BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
Facultad de Ciencias de la Computación

Programación de Dispositivos Móviles usando Web Services

Tesis

Que para obtener el título de

Ingeniero en Ciencias de la Computación

Presenta:
Rebeca Maritza Sánchez Becerril

Asesor:
M. C. Rafael De la Rosa Flores

Octubre 2009

Agradecimientos

Al Padre Celestial por darme la vida, las pruebas y bendiciones que me permiten ser quien soy en este momento.

A mi madre por su apoyo, comprensión, guía, ejemplo todos estos años, por su infinito amor, y por ayudarme a que este momento llegara. Gracias mamá.

A mi padre, por su amor, esfuerzo y herencia: mi educación. Aunque no estés conmigo físicamente ahora, lo estaremos en la eternidad.

A mis hermanas Diana y Mónica por su amor, apoyo, compañía en todos estos años y los que faltan.

A mis sobrinos Diana y Santiago, por ser esa chispa de nobleza, pureza y sencillez que hace falta en el día a día.

A mi asesor y sinodales, por el tiempo dedicado, la guía y sus aportes en la finalización de este trabajo. Gracias.

A mis compañeros y amigos que de una u otra forma, me ayudaron para que pudiera lograr este objetivo. Gracias.

Índice General

Introducción

Capitulo 1 – Marco Teórico **1**
1.1 Dispositivos Móviles 1
1.1.1 ¿Qué es un Dispositivo Móvil? 1
1.1.2 Características de un Dispositivo Móvil 1
1.1.3 Clasificación de los Dispositivos Móviles 3
1.1.4 ¿Para qué sirve un dispositivo móvil? 3
1.1.5 Ventajas y Desventajas de un Dispositivo Móvil 4
1.2 Cómo programar un Dispositivo Móvil 5
1.2.1 Proceso de Programación 5
1.2.2 Diferentes dispositivos, diferentes plataformas, diferente programación 7
1.3 .NET 10
1.4 Web Services 16
1.4.1 ¿Qué es un Web Service? 16
1.4.2 ¿Para qué son y cómo se utilizan los Web Services? 16
1.4.3 Ventajas y Desventajas de usar los Web Services 17
1.4.4 Arquitectura de los Web Services 18
1.4.5 Protocolos de los Web Services 19
1.4.6 ¿Cómo funcionan los Web Services? 20
1.4.7 XML 21
1.4.7.1 Serialización 23
1.4.8 Plataformas que soportan los Web Services 23
1.5 Arquitecturas Multinivel 23
1.6 Programación Web 25
1.6.1 Internet 25

1.6.2 Características de la Programación Web 26
1.6.3 Ventajas y Desventajas de la Programación Web 26
1.7 Programación de Aplicaciones de Escritorio 27
1.7.1 ¿Qué es una Aplicación de Escritorio? 27
1.7.2 Ventajas y Desventajas de Aplicaciones de Escritorio 27
1.8 Sistemas Distribuidos 28
1.8.1 Concepto de Sistema Distribuido 28
1.8.2 Características de los Sistemas Distribuidos 28
1.8.3 Ventajas y Desventajas de los Sistemas Distribuidos 28
1.8.4 Modelo Cliente-Servidor 29
1.8.5 Aplicaciones Distribuidas Usando Dispositivos Móviles 30

Capitulo 2 – Análisis y Diseño del Sistema 34
2.1 Planteamiento del Problema 34
2.1.1 Objetivo General 35
2.1.2 Objetivos Específicos 35
2.2 Descripción General 35
2.3 Plataforma 36
2.4 Herramientas de Programación y de Producción 36
2.5 Arquitectura del Sistema 37
2.6 Diagrama de Componentes 38
2.7 Proceso de Interacción del Sistema 40
2.8 Interoperabilidad entre los Elementos 43
2.9 Diseño de Interfaz con el Usuario 45
2.10 Diseño de la Base de Datos 47
2.11 Seguridad 50
2.12 Diagrama de Clases 51

Capitulo 3 – Implementación y Pruebas al Sistema 55
3.1 Software 55
3.2 Desarrollo 57

3.2.1 Componente de Almacenamiento 57
3.2.2 Componente de Servicios 63
3.2.3 Componente de Aplicaciones 70
3.3 Pruebas al Sistema 80

Capitulo 4 – Conclusiones y Perspectivas 100
4.1 Conclusiones 100
4.2 Perspectivas 101

Apéndice A – Sistemas Operativos 102
Apéndice B – UML 107
Apéndice C – Paso de Información entre Páginas Web 112

Bibliografía 117

Introducción

La expansión de computadoras personales en nuestro entorno hace que éstas sean ya una herramienta de trabajo muy necesaria y por supuesto, de diversión.

Hoy en día hay computadoras en casi todas partes, hecho originado fundamentalmente por la bajada de los precios de unos años a nuestros días, pero cada vez las computadoras son más potentes y nos dan muchas más posibilidades, lo que hace que no sólo estén en el lugar de trabajo y en nuestro hogar, sino que nos los podamos encontrar en cualquier ámbito de nuestra vida, como pueden ser en el coche, en la lavadora, en la televisión, etc.

La tecnología está haciendo posible que se reduzcan también los tamaños de los dispositivos y que nos los podamos meter en el bolsillo y transportarlos sin dificultad alguna, como es el caso de los asistentes personales digitales (Personal Digital Assistant - PDA). Estos dispositivos son computadoras en toda la extensión de la palabra, pues disponen de capacidad de procesamiento y almacenaje de datos.

Por tanto, vemos que la evolución se centra en fabricar aparatos más pequeños, dotándoles de habilidad de comunicación y potencia de cálculo. Independientemente del tipo de dispositivo, les requerimos que nos den más utilidades que las versiones anteriores, pero eso es un problema para los desarrolladores, porque se les pide mejorar sus productos, pero en sitios más pequeños cada vez.

Hasta hace poco, la programación de estos dispositivos móviles se hacía en código máquina o en ensamblador. La razón básica era que se disponía de entornos de ejecución muy restringidos, por lo que el uso de lenguajes de programación de alto nivel era impracticable. Esto implicaba el hecho de tener que desarrollar completamente a medida, es decir, específicamente para el dispositivo, con el consiguiente esfuerzo y lentitud en los desarrollos.

