicación Web (Web.config) mediante el Editor de código de Visual Studio. No obstante, .NET Framework es tanto conectable como extensible y ofrece acceso de bajo nivel, si es necesario.

Hay otros aspectos de las aplicaciones Web con los que deberá enfrentarse que forman parte de la infraestructura de la aplicación. Éstos incluyen:

- **Seguridad.** Con frecuencia, deberá autenticar y autorizar a los usuarios de la aplicación Web. Hay problemas especiales asociados con la seguridad en una aplicación Web, porque los usuarios tienen acceso a recursos basados en servidor y porque se tiene muy poco control sobre el lado cliente de la aplicación (el explorador o el dispositivo portátil). ASP.NET incluye funciones de seguridad que pueden configurarse y programarse en la aplicación Web.

- **Rendimiento y optimización.** Para ajustar el rendimiento de la aplicación, puede utilizar memoria caché de páginas y datos. ASP.NET mantiene una caché de resultados que almacena páginas pedidas anteriormente; al especificar configuraciones de caché, es posible controlar cuánto tiempo se mantienen las páginas en la memoria caché y bajo qué circunstancias se actualizan.

- **Seguimiento.** Dado que las aplicaciones Web se ejecutan en el servidor, a menudo un servidor remoto, no tienen otro resultado que el resultado de la aplicación (una página de formularios Web Forms, por ejemplo). ASP.NET, en consecuencia, ofrece la oportunidad de incluir información de seguimiento directamente en una página de formularios Web Forms.

3.4 SERVIDORES 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.

Un servidor de Base de datos es por tanto el equipo en el que corre un sistema de gestión de bases de datos. Durante la realización de la práctica se brindo apoyo al diseño y construcción de aplicaciones distribuidas en entorno Web con el motor de base de datos de Microsoft: SQL SERVER 2005 y se brindó soporte a otras aplicaciones existentes con diversos gestores de bases de datos entre los que se cuentan ORACLE, DBASE, Microsoft Access 97. En esta sección describiremos a SQL Server y Oracle debido a que son los SGBD comerciales más utilizados en la actualidad.

3.4.1 SQL Server 2005. SQL Server 2005 es un motor de bases de datos relacionales de procesamiento de transacciones en línea (OLTP); este tipo de bases de datos son óptimas para administrar datos que cambian. Suelen tener varios usuarios que realizan transacciones al mismo tiempo que cambian los datos en tiempo real. Aunque las solicitudes de datos realizadas individualmente por los usuarios suelen hacer referencia a pocos registros, muchas de estas solicitudes se producen al mismo tiempo.

Las bases de datos OLTP están diseñadas para permitir que las aplicaciones transaccionales escriban sólo los datos necesarios para controlar una sola transacción lo antes posible. Las bases de datos OLTP se caracterizan en general por lo siguiente:

- Admiten el acceso simultáneo de muchos usuarios que agregan y modifican datos con regularidad.

- Representan el estado en cambio constante de una organización, pero no guardan su historial.

- Contienen muchos datos, incluidos todos los datos utilizados para comprobar transacciones.

- Tienen estructuras complejas.

- Se ajustan para dar respuesta a la actividad transaccional.

- Proporcionan la infraestructura tecnológica necesaria para admitir las operaciones diarias de la empresa.

- Las transacciones individuales se completan rápidamente y se tiene acceso a cantidades de datos relativamente pequeñas. Los sistemas OLTP están diseñados y ajustados para procesar cientos o miles de transacciones que se indican al mismo tiempo.

Estas y otras características como la inclusión de un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente, la integración directa con Microsoft Visual Studio, nuevas herramientas de desarrollo como el generador de reportes, hicieron que este completo SGBD fuera seleccionado como repositorio de datos de la aplicación desarrollada. Sin embargo para presentar un panorama completo se enumeran a continuación las ventajas y desventajas de escoger un gestor de base de datos comercial como este.

- **Ventajas.**

1. Facilidad de manejo de grandes volumes 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.

- **Inconvenientes.**

1. El costo de actualización del hardware y software son muy elevados.
2. Costo (salario) del administrador de la base de datos es costoso.
3. El mal diseño de esta puede originar problemas a futuro.
4. Un mal adiestramiento a los usuarios puede originar problemas a futuro.
5. Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
6. Generan campos vacíos en exceso.
7. El mal diseño de seguridad genera problemas en esta.

- **Componentes.**

SQL Server 2005 contiene las siguientes tecnologías:

- **Motor de Base de Datos.** El Motor de base de datos de SQL Server 2005 de Microsoft es el servicio principal para almacenar, procesar y asegurar datos. El Motor de base de datos proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa. El Motor de base de datos también proporciona compatibilidad completa para mantener una alta disponibilidad.

- **Servidor de reportes.** Microsoft SQL Server 2005 Reporting Services es una solución basada en servidor que se utiliza para generar informes empresariales que extraen contenido de una variedad de orígenes de datos relacionales y multidimensionales, que publica informes que se pueden ver en diversos formatos, y que administra la seguridad y las suscripciones de manera centralizada. Los informes creados se pueden ver mediante una conexión Web o como parte de una aplicación de Microsoft Windows.

Reporting Services incluye herramientas y asistentes gráficos para crear y publicar informes y modelos de informes; herramientas de administración del servidor de informes para administrar Reporting Services; e interfaces de programación de aplicaciones (API) para programar y extender el modelo de objetos de Reporting Services.

- **Servicios de integración y análisis.** Microsoft SQL Server 2005 Integration Services (SSIS) es una plataforma que permite generar soluciones de integración de datos de alto rendimiento, entre las que se incluyen paquetes de extracción, transformación y carga (ETL) para el almacenamiento de datos.

Integration Services incluye herramientas y asistentes gráficos para generar y depurar paquetes; tareas para realizar funciones relacionadas con el flujo de trabajo, por ejemplo operaciones de FTP, la ejecución de instrucciones SQL y mensajería de correo electrónico; orígenes y destinos de datos para extraer y cargar datos; transformaciones para limpiar, agregar, mezclar y copiar datos; un servicio de administración, denominado servicio de Integration Services, para administrar Integration Services; e interfaces de programación de aplicaciones (API) para programar el modelo de objetos de Integration Services. Se compone de cuatro partes clave: el servicio, el modelo de objetos de, el tiempo de ejecución y los ejecutables de tiempo de ejecución de, y la tarea Flujo de datos que encapsula el motor de flujo de datos y los componentes de flujo de datos.

- **Réplica de SQL Server.** La réplica es un conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de base de datos desde una base de datos a otra, para luego sincronizar ambas bases de datos y mantener su coherencia. La réplica permite distribuir datos a diferentes ubicaciones y a usuarios remotos o móviles mediante redes locales y de área extensa, conexiones de acceso telefónico, conexiones inalámbricas e Internet.

3.4.2 ORACLE. Una BD Oracle es un conjunto de datos organizados según el modelo relacional.

Cada servidor de Oracle está constituido por una BD y una instancia.

- Base de Datos: Lugar donde se almacenan los datos
- Instancia: constituye el mecanismo que permite su manipulación.

La base de datos de Oracle tiene una capa lógica y otra física. La capa física consiste de archivos que residen en el disco y los componentes de la capa lógica son estructuras que mapean los datos hacia estos componentes físicos.

- **La Capa Física.** Ya se dijo que consiste de archivos físicos que se encuentran en los discos. Estos pueden ser de tres tipos diferentes:

- Uno o más datafiles
- Los datafiles almacenan toda la información ingresada en una base de datos. Se pueden tener sólo uno o cientos de ellos. Muchos objetos (tablas, índices) pueden compartir varios datafiles. El número máximo de datafiles que pueden ser configurados está limitado por el parámetro de sistema MAXDATAFILES.

- Dos o más archivos redo log (de deshacer)

- Los archivos del tipo redo log almacenan información que se utiliza para la recuperación de una base de datos en caso de falla. Estos archivos almacenan la historia de cambios efectuados sobre la base de datos y son particularmente útiles cuando se necesita corroborar si los cambios que la base de datos ya ha confirmado se han efectuado realmente en los datafiles.

- Uno o más control files

- Estos archivos contienen información que se utiliza cuando se levanta una instancia, tal como la información de dónde se encuentran ubicados los datafiles y los archivos redo log. Estos archivos de control deben encontrarse siempre protegidos.

- **La Capa Lógica.** La capa lógica de una base de datos consta de los siguientes elementos:

Uno o más tablespaces
El esquema de la base de datos (schema), el cual consiste de objetos como tablas, clusters, índices, vistas, procedimientos almacenados, triggers, secuencias y otros.

Una instancia es el conjunto de estructuras de memoria (Área Global del Sistema-SGA por sus siglas en inglés) y procesos en ejecutándose en segundo plano:

Procesos de usuario: Ejecutan el código de una aplicación.
Procesos de Oracle: Atienden a los procesos de usuario y realizan el mantenimiento de la BD.

- **Estructura de la Base de Datos.** Ficheros de datos y espacios de tablas. Oracle almacena lógicamente los datos en unas estructuras llamadas tablespaces, las cuales se almacenan físicamente en datafiles (ficheros de datos).

- BD se componen de uno o más tablespaces. Cada Tablespace consiste de uno o más ficheros de datos.

- Cada fichero de datos no puede contener más de un Tablespace.

- Oracle cuenta con un tablespace especial llamado SYSTEM creado automáticamente durante el proceso de instalación. Utilizado para la propia gestión de la BD.

- Una BD puede estar constituida únicamente por un tablespace SYSTEM: recomendable crear al menos un tablespace adicional.

- **Tareas del administrador de la BD.**

- Controlar el espacio de disco reservado para los datos.
- Añadir datafiles a los tablespaces.
- Asignar cuotas de espacio a los usuarios.
- Realizar copias de seguridad o recuperaciones parciales de la BD.
- Los tablespaces constituyen la 'ventana' a través de la cual los usuarios y diseñadores de la BD ven los datos almacenados en los datafiles. Administrador encargado de mantener las relaciones entre tablespaces y datafiles.

- **Objetos.** Un objeto Oracle es un elemento creado y almacenado en la BD (en los tablespaces). Ejemplos: tablas, vistas, sinónimos, índices, secuencias y clusters.

- **Tablas.**

- Unidad básica de almacenamiento de datos.
- Consta de un número fijo de columnas que describen los atributos de la entidad que representa la tabla.
- Cada columna es de un tipo de datos y se identifica por un nombre.
- Sobre la tabla se pueden imponer restricciones. Tipos de restricciones:
- Clave primaria (PRIMARY KEY)
- Valor nulo no admitido (NOT NULL)
- Columna exclusiva (UNIQUE)
- Valor por omisión (DEFAULT).
- Clave ajena (FOREIGN KEY).

- **Esquema.**
- Es el conjunto de objetos que posee una cuenta.
- Para referirnos a un objeto determinado deberemos indicar a que esquema pertenece: nom_esquema.nom_objeto.

- ***Vistas.***
- Una vista es básicamente un subconjunto de las columnas y/o filas de una tabla (u otras vistas).
- Se define como una consulta y es tratada como una tabla.
- Una vista no almacena datos, sólo se almacena la consulta que la define.

- **Secuencias.**

  - Cada secuencia genera una serie única de números.
  - Útil en la generación única de claves.
  - Pueden ser cíclicas o crecer hasta un valor máximo

- **Sinónimos.**

  - Identificador alternativo para denotar un objeto.
  - Se utilizan para: enmascarar el nombre y propietario de un objeto, dar transparencia a objetos remotos de BD distribuidas y simplificar sentencias SQL.

- **Índices.**

  - Proporcionan un acceso más rápido a los datos.
  - Una vez creados son mantenidos por Oracle y utilizados para la recuperación de datos.
  - Se pueden crear hasta un máximo de 32 columnas.
  - Implementación de índices mediante B+ trees que por ser árboles balanceados igualan el tiempo de acceso a cualquier fila.

- **Clusters.**

  - Agrupamiento de tablas que se almacenan juntas físicamente.
  - Ventajas:
  - Se reduce el acceso a disco cuando están involucradas esas tablas
  - Las columnas comunes se almacenan una sola vez.

- **Procedimientos, funciones, paquetes.**

  - Funciones y procedimientos son bloques de sentencias PL/SQL que se almacenan en el diccionario de datos.

  - Se pueden agrupar procedimientos y funciones en paquetes.

- **Disparadores.**

  - Procedimientos que se ejecutan cuando se produce un evento en la BD.
  - Se utilizan para aumentar la integridad referencial, conseguir mayor seguridad o mejorar las opciones de auditoria.

- **Enlaces de BD.**

  - Sirven para especificar una vía de acceso a un objeto situado en una BD remota.

4. DESARROLLO DE LA PRÁCTICA

El presente capítulo describe las actividades realizadas durante la práctica que contribuyeron a la consecución de los objetivos propuestos en el plan de trabajo. Es de resaltar que las labores desempeñadas estaban acordes al cargo designado dentro de la organización y no se encontraban ligadas al resultado del proyecto desarrollado sino al apoyo en los procesos de su diseño y construcción.

4.1 DESCRIPCIÓN DE LA APLICACIÓN DESARROLLADA

El proyecto principal asignado entre otros de menor envergadura consiste en el diseño y construcción de un sistema de información en ambiente Web para el manejo de la venta y ocupación de servicios funerarios para un parque cementerio.

Los módulos a desarrollar son:

- **Facturación.** Grabación y expedición de facturas por los diversos conceptos que generen ingreso a la empresa.[15]

- **Cartera.** Los servicios brindados por la empresa cliente son de un alto costo de adquisición lo que hace necesario el manejo y control de la financiación de los mismos; lo que implica la gestión de cobro, recaudo y en general aspectos propios del funcionamiento de una cartera.

[15] Entiéndase como Empresa a la organización objeto del desarrollo del proyecto.

- **Inventario.** Debido a la naturaleza de los productos brindados por el parque cementerio y sus servicios asociados se requiere un manejo especial de la venta y ocupación de los mismos. Así como la conservación de la información histórica de estos productos y su integración con el resto de los módulos del sistema ya que actualmente se cuenta únicamente con los archivos físicos de la misma.

- **Comisiones Nomina.** La comercialización de los servicios de la empresa se hace a través de asesores los cuales tienen participación en las ventas mediante comisiones; lo que obliga a un registro y control apropiado de las mismas empalmando esta información con el sistema de Nómina implantado por la empresa.

La herramienta seleccionada para este desarrollo fue establecida en las condiciones de la empresa cliente basada en criterios propios de lo que consideraron eran requisitos fundamentales en el nuevo sistema: confiabilidad, escalabilidad y tecnología de vanguardia. En su concepto estas características las cumple la plataforma .NET de Microsoft y su herramienta Visual Studio 2005; Además es de resaltar que los fuentes de este software son propiedad de la empresa cliente lo que restringe la exposición de sus componentes, pantallas, estructura de datos y en general cualquier producto obtenido durante el tiempo de la práctica.

4.2 ANÁLISIS Y DISEÑO DE LA APLICACIÓN

4.2.1 Estudio y refinación de los requerimientos del proyecto. En la primera fase de la práctica se hizo un estudio de los documentos resultado de la etapa de análisis de requisitos del sistema con el fin de familiarizarse con los alcances y funcionalidad definidos para el proyecto.

En el transcurso de esta etapa se estudiaron los mapas de procesos actuales del cliente concentrándose en detallar aquellos que servirían posteriormente para establecer las reglas del negocio (segunda capa); igualmente se analizaron y documentaron los requisitos funcionales y no funcionales detallando los necesarios para la construcción del primer prototipo, así como el diseño preliminar del modelo de datos.

En este punto los requisitos de Hardware y Software al igual que la arquitectura del sistema se encuentran en proceso de definición considerando que la incursión de ISL (empresa donde se desarrolló la práctica) en el manejo de las herramientas .NET es reciente y la adopción de las mismas demandó un proceso de adaptación de los procedimientos y estándares vigentes a la nueva ideología y arquitectura establecida por las mismas.

4.2.2 Esquematización y descripción de las reglas del negocio (Segunda Capa). Luego de la apropiación y refinación de los requisitos se esbozaron los procesos fundamentales del sistema, estableciendo las relaciones entre ellos y la interacción con el usuario; en esta etapa se comenzó la documentación del diseño y la preparación del documento final para ser entregado al cliente. Este documento debía contener, entre otros, los siguientes aspectos:

- Modelo actualizado del negocio.
- Árbol de menús.
- Diseño de reportes y pantallas.
- Diseño detallado de almacenamiento de datos.
- Especificaciones de comunicaciones.
- Especificaciones de respaldo y recuperación.
- Requerimientos seleccionados y diseño básico del primer prototipo.
- Especificaciones de programación.
- Plan de contingencia.