La tecnología actualmente, permite que algunos lenguajes de alto nivel se puedan utilizar para la programación de estos dispositivos móviles; tomando mucha fuerza en el ámbito laboral, se encuentra la nueva tecnología .NET que ha desarrollado Microsoft, la cual se utiliza tanto en computadoras de escritorio como en computadoras de bolsillo.

Esta tesis se encuentra divida de la siguiente manera:

Capítulo 1: Marco Teórico. En este capítulo se explica la teoría general de la programación de dispositivos móviles en la actualidad, así como se detallan algunos conceptos que se utilizaron en el proyecto.

Capítulo 2: Análisis y Diseño del Sistema. En este capítulo se mostrará y explicará el diagrama de bloques del sistema, así como algunos otros diagramas en donde se mostrará la funcionalidad del sistema; así mismo, se definirán la arquitectura y proceso que tendrá el sistema.

Capítulo 3: Pruebas al Sistema. Aquí se va a definir cada uno de los módulos del sistema así como la interacción entre ellos, además que se mostrará la manera de configurar la aplicación y se verá el funcionamiento del sistema.

Capítulo 4: Conclusiones y Perspectivas. En este capítulo se plasmarán las conclusiones que se hayan obtenido de la investigación y del proyecto. También se explicarán los alcances a futuro que tienen este tipo de tecnologías y las aplicaciones sobre los mismos.

Capítulo 1

Marco Teórico

1.1 Dispositivos Móviles

1.1.1 ¿Qué es un Dispositivo Móvil?

Los dispositivos móviles o mejor conocidos en sus términos en ingles como: “information device”, “information appliance”, “embedded device” o “small device”, son aparatos pequeños con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con memoria limitada, diseñados específicamente para una función, pero que pueden llevar a cabo otras más generales.

Normalmente se asocian al uso individual, tanto en posesión como en operación, los cuales pueden adaptarse al gusto de la persona.

Una característica importante es el concepto de movilidad: los dispositivos móviles son aquellos suficientemente pequeños para ser transportados y empleados durante su transporte. Normalmente se sincronizan con un sistema de sobremesa para actualizar aplicaciones y datos por medio de una red inalámbrica, dando la impresión al usuario de que los datos están almacenados en el propio dispositivo.

1.1.2 Características de un Dispositivo Móvil

Las principales características de un dispositivo móvil son:

- Portabilidad.
- Aparatos pequeños.
- Tienen una limitada capacidad de procesamiento.
- Pueden o no tener una conexión a una red.
- Su memoria es limitada.

- Diseñados específicamente para una función, aunque se pueden añadir algunas otras.
- Normalmente son para el uso individual.

Otra característica importante es el término en inglés “Wireless” (en español, inalámbrico); un dispositivo inalámbrico es aquel que es capaz de comunicarse o acceder a una red sin cables, por ejemplo un teléfono móvil o un PDA. Este tipo de dispositivos se comportan como si estuvieran conectados a una red mediante un cable, pero su tecnología es inalámbrica.

Algunas de las características que hacen que estos dispositivos sean diferentes de las computadoras de escritorio son las siguientes:

- Tienen una funcionalidad limitada.
- No es necesariamente extensible y actualizable.
- En pocos años, el usuario deberá cambiarlo.
- Son más baratos.
- Son menos complicados en su manejo.
- Fácil de aprender su operación.
- No se requiere un usuario experto.

Algunos dispositivos que entran en la categoría son:

- Asistentes Personales Digitales (PDA, Ipod, etc)
- Teléfonos móviles inteligentes (Smartphones, Iphone, Black Berry, etc.) con sistema operativo similar al de las PDA.
- Teléfonos móviles con capacidad de ejecución de programas Java

Algunos dispositivos que no entran en la categoría son:

- Computadoras portátiles (demasiado grandes, no portables).
- Tablet PC
- Teléfonos móviles sin capacidad de ejecutar programas.
- Agendas electrónicas simples.
- Calculadoras electrónicas programables.
- Cámaras digitales, reproductores MP3, etc.

1.1.3 Clasificación de los Dispositivos Móviles

Por su definición y características de los dispositivos móviles, se pueden clasificar en 4 grupos principales, como lo muestra la figura 1.1

Figura 1.1 - Clasificación de los dispositivos móviles

1.1.4 ¿Para qué sirve un Dispositivo Móvil?

La expansión de computadoras personales en nuestro entorno hace que éstas sean ya una herramienta de trabajo muy necesaria y por supuesto, también de diversión. Pero cada vez, las computadoras son más potentes y nos dan muchas más posibilidades, lo que hace que no sólo estén en el trabajo y en nuestro hogar, sino que también nos los podamos encontrar en cualquier ámbito de nuestra vida.
Por tanto, vemos que la evolución se centra en fabricar aparatos más pequeños, dotándoles de habilidad de comunicación y potencia de cálculo. Independientemente del tipo de aparato, les requerimos que más o menos nos den las mismas prestaciones en cualquier momento.

Los dispositivos móviles utilizados actualmente, abarcan una gran cantidad de funcionalidades, entre las principales y las más usadas son las siguientes:

- *Teléfono móvil:* Su principal función es la comunicación personal, dando la opción de que se le pueda localizar en cualquier momento y lugar, por medio de la recepción y emisión de llamadas. Actualmente, los teléfonos móviles evolucionaron y ya incluyen juegos, cámara fotográfica y de video, reproductor de mp3, mensajes de texto, directorio, Internet, televisión y rastreo satelital.
- *Smartphone / Black Berry:* Su principal función es la de comunicación personal, muy similar a un teléfono móvil, pero con la diferencia de que el Smartphone y la

Black Berry, trabajan con la tecnología Microsoft y poseen características interesantes, como es el poder trabajar con ventanas tipo Windows y la lectura y edición de documentos de la paquetería Office; además de los ya mencionados, como es conexión a Internet, sincronización con equipos de escritorio, cámara fotográfica, cámara de video, mp3, videos, películas, mensajes escritos, agenda, correo de voz, directorio y juegos.