- Plan de pruebas del software.[16]

De esta manera se presentan al cliente los primeros bosquejos sobre la apariencia del sistema, la combinación de colores, el diseño de pantallas y menús. En este punto ya se había seleccionado un nombre para el software (STigia – Sistema de Tradicionales) y en una presentación posterior del proyecto ante la administración de la empresa cliente se explicaría el significado del mismo.

4.2.3 Diseño, documentación y construcción de la base de datos del sistema. Se diseña y construye la primera versión de la base de datos, manteniendo las versiones preliminares de la misma como evidencia de desarrollo; se instala y configura el motor de base de datos seleccionado y se estipulan las primeras directivas de seguridad tales como:

- Definición y creación de usuarios de la base.
- Configuración del servidor para acceso remoto.
- Publicación de la estructura de datos.

Esto último se hizo con la intención de mantener una comunicación permanente entre los programadores y el DBA con el fin de establecer una retroalimentación y facilitar las tareas de construcción. Todo cambio en la estructura de la base era informado oportunamente y cualquier sugerencia tal como la inclusión de un nuevo campo o el cambio de tipo de dato era recibida y estudiada en conjunto con la dirección del proyecto.

4.2.4 Planeación y Organización del Proyecto. La etapa de diseño se centró, entre otros, en la definición de cuatro aspectos importantes:

[16] Tomado de **PTPS01 PROCESO PRODUCCIÓN DE SOFTWARE**. Documento Interno. ISL.

- **Definición y optimización de la arquitectura de la aplicación.** Una de las falencias de la arquitectura planteada (Modelo 3 capas, Arquitectura Web) es que el volumen de datos y los procedimientos que deben ser ejecutados multiplican considerablemente las peticiones al servidor, la carga de los formularios supera el tiempo de espera al que un usuario de este tipo de sistemas está acostumbrado lo que hace la aplicación, a los ojos del usuario, lenta y aburrida. Dentro del entorno de trabajo de Visual Studio 2005 recientemente se ha desarrollado un conjunto de componentes asociados bajo el nombre de “Atlas” que optimiza las aplicaciones Web en tanto reduce las operaciones de ida y vuelta al servidor reduciendo las peticiones al mínimo y aumentando considerablemente la velocidad de renderización. El principal objetivo de esta tecnología auxiliar es que el usuario no se quede observando una pantalla en blanco mientras ejecuta alguna consulta ó espera la respuesta a un evento por parte del formulario.

- **Manejo de herramientas de diseño y auxiliares.** Inicialmente se estudio la posibilidad de utilizar SQL Server 2005 y su herramienta de generación de reportes debido a la conveniencia con respecto a costos de licenciamiento ya que el cliente adquirió esta herramienta completamente licenciada y la inversión fue bastante alta; Sin embargo esta utilidad presenta restricciones de diseño e impresión en cuanto al desconocimiento general de la misma por parte del equipo de desarrollo. En consecuencia se optó por una herramienta en la que la empresa desarrolladora y su equipo de ingenieros tenían un mayor conocimiento y experiencia como es Cristal Reports 10, la cual viene incorporada en la versión empresarial de Visual Studio 2005.

- **Directivas de seguridad y definición de roles y usuarios.** ASP .NET maneja automáticamente los procesos de autenticación y permisos de usuario, sin embargo además de la autenticación por formularios existe un requerimiento del cliente que exige autenticación de red ó mediante el directorio activo del servidor lo que obliga a manejar variables de entorno y a incorporar métodos más

confiables de control y auditoria de procesos. Acorde a estos linimientos se definieron los siguientes roles de usuario:

- **Administrador General.** Este rol tiene la potestad de realizar cualquiera de los procesos implementados dentro del sistema. Esta encargado de la alimentación de las tablas básicas de la aplicación así como de la creación de usuarios y asignación de permisos. Por último es el encargado de realizar la auditoria a los procesos y verificar el correcto funcionamiento del software.

- **Administrador del módulo.** El administrador de cada módulo tiene permisos garantizados para todos los procesos de su área; así pues el Jefe del departamento de Cartera fungiría como administrador de la misma dentro del sistema con derechos de modificación y consulta total, ejecución de procesos especiales, alimentación y mantenimiento de tablas básicas.

- **Usuario del módulo.** Este rol permite que un usuario consulte, modifique y ejecute procesos propios de su área.

- **Usuario Consulta por módulo.** Este rol permite que un usuario cuyo rol sea perteneciente a un área diferente consulte reportes y resultados de procesos generados en otro módulo.

- **Usuario Visitante.** Usuario al que no se le permite realizar procesos dentro de ningún módulo pero puede hacer uso de algunas utilidades del sistema y consultar su información. Ejemplo: Los asesores pueden consultar las modalidades y tasas de financiación vigentes pero no pueden grabar ni modificar datos de ningún tipo.

Existirán procesos especiales que pueden ser asignados a un usuario en particular. Así pues un solo usuario puede tener varios roles dentro del sistema. Además en el momento es que sea retirado del Active Directory (Directorio activo

de usuarios de la red) por parte del administrador de la Red, aún cuando se conserva la información relacionada con este ya no podrá acceder al sistema.

- **Administración de la base de datos.** Entre las funciones a realizar por el administrador de la base de datos o DBA se encuentran: mantener actualizada la documentación del modelo de datos y corregir o incorporar nuevas estructuras a medida que se van necesitando en la programación; por ejemplo la creación de vistas y procedimientos almacenados que faciliten y optimicen los procesos de programación de la aplicación. Otras tareas de importancia que corresponden con frecuencia realizar a un DBA:

➢ Analizar datos y efectuar recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.

➢ Apoyar en el diseño y optimización de modelos de datos.

➢ Asistir a los desarrolladores con sus conocimientos de SQL y de construcción de procedimientos almacenados y *triggers*, entre otros.

➢ Apoyar en la definición de estándares de diseño y nomenclatura de objetos.

➢ Documentar y mantener un registro periódico de las mantenciones, actualizaciones de hardware y software, cambios en las aplicaciones y, en general, todos aquellos eventos relacionados con cambios en el entorno de utilización de una base de datos.

Este último rol fue desarrollado durante el tiempo restante de la práctica convirtiéndose en una de las principales actividades de la misma.

Debido a la incursión reciente de la empresa en la tecnología .NET, paralelamente a la fase de análisis y diseño se desarrollaron actividades de capacitación y consulta sobre las herramientas y su uso. Para esto fue necesario planificar un programa de trabajo interno que contemplara espacios de tiempo para la asimilación de los diversos conceptos de arquitectura y diseño de la plataforma de trabajo de Visual Studio para ser aplicados en la construcción de la aplicación en desarrollo.

4.3. CONSTRUCCIÓN DE LA APLICACIÓN

En esta etapa se tomaron los requerimientos seleccionados en la etapa de diseño para construir el primer módulo de la aplicación y puesta en marcha del mismo en ambiente de prueba. Dado que el módulo de inventario presentaba las características propicias para ser implementado en el primer prototipo debido a que la mayoría de sus procesos son independientes, no se tiene un precedente sobre su funcionamiento y toda la información concerniente al mismo debe ser registrada y migrada a la base de datos.

4.3.1 Construcción del Primer Prototipo. Se crearon las clases para cada una de las entidades que conforman el módulo y se establecieron sus relaciones mediante el diseñador de clases de VS2005 determinando sus propiedades, métodos y las relaciones de herencia entre otros aspectos.

Se determinó la división de las funcionalidades de la aplicación en el entorno de desarrollo mediante la selección de los equipos que cumplirían los roles de Servidor de Aplicaciones, Servidor de base de datos y Repositorio de archivos del proyecto. El servidor de aplicaciones se encarga de brindar los servicios de acceso a la aplicación desde cualquier punto de la red local mediante el protocolo

TCP/IP. Para esto se configuró dentro de los servicios de IIS (Internet Information Server) una carpeta virtual y los parámetros de funcionamiento de ASP .NET para lo cual tuvo que ser instalado el Marco de trabajo de Microsoft: NET Framework en su versión 2.0.