- *PDA:* El objetivo de este dispositivo es, como su nombre lo indica, un asistente personal, en donde se tienen funciones como agenda, juegos, recordatorios, videos, música, fotografías, editor de texto, sincronización con equipos de escritorio, reproductor de Mp3 y videos y conexión a Internet.
- *Pocket PC:* El objetivo de este dispositivo es similar al de PDA, solo que éste trabaja particularmente con la tecnología Microsoft, por lo que se tienen opciones de poder manejar documentos de la paquetería Office, por lo que ofrece mucha funcionalidad para el trabajo. También tienen acceso a Internet, agenda, música, videos, fotografías, correo electrónico, chat y sincronización con equipos de escritorio.
- *Iphone:* De la compañía Apple Inc., es un teléfono inteligente multimedia con conexión a internet, pantalla táctil con tecnología multi-touch y una interfaz de hardware minimalista. Ya que carece de un teclado físico, se muestra uno virtual en la pantalla. El iPhone dispone de una cámara de fotos y un reproductor de música (equivalente al del iPod) además de software para enviar y recibir mensajes de texto y mensajes de voz. También ofrece servicios de internet como leer correo electrónico, cargar páginas Web y conectividad por Wi-Fi.

1.1.5 Ventajas y Desventajas de un Dispositivo Móvil

A continuación se muestra un listado de ventajas y desventajas de los dispositivos móviles, contra los equipos de escritorio:

Ventajas de trabajar con equipos móviles:

- Al ser equipos pequeños son fáciles de transportar.
- El costo es menor.
- Actualmente un dispositivo móvil tiene muchas funciones.
- Al poder llevarlos a todos lados, ayudan en la organización personal.
- Equipados con batería, no es necesario conectarlos a la corriente eléctrica.
- Algunos de ellos, al tener rastreo satelital, brindan incluso seguridad al usuario.
- Actualmente, se pueden conectar a Internet casi en cualquier lugar.
- Son fácilmente accesibles para cuando se les necesite.

Desventajas de trabajar con equipos móviles:

- Tienen memoria limitada.
- Capacidad de procesamiento es menor.
- En pocos años se tendrá que cambiar de dispositivo.
- La batería tiene un tiempo de energía limitado.
- Algunas de sus pantallas no tienen la resolución suficiente para poder navegar completamente en Internet.
- Algunos no tienen una capacidad completa para el uso en el trabajo.
- La velocidad es menor, por lo tanto los tiempos de respuesta son mayores.

1.2 Cómo Programar un Dispositivo Móvil

1.2.1 Proceso de Programación

El desarrollo de aplicaciones destinadas a dispositivos móviles, desde el punto de vista de la ingeniería del software, no debe diferir sustancialmente de los pasos a dar cuando se construyen aplicaciones para computadoras de escritorio, sólo cambiará la última parte, la cual es: la compilación o generación de código ejecutable sobre el sistema operativo contenido en el dispositivo.

Así, podríamos establecer los siguientes pasos:

a) *Análisis de Requerimientos*

El programador deberá determinar, las necesidades que los usuarios finales tienen y los requerimientos que se les pedirá tener a la aplicación. Por ejemplo, en el caso de un análisis para una aplicación que se ejecutará en un dispositivo móvil, algunos de los requerimientos podrían ser la facilidad de uso, que se pueda ejecutar en los teléfonos móviles, PDAs y Smartphone´s, que permita una conexión a una entidad mayor para obtener datos actualizados o también que sea capaz de almacenar cierta información de manera persistente.

b) *Diseño de la Aplicación*

Es muy importante en este tipo de aplicaciones el crear programas separados por cada uno de los posibles usos que se le dé a la aplicación; de esta manera cada programa será más pequeño y se adaptará mucho mejor a las características de los

dispositivos móviles. Ya en la fase de implementación se tendrá que establecer un mecanismo que controle las diferentes aplicaciones.

En cuanto al diseño del interfaz de usuario, debemos decidir la correspondencia entre la aplicación y la pantalla. Los diseñadores en esta fase no deben considerar cómo los usuarios operarán con el dispositivo para llevar a cabo una tarea, o cómo se notificará a la aplicación las acciones del usuario, se deben concentrar sólo en el objetivo de la pantalla y en la tarea que permitirá llevar a cabo.
El almacenamiento persistente es un aspecto a tener en cuenta en nuestro diseño, la pregunta a responder es: ¿qué datos deben sobrevivir a la finalización de la aplicación y estar disponibles para la siguiente vez que se vaya a ejecutar? Otra cuestión, que no se debe plantear en esta fase sino en la de implementación es qué utilizar para realizar ese almacenamiento.

Finalmente, debemos tener en cuenta dentro del diseño aspectos relacionados con la conectividad y con la entrada / salida, ya que son puntos muy importantes que van a determinar la portabilidad de la aplicación. Por tanto, en este momento deberemos tomar decisiones en un nivel de abstracción alto, que luego se concretarán cuando determinemos claramente el tipo de dispositivo y sus características.

c) *Desarrollo de la Aplicación*

Es en esta parte en la que el desarrollador se dedica a programar la aplicación para que cumpla con los requerimientos especificados; se dedicará a escribir código, crear pantallas, adaptar controles y se harán las pruebas hasta que la aplicación esté lista.
Hay dos formas de desarrollar aplicaciones para los dispositivos móviles: desde una computadora de escritorio o sobre el propio dispositivo móvil.

a. *Computadora de Escritorio*

Esta es la forma más común a la hora de programar un dispositivo móvil. Esta técnica consiste en realizar la aplicación a través de alguna herramienta ejecutada en una PC. Estas herramientas pueden ser extensiones para los lenguajes y entornos habituales de las PC o bien entornos de desarrollo específicamente creados para generar código ejecutable sobre los dispositivos móviles.

Dado que supondría un problema tener que cargar en el dispositivo el programa en desarrollo cada vez que se quiera probarlo, conviene contar con

emuladores de los dispositivos concretos y sus sistemas operativos. En el mercado existe diversa variedad de emuladores para los diferentes dispositivos móviles; Estos emuladores cambian de acuerdo con el sistema operativo y del tipo de dispositivo móvil.

Normalmente, el proceso de las aplicaciones concluye con la generación del código de la aplicación que se instalará en dispositivo móvil. En este caso hay dos formas de hacerlo, el código generado es directamente ejecutable en la máquina o bien se genera un código intermedio que al instalarse requerirá de un programa intérprete que lo traduzca y ejecute definitivamente.

Cuando trabajamos sobre una computadora de escritorio para programar dispositivos móviles, debemos lograr pasar el código o programa ejecutable al dispositivo móvil final, esto se hace con alguna de las siguientes opciones:

- Se descargará la aplicación a través de una conexión de red, se cargará en memoria, se ejecutará la aplicación, y finalmente se eliminará cualquier traza de ésta en el dispositivo.

- Siempre que el dispositivo lo permita, se instalará físicamente.

b. En el Dispositivo Móvil

Esta forma de programar consiste en realizar la aplicación directamente en el mismo dispositivo móvil donde será ejecutado.

Debido a las limitaciones de recursos de los dispositivos móviles, los editores y traductores de esta categoría suelen ser limitados en cuanto a la facilidad para el trabajo de programación y puesta a punto.

1.2.2 Diferentes dispositivos, diferentes plataformas, diferente programación

Los lenguajes de programación y sus entornos de desarrollo asociados disponibles para crear aplicaciones para los dispositivos móviles son, esencialmente los mismos que se utilizan para las demás computadoras, no obstante ya existen algunos lenguajes específicamente diseñados para ello.

Como en todo, hay siempre una solución de compromiso entre requerimientos de cómputo, facilidad de uso y de aprendizaje, costo y calidad de los resultados que se obtienen. Al

desarrollar aplicaciones para los dispositivos móviles hay que tener en cuenta su especificidad. Los dispositivos móviles no son una PC ni una laptop.

- Tamaño de la pantalla, generalmente de 160 x 160 pixeles.
- Patrón de utilización: varias veces al día por períodos muy cortos.
- Pocas entradas. No se requiere el ingreso de grandes volúmenes de datos.
- Potencia del procesador: Normalmente de 16 a 33 MHz
- Baterías pequeñas.
- Memoria que suele llegar hasta 64 MB.

A continuación se presentan algunas de las plataformas de desarrollo más conocidas en el mercado:

- PDA TOOLBOX

  Se trata de un entorno de desarrollo gráfico rápido y fácil de usar sobre PC´s, que permite a desarrollar aplicaciones para el sistema operativo Palm OS, casi sin código. Contiene un repertorio de más de 60 componentes de interfaz como formularios, imágenes de color, botones, campos, botones de radio, ayuda en línea y otros que son usuales en las aplicaciones de los computadores de sobremesa.

- SATELLITE FORMS

  Es un Entorno de desarrollo rápido basado en Visual Basic, que genera código ejecutable para Palm OS y para Pocket PC. Incluye facilidades para garantizar la ejecución de las aplicaciones en regímenes de conexión por cable e inalámbricas.

- SCOUTBUILDER

  Es un software dinámico con el que se pueden crear aplicaciones para Palm OS de manera rápida. Contiene un lenguaje de programación basado en BASIC y facilita la importación de tablas de bases de dados ODBC ya existentes. Utiliza el recurso Intellisense para ayudar a completar líneas de código automáticamente.

- CASL

  Entorno de desarrollo visual que se aloja en un PC de sobremesa y genera código para Palm OS o Pocket PC. Permite el intercambio de datos entre el dispositivo móvil y el PC a través de los programas de conexión. El lenguaje contenido en

CASL es orientado a objetos y dirigido por eventos y permite incluir segmentos de código escritos en C.

- VISUAL STUDIO .NET

  .NET ofrece características de Internet móvil que permiten a los programadores crear una interfaz Web móvil única, compatible con una amplia gama de dispositivos como los Smartphone, Pocket PC, dispositivos de mano y localizadores (pagers). Los controles móviles del servidor generan de manera inteligente la reproducción y paginación apropiada para cada dispositivo Web, proporcionando una experiencia completa y coherente al usuario a la vez que mantiene la flexibilidad del programador.

- HS PASCAL

  HS Pascal apoya la plataforma Computing de Palm Source. El compilador produce rápidamente, pequeños y optimizados programas ejecutables directamente sobre Palm OS.

- JAVA

  Java nos da la posibilidad de escribir una vez el programa y poder ejecutarlo en cualquier tipo de plataforma sin tener que recompilarlo de nuevo. En muchos casos es la única alternativa que dispone el programador, pues varios fabricantes han optado únicamente por él para desarrollar aplicaciones.

- SUPER WABA.

  Super Waba es un lenguaje que está basado en Java. Super Waba requiere la instalación del jdk de java, es libre, tiene manejo con XML, realiza mejoras a la interfaz que ofrece Java y forma código ejecutable para cada tipo de dispositivo móvil.

Con todo esto, lo que se puede observar es que la programación de dispositivos móviles depende de sobre qué tipo de dispositivo será destinada la aplicación y de qué sistemas operativos maneje dicho dispositivo. No se puede programar un dispositivo que maneja por ejemplo, sistema operativo Palm OS con un lenguaje que está destinado para Pocket PC, aunque sus funciones y características pueden ser similares, a nivel de sistema operativo son muy distintas, por lo que las PDA con sistema operativo Palm OS deben de programarse específicamente para su sistema operativo (programación Palm), de la misma

forma el Smartphone / Black Berry (programación Windows), el teléfono móvil (programación Java) y la Pocket PC (programación Windows).

1.3 .Net

Microsoft .NET es un conjunto de tecnologías que están tomando mucha fuerza hoy en día para el desarrollo tanto de aplicaciones de escritorio, aplicaciones Web, Web Services, aplicaciones para móviles, etc.
Microsoft .NET es un conjunto de tecnologías de software de Microsoft para conectar su mundo de información, gente, sistemas y dispositivos.

La plataforma .net provee de un gran número de tecnologías, estas tecnologías se describen en los siguientes puntos.

- **El .NET Framework**

El .NET Framework está basado en un nuevo 'Lenguaje Común en Tiempo de Ejecución' (CLR). El CLR provee de un conjunto de servicios para la creación de proyectos construidos en Visual Studio .NET, sin depender del lenguaje.

Microsoft Visual Basic, Microsoft Visual C++ y otros lenguajes de programación Microsoft, han sido planeados para tomar la mayor ventaja de estos servicios. Microsoft Visual J#.NET ha sido desarrollado para los desarrolladores del lenguaje Java, que han querido construir aplicaciones y servicios usando el .NET Framework.

- **.NET My Services**

.Net my Services es un conjunto de XML Web Services. Con el .Net my Services, los usuarios reciben la información relevante que ellos necesitan, dependiendo del dispositivo que estén usando y basado en las preferencias que ellos establezcan. Usando el .NET my Services, las aplicaciones pueden comunicarse directamente usando SOAP y XML desde cualquier plataforma que soporten SOAP.