El servidor de base de datos contiene la base de datos y gestiona el acceso de la aplicación a los mismos. En este equipo debe instalarse SQL Server 2005 y, al igual que en el servidor de aplicaciones, el Framework 2.0 este equipo cuenta además con la herramienta: SQL Server Management Studio, la cual permite administrar y configurar todos los servicios de SQL Server además de tareas de creación, respaldo y recuperación de datos. El servidor de la empresa desarrolladora sirvió para mantener centralizados los archivos del proyecto facilitando el trabajo en equipo y su conservación; ya que el servidor tiene un estricto plan de mantenimiento entre los cuales se cuenta con un Backup semanal que garantiza que ante cualquier eventualidad se puede tener una última versión del proyecto.

La construcción de este módulo estuvo acompañada de entregas parciales y de trabajo conjunto con el departamento de sistemas del cliente con el fin de apoyar la migración de la información de la base y la verificación de la consistencia de la misma. En este punto se llevo a cabo la instalación y configuración del servidor de base de datos en el cliente. Igualmente se instaló y configuro el servidor de aplicaciones provisional del sistema con el fin de pasar a la siguiente etapa: la implementación del módulo de inventario en ambiente de prueba.

4.3.2 Ejecución y documentación del plan de pruebas del modulo de Inventario. Durante la implementación de este módulo se llevó a cabo el plan de pruebas definido en la fase de análisis y se hizo recopilación de las impresiones

de los usuarios así como de los nuevos requerimientos que con respecto a las funcionalidades del prototipo surgieron durante esta etapa.

Adicionalmente se presentaron cambios internos dentro de la organización cliente, más específicamente cambios en la administración y el enfoque de negocio que se venía manejando resultando en la aparición de nuevos requerimientos funcionales que fueron estudiados e incorporados en la siguiente etapa.

4.3.3 Ajuste del plan de trabajo, análisis en incorporación de nuevos requerimientos. Con el cambio de administración la dirección del proyecto se vio en la necesidad de replantear el cronograma de trabajo con el fin de ser coherentes con el funcionamiento del sistema; fue por esta razón que se determinó invertir el orden de entrega de los módulos de Facturación y Cartera de tal manera que el equipo de desarrollo se volcó en pleno en la elaboración del módulo de Facturación.

Paralelamente se realizaron los cambios solicitados por el cliente en el módulo de Inventario y se modificó el diseño de almacenamiento de datos para que contemplara los nuevos requerimientos hechos por la administración.

Dentro del nuevo enfoque del proyecto expresado por la administración se encontraba la inclusión de un módulo de *Gestión Comercial* que apoyara los procesos de la dirección comercial de la empresa. El análisis y recopilación de requisitos de este nuevo módulo significó una gran dedicación por parte del equipo de desarrollo ya que representaba un evidente retraso al tener que volver a ejecutar procesos de la primera fase del proyecto, lo que desvió la atención del desarrollo y retrasó ostensiblemente el cronograma de trabajo.

4.3.4 Diseño y construcción del módulo de Facturación. Dentro de este módulo se estuvo a cargo de la construcción de la clase “Cliente” la cual contempló los procesos de grabación y modificación de clientes, el registro de la información auxiliar (Ej. Referencias personales del cliente). La definición de los procesos de grabación de Negocio. (Orden de pedido, proceso anterior al registro de la venta), diseño y construcción de tablas básicas del módulo. Entre otras actividades que apoyaron la tarea de documentación de procesos y sentaron los lineamientos de implementación y puesta en marcha del módulo en ambiente de prueba.

Esta etapa presentó otras novedades como la incorporación de un nuevo elemento al grupo desarrollador ante el inesperado retiro de un miembro del equipo. Lo que obligó a una fase de acoplamiento ya que tuvo que capacitarse al nuevo individuo en la información del proyecto y ponerlo al corriente de la metodología de trabajo y el estado actual del desarrollo.

4.3.5 Apoyo a la realización de la documentación técnica y del usuario de la aplicación. Paralelamente a este proceso se documentaron los procesos de los módulos construidos hasta el momento para su incorporación en el manual de usuario final y como parte de la documentación técnica del proyecto.

Estos documentos fueron creados en base a los estándares de la empresa desarrolladora y acorde a los lineamientos que sobre el particular define su sistema de gestión de calidad. Sin embargo estas versiones son solo borradores y deben ser completadas por las personas que lleguen a ocupar el lugar del estudiante en práctica.

En este punto se estaba llegando al final de la etapa de ejecución de la práctica y se comenzó con las actividades de documentación y preparación de la entrega del cargo lo que lleva a que la última parte del cronograma planteado para la práctica

sea ejecutado casi en su totalidad a excepción de la construcción del módulo de facturación, la cual no fue finalizada debido a los múltiples retrasos surgidos durante el tiempo de realización del proyecto y los cuales estaban completamente fuera del control del estudiante en práctica.

4.3.6 Diseño y Construcción del módulo de Cartera. Como se explicaba en el numeral anterior, debido a circunstancias ajenas al control del estudiante en práctica no se completó la construcción de este modulo. Sin embargo y en concordancia con el objetivo número 2 del proyecto, se presenta las actividades ejecutadas en diferentes etapas de la práctica encaminadas a soportar o apoyar la construcción del módulo de Cartera:

- **Definición y descripción de procesos críticos del departamento de Cartera.** En diversas reuniones realizadas con los encargados del área de cartera del cliente se revisaron y diseñaron los reportes que deberán ser generados por el sistema para este modulo, incluyendo el replanteamiento de algunos y la descripción de los nuevos.

Igualmente se realizaron ejercicios prácticos de procesos como la refinanciación de deudas y devoluciones los cuales serán las bases para la sistematización de estos procesos y de otros similares.

- **Clarificación de conceptos y replanteamiento de proceso.** Dentro del desarrollo de este tipo de sistemas de información se encuentra el estudio de los procesos y el planteamiento de mejoras a los mismos, ante lo cual se recomendó la eliminación de algunos procesos que generaban sobrecostos innecesarios al cliente y manejo excesivo de papelería; de la misma manera se replantearon algunos procesos y la inclusión de algunos nuevos. (Ejemplo: la inclusión de una funcionalidad para el cliente que le permita consultar la proyección de sus pagos según el producto o servicio que desee adquirir – Cotización)

5. DESCRIPCIÓN DE OTRAS ACTIVIDADES Y RESPONSABILIDADES A CARGO

Como ya se ha dicho en capítulos anteriores la vinculación del estudiante en práctica se hizo mediante la figura de contrato laboral, lo que significa que se llega a cumplir con las actividades propias de un cargo además de las planificadas para el proyecto. Siguiendo este precepto este capítulo está dedicado a la descripción de estas actividades, igualmente importantes, ejecutadas durante los seis meses de trabajo bajo esta modalidad.

5.1. ACTIVIDADES DE SOPORTE

5.1.1 Apoyo al soporte a usuarios de los sistemas existentes dentro de la organización. Cada auxiliar de sistemas (cargo asignado) tenía dos clases de obligaciones o roles: desarrollo de aplicaciones y soporte a usuarios. ISL como empresa desarrolladora de software con posicionamiento regional y nacional ofrece a sus clientes aplicaciones de diversos tipos y brinda servicios de mantenimiento y capacitación en el manejo de estas aplicaciones.

REQUEST anglicismo que significa “Solicitud” es una aplicación para el registro y control de peticiones propias de un sistema de gestión de calidad. Este sistema creado originalmente para el apoyo al proceso propio de certificación de la empresa se convirtió rápidamente en un software genérico aplicable a cualquier tipo de empresa. Esta aplicación realizada en la herramienta Visual Basic 6.0 admite dos motores de base de datos diferentes: Access y Oracle. El número de

clientes de este producto viene en ascenso constituyéndose así la necesidad de hacer nuevas instalaciones, brindar capacitación sobre su funcionamiento así como detectar y corregir posibles errores.

5.2 ACTIVIDADES DE HELP DESK

5.2.1 Recepción de solicitudes y problemas reportados por los usuarios. Si bien se estaba a cargo de una sola aplicación, dentro de los lineamientos del sistema de gestión de calidad se establece que la recepción de peticiones, quejas ó reclamos expresadas por los clientes debe ser atendida y registrada por cualquier persona en la empresa que se encuentre disponible para atender dicha comunicación, para luego ser redirigida al encargado del proceso correspondiente o en su defecto a la administración de la empresa.