- **El .Net Manejador de Servidores**

El .Net Manejador de Servidores provee escalabilidad, manejabilidad, integración con otras organizaciones, y muchas otras características como las que describe la siguiente tabla.

SERVIDOR	DESCRIPCION
Microsoft SQL Server	Incluye funcionalidad XML, soportado por los estándares de World Wide Web Consortium (W3C), tiene la capacidad de manipular los datos XML usando Transact SQL (T-SQL), flexible y poderoso analista basado en Web, y de acceso seguro a los datos a través de la Web usando http.
Microsoft BizTalk Server	Provee una integración de aplicaciones empresariales (EAI), integración negocio a negocio y una avanzada tecnología BizTalk, para construir procesos dinámicos y aplicaciones que también pueden usarse para Internet.
Microsoft Host Integration Server	Provee la mejor forma de unirse a Internet, intranet y a las tecnologías cliente-servidor.
Microsoft Exchange Enterprice Server	Construido en una poderosa colaboración tecnológica por introducir importantes mejoras en la escalabilidad y en el rendimiento de su arquitectura.
Microsoft Application Center	Provee de herramientas de desarrollador y manejador, que son útiles para las aplicaciones Web.
Microsoft Internet Security and Acceleration Server	Provee seguridad, velocidad y una conexión a internet mejor. La seguridad en internet y la aceleración del servidor integran un firewall extensible y multicapas. Esta construido sobre las políticas de seguridad y aceleración para el trabajo de Internet.
Microsoft Commerce Server	Provee un Framework a las aplicaciones, sofisticados mecanismos de regeneración y capacidades analíticas.
Microsoft SharePoint Portal Server	Provee la capacidad de crear portales Web con manejador de documentos, búsqueda de contenidos y colaboración de equipos.
Microsoft Content Management Server	Ofrece el completo conjunto de características para su contribución y entrega, desarrollo de sitios, y manejador de sitios empresariales, desarrollo y manejo de Internet, intranet y sitios Web externos.

- **Visual Studio .NET**

Visual Studio .NET provee un ambiente de desarrollo para la construcción de aplicaciones en la plataforma .NET. Provee importantes tecnologías para simplificar la creación, desarrollo y la evolución de seguridad, escalabilidad y alta disponibilidad de aplicaciones Web y Web Services XML. [3]

El .NET Framework

El .NET Framework provee de los elementos necesarios para la compilación y ejecución para construir y desarrollar las aplicaciones basadas en .NET.

El .Net Framework esta formado por componentes (Véase la figura 1.2), los cuales se definirán a continuación.

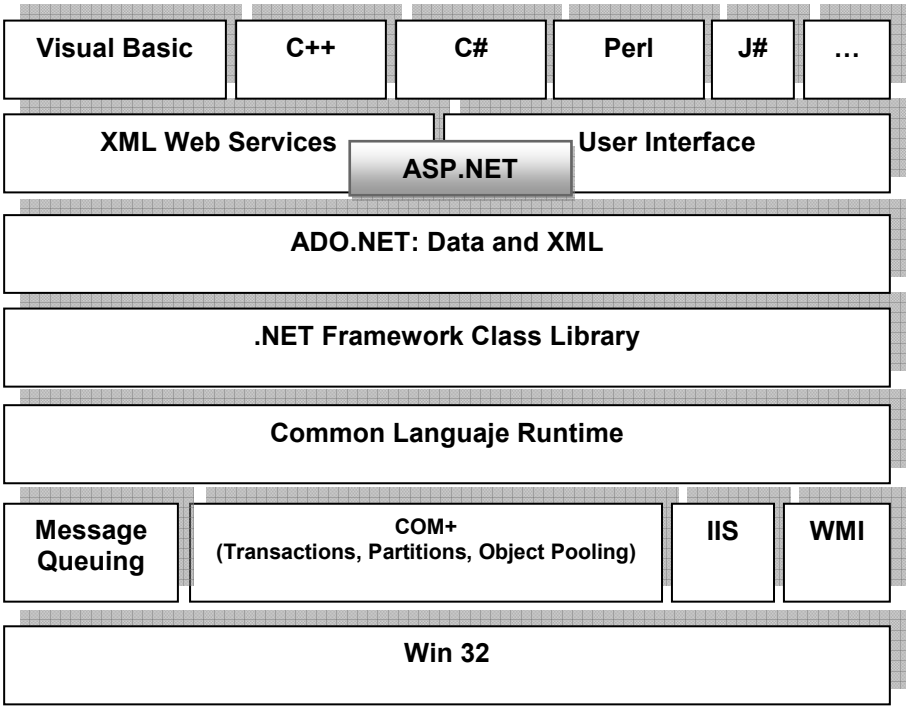

Figura 1.2 – Arquitectura de la plataforma del .NET Framework

Contenido de la Plataforma

El .Net Framework debe correr en un sistema operativo tipo Microsoft Win32. En un futuro, el .NET Framework será extendido a que pueda correr en otras plataformas, como puede ser Microsoft Windows CE. [5]

Servicios de la Aplicación

Cuando se corre en Windows, los servicios de la aplicación, como son los servicios de los componentes, la cola de mensajes, el Internet Information Server (IIS) y el Windows Management Instrumentation (WMI), están disponibles para el desarrollador. El .NET

Framework expone los servicios de la aplicación a través de las clases de la librería de clases del .NET Framework.

Common Language Runtime

El Common Language Runtime simplifica el desarrollo de la aplicación, provee un robusto y seguro entorno de ejecución, soporta múltiples lenguajes y simplifica el despliegue de la aplicación y su manejo.
El Common Language Runtime es también referido al ambiente de manejo, en el que se incluyen los servicios comunes, como son el colector de basura y la seguridad, son automáticamente proporcionados.

.NET Framework Class Library

La librería de clases del .NET Framework expone características del tiempo de ejecución y provee otros servicios que todo desarrollador necesita. Las clases simplifican el desarrollo de las aplicaciones basadas en .NET. Los desarrolladores pueden extenderlos, creando sus propias librerías de clases.

ADO.NET

ADO.NET es la próxima generación de la tecnología Microsoft ActiveX Data Objects (ADO). ADO.NET provee la ayuda mejorada para la desconexión del modelo de programación. También provee amplia ayuda de XML.

ASP.NET

Microsoft ASP.NET es un marco de programación que se construye en CLR. ASP.NET puede ser usado en un servidor para construir poderosas aplicaciones WEB.

ASP.NET Web Forms proporcionan una sencilla y poderosa forma de construir interfaces de Web dinámicas (UI).

XML Web Services

XML Web Services son componentes Web programables que pueden ser compartidos a través de las aplicaciones en internet o intranet. El .NET Framework proporciona herramientas y clases para construir, probar y distribuir los XML Web Services.

Interfaces de Usuario

El .NET Framework soporta 3 tipos de interfaces de usuario:

- Las Web Forms, que son las que trabajan con ASP.NET.
- Windows Forms, las cuales corren en las computadoras tipo Win32.
- Aplicaciones de consola.

Lenguajes

Cualquier lenguaje que se conforme con la especificación del lenguaje común (CLS), puede correr en el CLR. En .NET Framework, Microsoft proporciona Visual Basic, Visual C++, Microsoft C#, Visual J# y Microsoft JScript.

.NET Compact Framework

Microsoft .NET Compact Framework es un componente integral de los dispositivos Windows Mobile y Windows Embedded CE que permite generar y ejecutar aplicaciones administradas y utilizar Web Services. .NET Compact Framework incluye un Common Language Runtime (CLR) optimizado y un subconjunto de la biblioteca de clases de .NET Framework, que admite características como Windows Communication Foundation (WCF) y formularios Windows Forms. También contiene clases que están diseñadas exclusivamente para .NET Compact Framework.

.NET Compact Framework hereda la arquitectura .NET Framework completa de Common Language Runtime para ejecutar código administrado. Proporciona interoperabilidad con el sistema operativo Windows CE de un dispositivo para tener acceso a funciones nativas e integrar los componentes nativos favoritos en una aplicación.

Puede ejecutar aplicaciones nativas y administradas de manera simultánea. El host del dominio de aplicación, que también es una aplicación nativa, inicia una instancia del Common Language Runtime para ejecutar el código administrado. [6]

Arquitectura

En la ilustración siguiente se resume la arquitectura de la plataforma .NET Compact Framework.

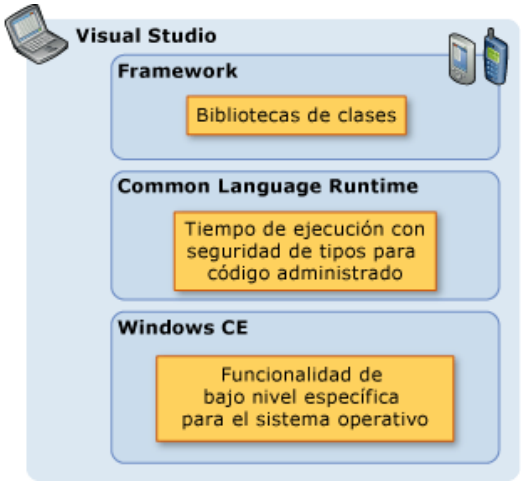

Figura 1.3 – Arquitectura del .Net Compact Framework

Windows CE

.NET Compact Framework utiliza el sistema operativo Windows CE para la funcionalidad central y para diversas características específicas de dispositivos. Varios tipos y ensamblados, como los de los formularios Windows Forms, gráficos, dibujos y Web Services, se han recompilado para que se ejecuten eficazmente en los dispositivos, en lugar de copiarse de .NET Framework completo.

.NET Compact Framework ofrece la siguiente interoperabilidad con Windows CE:

- Compatibilidad con seguridad nativa.
- Integración completa con programas de instalación nativos.
- Interoperabilidad con código nativo mediante la interoperabilidad COM y la invocación de plataformas.

Common Language Runtime

También el Common Language Runtime (CLR) de .NET Compact Framework se ha vuelto a generar para permitir que los recursos restringidos se ejecuten en memoria limitada y lograr un uso eficaz de la energía.

Entre Windows CE y el Common Language Runtime existe una capa de adaptación de plataforma, que no aparece en la ilustración, para asignar las interfaces de servicios y dispositivos necesarias para CLR y Framework a los servicios e interfaces de Windows CE.

Framework

.NET Compact Framework es un subconjunto de .NET Framework pero también contiene características diseñadas en exclusiva. Ofrece prestaciones y facilidad de uso para acercar a los desarrolladores de aplicaciones nativas para dispositivos a .NET Framework, y para acercar a quienes desarrollan aplicaciones de escritorio a los dispositivos.

1.4 Web Services

1.4.1 ¿Qué es un Web Service?

Un Web Service es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, son los elementos fundamentales en la evolución hacia la computación distribuida a través de Internet. Se están convirtiendo en la plataforma de integración de aplicaciones gracias a los estándares abiertos y al énfasis en la comunicación y colaboración entre personas y aplicaciones. Las aplicaciones se crean utilizando los diversos Web Services que funcionan conjuntamente, sin importar su ubicación o la forma en que se implementaron. [7]

La idea general de lo que son los Web Services incluye lo siguiente:

- Los Web Services ofrecen funciones muy útiles a usuarios de la Web ya que emplean un protocolo Web estándar que, casi en todos los casos, es SOAP.
- Los Web Services permiten describir sus interfaces con suficiente detalle para que el usuario diseñe una aplicación cliente que permita comunicarse con ellas. Esta descripción se proporciona normalmente en un documento XML denominado WSDL (Lenguaje de descripción de Web Services).
- Los Web Services se registran para que los futuros usuarios los encuentren fácilmente. Este registro se realiza a través de UDDI (descripción, descubrimiento e integración universales).

1.4.2 ¿Para qué son y cómo se utilizan los Web Services?

El objetivo principal de los Web Services es por medio de protocolos y estándares, poder intercambiar datos entre aplicaciones. La exposición de aplicaciones existentes como Web Services permitirá que los usuarios creen aplicaciones más potentes y que utilicen estos Web Services como elementos clave.
Los Web Services al igual que una página Web está definido por una URL (Uniform Resource Identification) y por su interfaz, a través del cual se puede acceder a él. De esta

forma, las aplicaciones se convierten en clientes que integran Web Services procedentes de diferentes proveedores, y además, se abre la posibilidad de que se cobre por uso del servicio, no por cada copia de la aplicación vendida.