5.2.2 Diagnostico preeliminar de problemas y registro de seguimiento a la solución de los mismos. ISL como empresa certificada, establece procedimientos para el registro, mantenimiento y solución de solicitudes tanto internas como externas con miras al mejoramiento continuo de los procesos mediante la medición de los tiempos de respuesta e indicadores de gestión. Para lo cual se apoya en dos productos propios destinados al manejo de estas solicitudes: Request SGC, el cual maneja las solicitudes relacionadas con el sistema de gestión de calidad interno y Request Soporte, el cual maneja las solicitudes técnicas de los clientes de la empresa y sirve a la vez para medir el rendimiento de los auxiliares mediante los indicadores de tiempos de solución de las solicitudes que tengan a cargo. Toda solicitud, exceptuando las solicitudes de instalación y capacitación, debe ser registrada en ambos sistemas y hacer el correspondiente seguimiento de las actividades a cargo mediante los mismos.

5.3 ACTIVIDADES DE CALIDAD

5.3.1 Apoyo al mejoramiento continuo de la organización desde los procesos a cargo. Para alcanzar este objetivo debe participarse de las actividades que para tal fin planifica la empresa; igualmente es necesario acatar los procedimientos establecidos en el sistema de gestión de calidad para realizar cada uno de los procesos que se tengan a cargo. Además se tiene que estar dispuesto a la autoevaluación y a ser proactivo desde su cargo de tal manera que siempre se presenten propuestas que conlleven a la mejora de los procesos de los cuales se es responsable o del ambiente de trabajo en general.

6. CONCLUSIONES Y RECOMENDACIONES GENERALES

Gracias a las actividades realizadas durante la práctica, se ampliaron los conocimientos en el desarrollo de productos software mediante el estudio y aplicación de los procedimientos estipulados por la empresa para tal fin.

El uso de herramientas de desarrollo licenciadas supone un reto para los Ingenieros recién egresados ya que estas evolucionan más rápidamente gracias a toda la infraestructura tecnológica y de talento humano que la soporta. El manejo de las mismas posee la ventaja de tener una gran cantidad de literatura (la mayoría en inglés), comunidades de desarrollo y posibilidades de capacitación (tutoriales y Webcasts).

La adaptación de la empresa a las nuevas tecnologías demanda además la reestructuración de las metodologías utilizadas y la división del trabajo. Se recomienda evolucionar de la metodología de prototipado evolutivo a paradigmas un poco más complejos que permitan detallar en mayor grado cada uno de los componentes de las aplicaciones a desarrollar (Por ejemplo RUP).

Igualmente se recomienda la revisión y documentación detallada de la manera adecuada de instalar y configurar Visual Studio 2005 y SQL Server 2005, con el fin de asegurar el correcto funcionamiento de las mismas y explotar al máximo sus capacidades.

Por último se considera importante brindarle continuidad a este tipo de proyectos ya que explotan al máximo las cualidades de los estudiantes próximos a graduarse ya que evalúan diversas destrezas más allá de los conocimientos académicos

como la iniciativa, la capacidad de resolución de problemas, el trabajo en equipo, manejo de personal, atención al cliente, entre otras; que permiten complementar la formación académica recibida durante el transcurso de su carrera.

7. BIBLIOGRAFÍA

CATALDI, Zulma. Metodología de diseño, desarrollo y evaluación de software educativo. Tesis de Magíster en Informática. (Versión resumida) Facultad de Informática. UNLP. 2000.
www.unex.es/didactica/RELATEC/Relatec_2_1/cataldi_lage_2_1.pdf

CASTRO GIL, Robin Alberto. Estructura básica del proceso unificado de desarrollo de software. Universidad ICESI. 2004

GÓMEZ FLOREZ, Luís Carlos. Planeación de Proyectos Informáticos. Grupo de Investigación en Sistemas y Tecnología de la Información STI. Bucaramanga, 2001

LETELIER, Patricio y **PENADÉS,** Carmen. Metodologías ágiles para el desarrollo de software: Extreme Programming (XP). Universidad Politécnica de Valencia.
http://www.willydev.net/descargas/masyxp.pdf

MENDOZA SÁNCHEZ, María A. Metodologías de Desarrollo Software
http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

VARAS C. Marcela. Gestión de Proyectos de Desarrollo de Software. 2000.
http://www.inf.udec.cl/~mvaras/gpis/apunteGPDS.pdf

Descargas, documentación y noticias importantes sobre el paquete de herramientas Visual Studio .NET 2005.
http://www.microsoft.com/spanish/msdn/vs2005/default.mspx

SQL Server 2005. Instalación y configuración del servidor de base de datos, documentación y utilidades.
http://www.microsoft.com/latam/sql/

ANEXOS

ANEXO A. REQUISITOS DE HARDWARE DE VISUAL STUDIO

Durante la práctica se hizo uso de la versión ***Profesional*** de Visual Studio .NET. El equipo en el que instale esta edición de Visual Studio debe cumplir los siguientes requisitos del sistema.

Procesador: Procesador de 600 MHz. Se recomienda: procesador de 1 Gigahercio (GHz);

Memoria RAM: Mínimo 192 MB. Se recomiendan: 256 MB; El rendimiento no se ha optimizado para la configuración mínima del sistema. Si aumenta la cantidad de memoria RAM por encima de la configuración mínima recomendada para el sistema mejorará el rendimiento; especialmente si ejecuta varias aplicaciones simultáneamente, si trabaja en proyectos de gran tamaño o si desarrolla aplicaciones empresariales.

Espacio Disponible en disco:

Sin MSDN:

- Se requiere 1 GB de espacio disponible en la unidad del sistema
- Se requiere 2 GB de espacio disponible en la unidad de instalación

Con MSDN:

- Se requiere 1 GB de espacio disponible en la unidad del sistema
- Se requieren 3,8 GB de espacio disponible en la unidad de instalación con una instalación completa de MSDN

Se requieren 2,8 GB de espacio disponible en la unidad de instalación con una instalación predeterminada de MSDN. Cuando se inicia el instalador de Visual Studio, la ubicación predeterminada de la instalación es la unidad del sistema, que es la unidad que inicia el sistema. Sin embargo, puede instalar la aplicación en cualquier unidad. Independientemente de la ubicación de la aplicación, el proceso de instalación instala algunos archivos en la unidad del sistema. En consecuencia, compruebe que dispone de la cantidad de espacio indicada anteriormente en la unidad del sistema independientemente de la ubicación de la aplicación, y verifique que dispone de espacio adicional, tal como se indicó anteriormente, en la unidad en la que instala la aplicación.

Sistema operativo: Service Pack 4 de Windows® 2000, Service Pack 2 de Windows XP, Service Pack 1 de Windows Server 2003 o versiones posteriores; Windows XP Home no es compatible con el desarrollo de aplicaciones Web locales, que sólo se puede realizar en las versiones Professional o Server de Windows. Microsoft Windows 2000 Datacenter Server no es un sistema operativo compatible.

Vídeo: 800 X 600, 256 colores. Se recomienda: 1024 X 768, color de alta densidad de 16 bits

Unidad de CD-ROM o DVD-ROM: El tipo de medio proporcionado con el producto determinará si será necesario un CD-ROM o un DVD-ROM.

ANEXO B. DISEÑADOR DE CLASES DE VISUAL STUDIO .NET[17]