La manera general de utilizar los Web Services es, creando un Web Service que brindará funcionalidades a la aplicación que estemos creando; una vez teniendo el Web Service funcionando, o si ya existe alguno en Internet y queremos utilizarlo, debemos obtener la URL que nos dirige al acceso de ese Web Service, se añade la referencia Web a nuestra aplicación y podremos utilizar sus métodos y propiedades libremente.
Para cuando deseamos hacer uso de una aplicación que utiliza un Web Service, simplemente accedemos a ella normalmente, la aplicación es la que se encargará de hacer el llamado al Web Service, para el usuario de la aplicación, el uso de los Web Services es transparente. [9]

1.4.3 Ventajas y Desventajas de utilizar los Web Services

Ventajas de los Web Services

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los Web Services fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los Web Services pueden aprovecharse de los sistemas de seguridad "firewall" sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándares.

Desventajas de los Web Services

- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI (Remote Method Invocation), CORBA o DCOM (Distributed Component Object Model). Es uno de los inconvenientes derivados de adoptar un formato basado en texto y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall, cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.
- Existe poca información de Web Services para algunos lenguajes de programación.

1.4.4 Arquitectura de los Web Services

Los Web Services se dividen en servicios de transporte, de mensajería, de descripción *y* de descubrimiento. (Véase la figura 1.4)

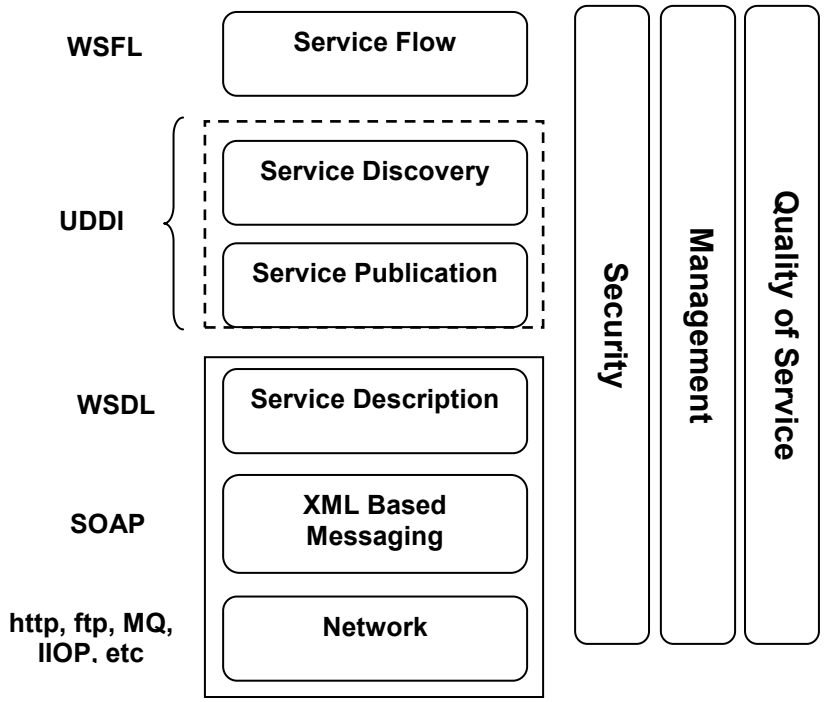

Figura 1.4 – Arquitectura de los Web Services

En la parte más baja de la figura 1.4 se encuentran los servicios de transporte, que establecen la conexión y el puerto usado. Generalmente se usa HTTP, el mismo protocolo que la WWW, pero se puede usar también SMTP (el mismo protocolo que el correo electrónico), FTP (File Transfer Protocol), o BEEP (blocks extensible exchange protocol), un protocolo específico para Web Services, que, a diferencia de los anteriores, no es cliente-servidor, sino "entre pares"; los dos ordenadores entre los que se establece la comunicación actúan como clientes y servidores a la vez. Es además extensible, y está especificado en XML; por eso se está haciendo mucho más popular para aplicaciones Web.

En el siguiente nivel superior, están los servicios de mensajería, que especifican cómo se tiene que codificar el mensaje, que contiene los datos que se intercambian, entre el cliente y el servidor. El protocolo más usado en esta capa es el SOAP; este protocolo puede usar cualquiera de los transportes anteriores, se pueden escribir clientes y servidores en

cualquier lenguaje, y usa XML como lenguaje para especificar los mensajes. XML-RPC (Remote Procedure Call mediante XML) es un método de llamada remota a procedimientos usando XML, que usa como capa de transporte HTTP; a diferencia de SOAP, que puede usar cualquiera.

Los servicios deben de especificarse, para que una aplicación sepa de forma automática qué formato usar para comunicarse con un servicio. Para ello se usa principalmente WSDL, *Web Services description language*, que permite especificar la dirección de un servicio y la interfaz que se usa para acceder a él, sea SOAP o HTML.

En el siguiente nivel superior esta UDDI, *Universal Description, Discovery, and Integration*, un protocolo que lleva WSDL un poco más allá, permitiendo no sólo describir Web Services, sino productos, la empresa y cómo está dispuesta a llevar a cabo transacciones. El Registro UDDI permite buscar negocios, servicios por categorías, y te devuelve información sobre cómo acceder a ellos.

Por último, en la capa más alta se encuentra la WSFL, *Web Services Flow Language*, se trata de un lenguaje diseñado exclusivamente para representar flujos de Web Services y de este modo facilitar la creación de estos servicios. Se trata de un lenguaje muy sencillo y basado en XML.