Introducción

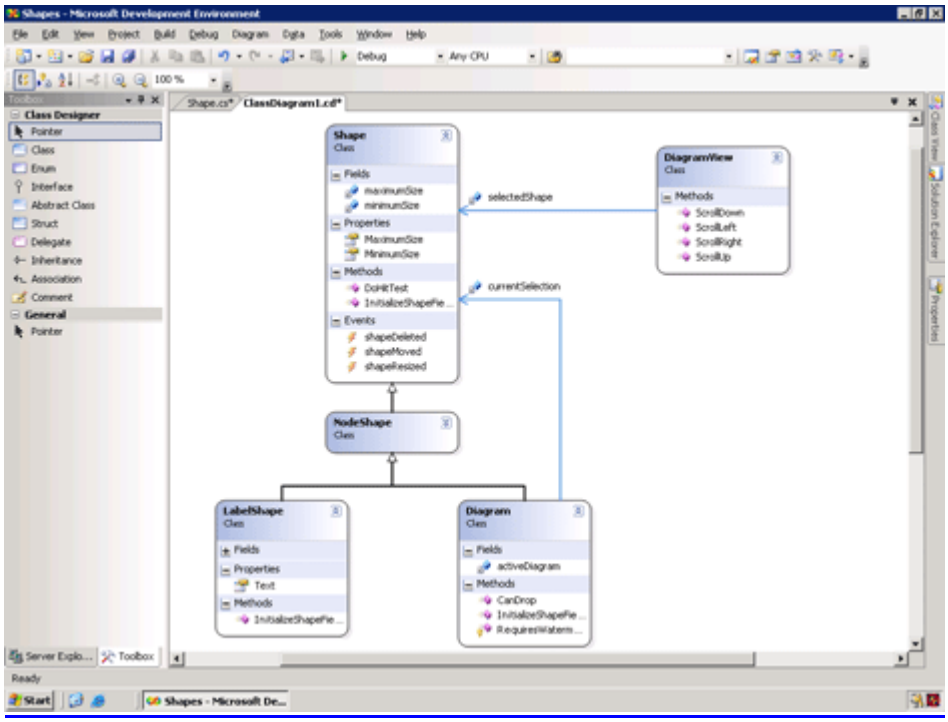
El diseñador de clases de Visual Studio es un entorno de diseño visual completamente funcional para Common Language Runtime. Este diseñador permite visualizar la estructura de un conjunto determinado de clases u otro tipo de elementos y, a través de estas representaciones visuales, editar su código fuente. Los cambios realizados en el diagrama de clases se verán reflejados inmediatamente en el código, y los cambios de éste afectarán de forma inmediata al aspecto del diseñador. Esta relación sincrónica entre el diseñador y el código facilita la creación y configuración visual de tipos CLR complejos.

El diseñador de clases contiene una serie de características diseñadas especialmente para facilitar la refactorización de código, el cambio de nombre de identificadores y la sobrescritura de métodos. Se pueden generar automáticamente clases y estructuras, e implementar interfaces a través de la generación automática de códigos auxiliares.

Por último, el diseñador de clases también actúa como herramienta de comunicación, al permitir la fácil comunicación entre áreas específicas de su base de código y sus colegas. Los diagramas de clases se pueden imprimir en papel o guardar como imágenes para visualizarlas posteriormente en páginas HTML o en presentaciones de PowerPoint.

[17] http://www.microsoft.com/spanish/msdn/articulos/archivo/050804/voices/clssdsgnr.asp

El diseñador de clases



Razones para utilizar un diseñador de clases visual. El diseño de software es una tarea ardua y compleja. Es necesario afrontar numerosos retos durante todo el ciclo de desarrollo, desde las fases tempranas de diseño y las revisiones de código, hasta la documentación del producto final. Un diseñador de clases visual puede ser una herramienta de gran utilidad durante todo el ciclo de desarrollo, como se muestra en los escenarios siguientes:

- *Comprensión del código existente:* Las bases de código existente pueden resultar complicadas y confusas. El uso de un diseñador de clases visual permite explorar gráficamente las jerarquías de clases existentes y comprender las relaciones establecidas entre las mismas.

- *Diseño de clases:* Un diseñador de clases visual permite crear gráficamente el diseño y la implementación de alto nivel del software.

- *Revisión y refactorización de código:* Un diseñador de clases visual constituye una herramienta eficaz para llevar a cabo revisiones y refactorizaciones. El diseñador permite anotar los diagramas de código existente para su revisión y facilita la refactorización de código, lo que conlleva un ahorro de tiempo considerable.

- *Diagramas de clases para documentación:* Los diagramas de clases se pueden utilizar para documentar las jerarquías de clases existentes. Los árboles de jerarquía se pueden visualizar gráficamente. Los diagramas de clases también resultan útiles para la comunicación de ideas entre colegas, a través de correo electrónico o presentaciones visuales.

Diseñador de clases de Visual Studio. El diseñador de clases de Visual Studio es una herramienta de diseño de código visual integrada en .NET Framework. La experiencia visual del diseñador de clases está estrechamente relacionada con la de Common Language Runtime. Las formas CLR, como clases, estructuras e interfaces, están representadas por formas visualmente distintas, las cuales indican su identidad. Además, la terminología del diagrama es específica del lenguaje. Por ejemplo, en Visual Basic, puede trabajar con los niveles de acceso Public, Private y Friend, mientras que en C#, estos niveles aparecerán como public, private e internal. La estrecha relación que existe entre el diseñador de clases y CLR convierte al primero en una herramienta ideal para la autorización de clases a través de .NET Framework.

El diseñador de clases de Visual Studio desempeña una función relevante durante todo el ciclo de desarrollo, proporcionando funcionalidad para todos los escenarios clave descritos anteriormente. Por ejemplo:

- **Comprensión del código existente:** El uso del diseñador de clases de Visual Studio permite determinar rápida y fácilmente las relaciones existentes entre las clases. No sólo puede analizar la jerarquía de herencia del código existente, sino también los tipos a los que se hace referencia y los conjuntos .NET, lo que permite explorar visualmente los tipos existentes y familiarizarse con ellos.

- **Diseño de clases:** El diseñador de clases de Visual Studio facilita el diseño rápido de clases y jerarquías de éstas. Utilizando la funcionalidad común de arrastrar y colocar, se pueden crear diagramas de clases al tiempo que se mantiene la sincronización con el editor de código. Los cambios realizados en el diagrama de clases se verán inmediatamente reflejados en el código y viceversa. El diagrama de clases siempre muestra una vista activa del código.

- **Revisión y refactorización de código:** El diseñador de clases de Visual Studio constituye una herramienta eficaz para llevar a cabo revisiones y refactorizaciones. Se pueden agregar comentarios a los diagramas de código existente para realizar acciones posteriores. Asimismo, las características de refactorización integradas permiten realizar tareas triviales de forma rápida y sin apenas esfuerzo, como el cambio de nombre de un símbolo o la encapsulación de campos en propiedades.

- **Diagramas de clases para documentación:** Los diagramas de clases existentes se pueden visualizar de varias formas: se pueden imprimir o guardar como imágenes para su visualización en páginas HTML o en presentaciones de Microsoft PowerPoint.

Creación de clases con el diseñador de clases. El diseñador de clases facilita la creación y configuración de clases en proyectos. El diagrama de clases es en realidad una vista activa del código. Los cambios realizados en el diagrama se sincronizan automáticamente con el código y viceversa. Puede crear una clase simple arrastrándola desde el cuadro de herramientas hasta la superficie de diseño de clases. Una vez creada en el proyecto, puede abrir el editor de código y agregar código directamente a las clases nuevas. Los cambios realizados se verán reflejados en el diagrama de clases.

Tras crear una clase, puede agregar miembros utilizando la ventana de detalles de clase. Para agregar un método, por ejemplo, haga clic en la opción de <adición de métodos> en la ventana de detalles de clase y escriba el nombre del método. A continuación, puede indicar el tipo de devolución, nivel de acceso y agregar comentarios sobre el método. Una vez creado el método, puede agregar parámetros (al nombre del método del mismo modo que se agregan métodos) indicando, en primer lugar, el nombre del parámetro y, a continuación, el tipo, el modificador y los comentarios que desee. Las propiedades, los campos y los eventos se agregan del mismo modo que los métodos. La edición de métodos a través del control de árbol es muy similar a la escritura en un editor de código (las mismas pulsaciones de tecla permiten explorar las celdas, y la ayuda de IntelliSense está disponible).

Implementación de una interfaz. El diseñador de clases facilita la implementación de interfaces en las clases. De hecho, si la interfaz se visualiza en la superficie del diseñador de clases, se podrá implementar utilizando el mismo procedimiento empleado para heredar una clase: trazando una línea de herencia desde la clase a la interfaz. No obstante, la implementación de la interfaz también resulta fácil si ésta no se visualiza en el diseñador de clases. Basta con arrastrar la interfaz desde la vista de clase hasta la clase que desea implementar. Se generarán códigos auxiliares para los métodos definidos en la interfaz. Una vez

implementada, puede agregar el código de implementación específico en el editor de código.

Visualización de una jerarquía de herencia. El diseñador de clases se puede utilizar también para visualizar las jerarquías de herencia de un proyecto determinado. Para mostrar la clase base de una clase heredada, haga clic con el botón secundario del mouse en el área de encabezado de la clase y, a continuación, haga clic en la opción de visualización de clase base. La clase base aparecerá en el diagrama.

Para mostrar clases que heredan de una clase existente, haga clic con el botón secundario del mouse en el área de encabezado de la clase y, a continuación, seleccione la opción de visualización de tipos derivados. Las clases derivadas aparecerán en el diagrama, conectadas a la clase a través de una línea de herencia.

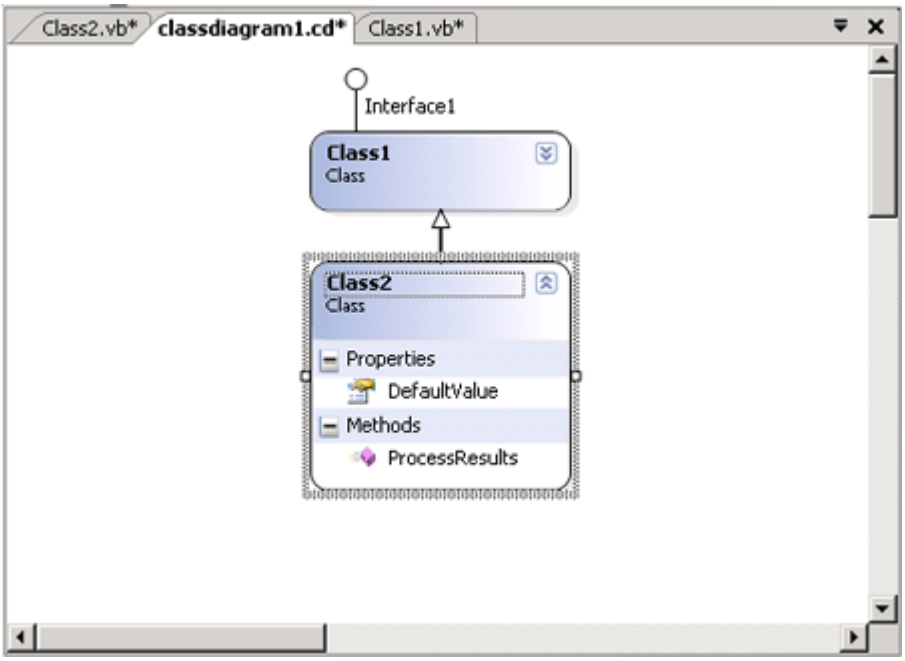
Integración con Visual Studio. El diseñador de clases se integra perfectamente con Visual Studio 2005 Team System. Puede trabajar con el diseñador de clase en el mismo espacio y del mismo modo en el que utilizaría las herramientas habituales de Visual Studio. Al trabajar con el diseñador de clases, las herramientas del diseñador poblarán el cuadro de herramientas, a las que se puede obtener acceso arrastrando y colocando. Por su parte, la ventanas de detalles de clase y Propiedades proporcionan acceso a los miembros de tipo. La acción de arrastrar y colocar también está disponible en las ventanas de herramientas estándar de Visual Studio, como Explorador de soluciones o Vista de clases. Todos los cambios realizados en el diseñador de clases se reflejan inmediatamente en los archivos de código correspondientes. Estos archivos de código se pueden abrir y editar a continuación de forma regular utilizando el editor de código de Visual Studio.

El diagrama de clases. El diagrama de clases es una vista activa del código que se actualiza continuamente con el fin de mostrar los últimos cambios realizados. En el diagrama aparecen las clases de un proyecto existente. El uso del diagrama de clases permite visualizar las relaciones existentes entre las clases de un proyecto, así como editar y agregar miembros a clases individuales. El archivo del diagrama de clases forma parte del proyecto, se conserva, y se puede utilizar siempre para ver el código de forma gráfica.

Puede agregar un diagrama de clases nuevo a un proyecto existente. Para ello, seleccione Nuevo elemento en el menú Proyecto y elija el diagrama de clases. El diagrama de clases no es una parte que depende del proyecto, sino que es una herramienta que ayuda en el proceso de creación y edición de clases. Una vez creadas, puede agregar clases existentes al diseñador de clases arrastrándolas desde la vista de clases a la superficie de diseño. También puede agregar clases nuevas y otros tipos arrastrándolos desde el cuadro de herramientas al diseñador de clases.

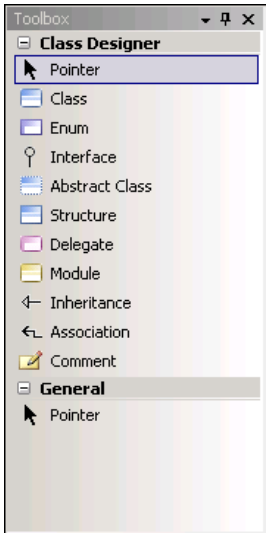
Una vez agregadas al diseñador de clases, las clases y otros tipos estarán representadas por una serie de formas, las cuales se pueden seleccionar y manipular. Al seleccionar una forma en el diseñador de clases, sus detalles aparecerán en la ventana de detalles de clase. El diagrama de clases sólo almacena información visual, no contiene información sobre el contenido del código. La eliminación del archivo de diagrama de clases no conlleva la pérdida de código.

El diagrama de clases



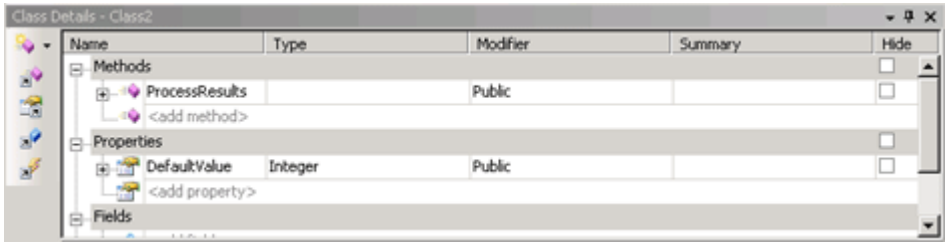
Cuadro de herramientas. Al visualizar un diagrama de clases, podrá agregarle miembros desde el cuadro de herramientas. Éste contiene clases, estructuras, delegados, enums y otros tipos que se pueden agregar al diagrama de clases. Al agregar un tipo desde el cuadro de herramientas, se agregarán al proyecto el código y los archivos de código adecuados.

El cuadro de herramientas



La ventana de detalles de clase. En la ventana de detalles de clase aparecen los métodos, las propiedades, los campos y los eventos del tipo visualizado en el diseñador de clases. Esta ventana permite editar rápidamente los miembros de una clase o estructura en el diseñador de clases. Por ejemplo, se puede cambiar el tipo de devolución de un método, agregar parámetros o cambiar el nivel de acceso. Todos los cambios realizados en esta ventana quedan inmediatamente reflejados en el código.

La ventana de detalles de clase



Conclusión. El diseñador de clases es una herramienta nueva para Visual Studio 2005 que permite crear y configurar rápidamente las clases e interfaces de un proyecto determinado. En este artículo se han tratado algunos de los escenarios más básicos en los que se puede utilizar el diseñador, como la comprensión de código existente, el diseño de clases, la revisión y refactorización de código y el uso de diagramas para la documentación. Le animamos a que explore con más detenimiento las distintas capacidades del diseñador de clases y saque el máximo partido a esta nueva herramienta de desarrollo de software.

Tomado de *"Diseñador de Clases de Visual Studio 2005"* por Matthew A. Stoecker.

URL:

http://www.microsoft.com/spanish/msdn/articulos/archivo/050804/voices/clssdsgnr.asp

ANEXO C. SEGURIDAD DE APLICACIONES WEB EN TIEMPO DE DISEÑO EN VISUAL STUDIO

Cuando se trabaja en Visual Studio para crear una aplicación Web, existen requisitos de seguridad específicos pertenecientes al acceso a los recursos necesarios durante el desarrollo. Dichos requisitos son distintos de los aplicables a los usuarios de la aplicación. En tiempo de diseño, los requisitos de seguridad son:

- Obtener acceso al servidor de trabajo, al servidor de bases de datos y a otros recursos que forman parte de la aplicación.

- El método de acceso a Web elegido (la forma de transmitir los datos al servidor Web).

- confianza.

- Depurar.

- Implementar la aplicación.

Acceso a los recursos de desarrollo. Para poder crear y probar aplicaciones Web en Visual Studio, deberá tener acceso a un equipo que ejecute los Servicios de Internet Information Server (IIS). Se debe disponer en dicho servidor (servidor de desarrollo, para diferenciarlo del servidor de producción, en el que se implementará la aplicación) de unos privilegios mínimos, incluida la capacidad de escribir archivos en el servidor y ejecutarlos.

El servidor Web no tiene por qué estar situado en el mismo equipo que se utiliza para el desarrollo. Sin embargo, en el servidor Web se deben instalar como mínimo los componentes de servidor de Visual Studio, para que admita la depuración e implementación de la aplicación.

El acceso a los recursos en tiempo de diseño se suele gestionar mediante autenticación de Windows NT; es decir, el programador utiliza las credenciales de inicio de sesión para obtener acceso a los recursos necesarios. Una vez instalado Visual Studio, se crea un grupo en el servidor, denominado VS Developers (Programadores de VS). Este grupo dispone de todos los permisos de archivo, de uso compartido e IIS necesarios para desarrollar proyectos Web en ese equipo El administrador del equipo puede ajustar la configuración de los permisos del grupo, pero no es necesario. Como programador, deberá formar parte del grupo, ya sea como individuo o como parte de otro grupo al que pertenezca. Los derechos específicos que se conceden al grupo Programadores de VS son:

- Acceso al directorio wwwroot del servidor Web.
- Permisos para crear, modificar y ejecutar archivos en directorios Web.
- Permiso para depurar aplicaciones Web remotas.

Nota de seguridad. Los programadores no necesitan derechos administrativos en un servidor para programar en él. Sin embargo, todos los desarrolladores deben ser miembros del grupo Programadores de VS. La concesión de derechos administrativos innecesarios a los programadores constituye un riesgo para la seguridad de la red.

Si la aplicación necesita obtener acceso a recursos adicionales, deberá establecer de forma similar derechos de acceso a los mismos, ya sea individualmente o como parte de un grupo. Un ejemplo típico es un SQL Server. Como programador de aplicaciones Web, necesitará poder leer y quizá actualizar las tablas de base de

datos que la aplicación necesita. En ciertas situaciones, también se necesitará permiso para escribir procedimientos almacenados en el servidor; en otras, quizá se necesite permiso para agregar, modificar o quitar tablas.

Métodos de acceso al Web. Visual Studio permite obtener acceso a los proyectos de aplicación Web de dos formas:

- Mediante acceso por compartir archivos (acceso UNC), en el que Visual Studio copia archivos al servidor mediante comandos de administración de archivos basados en Windows.

- Acceso mediante las extensiones de servidor de FrontPage, en las que los archivos se transfieren mediante HTTP.

El acceso por compartir archivos exige estar en el mismo dominio que el servidor Web. En términos prácticos, significa que el acceso por compartir archivos sólo puede utilizarse en la red propia.

En cambio, el acceso de FrontPage permite crear y gestionar la aplicación a través de un servidor de seguridad (mientras el servidor de seguridad pase solicitudes HTTP). Esto hace posible trabajar a través de Internet (o en una red local, si se prefiere acceso HTTP).

Ambos métodos de acceso aún necesitan que el servidor se configure de la forma apropiada con los componentes de servidor de Visual Studio, el grupo VS Developers. Análogamente, aún se necesitan los privilegios suficientes en el servidor para poder crear, escribir y ejecutar archivos.

Ejecutar código con plena confianza. Cuando se está ejecutando Visual Studio, el código de usuario que el diseñador ejecuta en tiempo de diseño se ejecutará con plena confianza. Esto es así, aun cuando el código se implemente en un entorno con una seguridad más restrictiva. Las implicaciones son dos:

Es posible exponer el equipo local a un riesgo de seguridad mediante la importación de código no seguro a un proyecto. Puesto que Visual Studio se ejecuta en modo de plena confianza, el código importado hereda sus permisos de Visual Studio, y el código no seguro se ejecutará como si fuera seguro. Esto sólo representa un problema en el caso de que un usuario malintencionado cree una pieza de código dañina (por ejemplo, un control personalizado) que se importa y ejecuta de forma inadvertida. Por consiguiente, se debe ser especialmente cuidadoso al importar código al proyecto.

Las aplicaciones creadas en Visual Studio pueden no funcionar correctamente al implementarse, ya que el código se ejecuta en un contexto de seguridad distinto.

Depuración. La depuración requiere la capacidad de conectarse a los procesos que se ejecutan en el equipo o, si el servidor Web se encuentra en otro equipo, en un equipo remoto. La depuración requiere que ejecute la aplicación con privilegios de administrador. Si se depura en un equipo remoto, se debe disponer de dichos permisos tanto en el equipo local como en el remoto.

Cuando se instala Visual Studio, se crea un grupo denominado Usuarios del depurador, que dispone de los permisos necesarios para poder depurar. El hecho de disponer de grupos independientes para programadores y para usuarios del depurador permite que el administrador del servidor conceda los privilegios necesarios para depurar a un grupo más selecto que la totalidad de usuarios de Visual Studio.

Nota de seguridad. Los miembros del grupo Usuarios del depurador obtienen privilegios de administrador. No incluya en este grupo a nadie que no deba tener este nivel de privilegios.

Implementar la aplicación. Una vez terminada la aplicación, debe poder implementarse en el servidor de producción. Si el servidor de producción no es de su propiedad, es posible que el acceso sea mucho más limitado que en el caso del servidor de desarrollo.

Una de las formas de efectuar la implementación de una aplicación Web es mediante el uso de las herramientas de implementación de Visual Studio. Este tipo de implementación permite efectuar una instalación completa de la aplicación, incluido el registro y la configuración. Sin embargo, requiere derechos administrativos en el equipo que contiene el servidor Web.

Otra posibilidad, en el caso de que el servidor de producción tenga instaladas las extensiones de servidor de FrontPage, es utilizar el comando Copiar proyecto a través de HTTP. Este tipo de implementación proporciona menos funcionalidad, pero permite implementar a través de un servidor de seguridad. Aún se necesitan los privilegios suficientes en el servidor de destino para poder escribir archivos.

Tomado de el artículo *"Conceptos de Visual Basic y C#;* ***Seguridad de aplicaciones Web en tiempo de diseño en Visual Studio****"*

URL:
http://msdn.microsoft.com/library/SPA/vbcon/html/vbconWebApplicationSecurityAtDesignTimeInVisualStudio.asp

ANEXO D. PLAN DE MANTENIMIENTO PARA LA BASE DE DATOS STIGIA

SQL Server cuenta entre sus herramientas de gestión y administración de datos con el Asistente para planes de mantenimiento el cual puede utilizarse para ayudar a configurar las principales tareas de mantenimiento para garantizar el buen funcionamiento de la base de datos, la realización periódica de copias de seguridad en caso de error del sistema y la comprobación de incoherencias. El Asistente para planes de mantenimiento crea uno o varios trabajos del Agente SQL Server que realizan estas tareas de mantenimiento automáticamente a intervalos programados.

Algunas de las tareas de mantenimiento que pueden programarse para su ejecución automática son:

- Reorganizar los datos de las páginas de datos y de índices mediante una nueva generación de los índices con un nuevo factor de relleno. Esto garantiza que las páginas de la base de datos contienen espacio libre y una cantidad de datos distribuidos de forma equitativa. También permite un crecimiento más rápido en el futuro.

- Comprimir archivos de datos mediante la eliminación de las páginas de base de datos que estén vacías.

- Actualizar las estadísticas de los índices para asegurarse de que el optimizador de consultas dispone de información actualizada acerca de la distribución de los valores de los datos en las tablas. Esto permite al optimizador de consultas elegir el método más adecuado para obtener acceso a los datos, ya

que dispone de más información acerca de los datos almacenados en la base de datos. Aunque SQL Server actualiza periódicamente las estadísticas de los índices de forma automática, esta opción puede obligar a que se actualicen inmediatamente.

- Realizar comprobaciones de coherencia interna de los datos y de las páginas de datos de la base de datos para asegurarse de que no se han dañado debido a un problema de software o del sistema.

- Realizar copias de seguridad de la base de datos y de los archivos de registro de transacciones. Las copias de seguridad de la base de datos y del registro pueden mantenerse durante un período especificado. Esto le permite crear un historial de copias de seguridad para utilizarlo si tiene que restaurar la base de datos a una fecha anterior a la de la última copia de seguridad de la base de datos. También puede realizar copias de seguridad diferenciales.

- Ejecutar trabajos del Agente SQL Server. Esta tarea se puede utilizar para crear trabajos que realicen una serie de acciones y, también, para crear los planes de mantenimiento para ejecutar los trabajos