Dentro de WSFL sin duda los conceptos más importantes son los de actividad y proveedor de servicios. Las actividades son las tareas a realizar mientras que los proveedores de servicios son las personas /roles encargados de realizar dichas actividades. [8]

1.4.5 Protocolos de los Web Services

Existen diferentes protocolos utilizados en los Web Services, cada uno enfocado a cierta funcionalidad y objetivos; a continuación se describen algunos de estos protocolos:

- **SOAP:** Es el protocolo de comunicaciones para los Web Services. SOAP define el formato XML para mensajes. Si se tiene un fragmento creado correctamente en XML e incluido en un par de elementos SOAP, dispondrá ya de un mensaje SOAP. Existen otras partes en la especificación SOAP que describen cómo representar los datos de un programa como XML y cómo utilizar SOAP para realizar llamadas a procedimiento remoto (RPC). Estas partes opcionales de la especificación se utilizan para implementar aplicaciones estilo RPC en las que el cliente envía un mensaje SOAP que contiene una función a la que se puede llamar, además de los parámetros para pasar a la función. El servidor, por su parte, devuelve un mensaje con los resultados de la función ejecutada. [8]

La última parte opcional de la especificación SOAP define los contenidos de un mensaje HTTP que contiene un mensaje SOAP. Este enlace HTTP es importante porque casi todos los sistemas operativos actuales, y otros no tan actuales, admiten HTTP. Este enlace es opcional pero casi todas las implementaciones SOAP lo admiten porque es el único protocolo estandarizado para SOAP. Por esta razón, existe la falsa idea general de que SOAP necesita HTTP. Algunas implementaciones admiten MSMQ, series MQ, SMTP o TCP/IP. Sin embargo, la mayoría de los Web Services utilizan HTTP porque es ubicuo. Dado que HTTP es un protocolo imprescindible para el medio Web, muchas empresas disponen de infraestructuras de redes que lo admiten y personal con conocimientos y experiencia para gestionarlo. En la actualidad ya se encuentran disponibles la seguridad, el control y la infraestructura de equilibrio de cargas para HTTP.
La ubicuidad de HTTP y la sencillez de SOAP los convierte en una base perfecta para implementar Web Services que pueden llamarse desde prácticamente cualquier entorno. [8]

- **WS-Security (Web Service Security):** Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados. [8]

- **Otros Protocolos:** Los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) o SMTP (Simple Mail Transfer Protocol). [8]

1.4.6 ¿Cómo funcionan los Web Services?

Como ya se explicó un Web Service brinda funcionalidades, que pueden o no llevar el intercambio de datos, para que los podamos utilizar en nuestras aplicaciones, ya sean móviles, Web o de escritorio.
Dichos Web Services se alojan físicamente en un servidor, el cual interactuará con el servidor de base de datos que se esté utilizando; cuando se tiene configurado los puertos del servidor y las salidas del Web Service, se puede hacer un llamado a ese servicio, por medio de un navegador de internet, ingresando la dirección del servicio, lo que nos mostrará una pantalla similar a la de la figura 1.4.

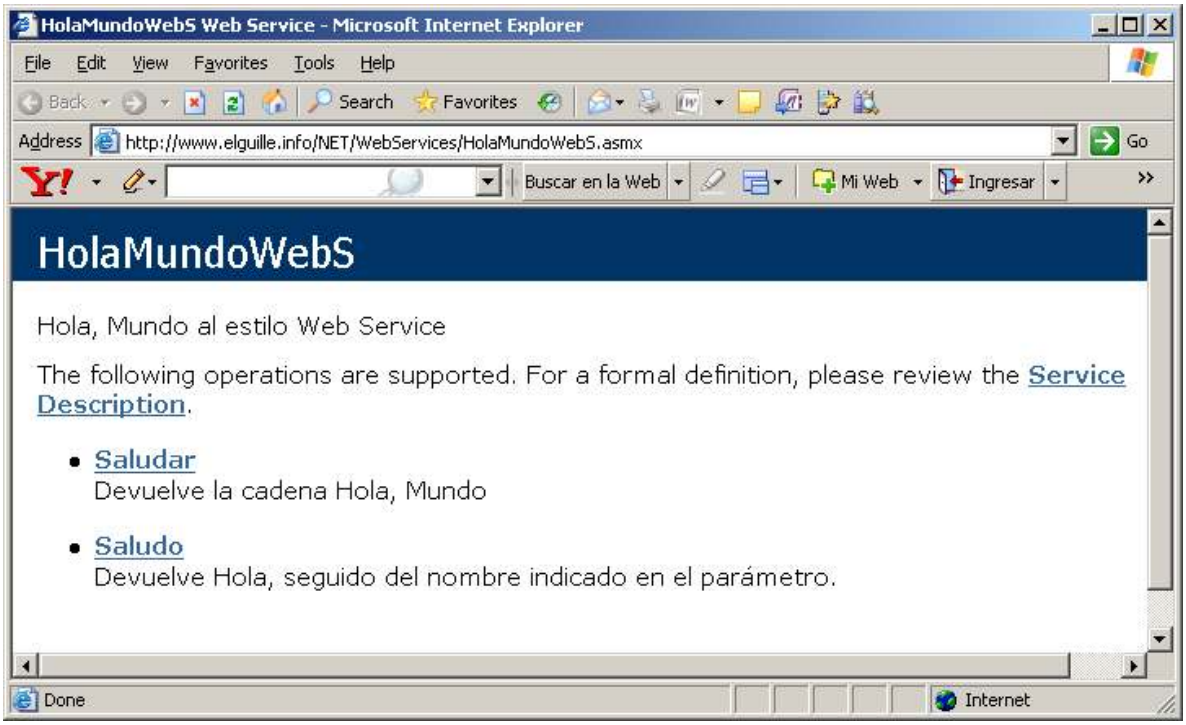

Figura 1.4 - Llamado a un Web Service desde un navegador de internet

Ahora, se debe construir una aplicación, Windows, Web o móvil que será la que hará el llamado al Web Service y tomará la información que éste regrese; se podrá manipular la información, guardarla, actualizar y mostrar resultados dependiendo de lo que el Web Service nos permita hacer y de lo que nuestra aplicación sea capaz de hacer.
En la figura 1.5 se muestra, de forma resumida, el funcionamiento de los Web Services.

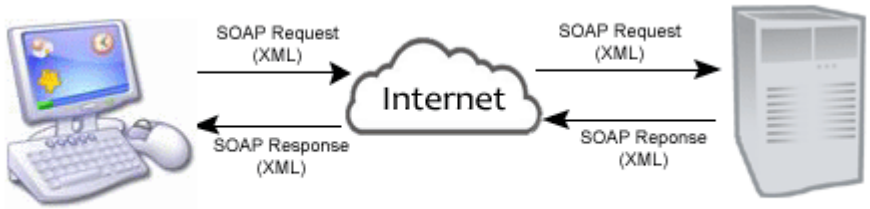

Figura 1.5 - Funcionamiento de los Web Services

1.4.7 XML

XML (Extensible Markup Language) es un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

XML puede tener tres posibles roles:

a) *De contenedor de información:* Cuando el XML actúa de contenedor, ignora por completo la información que contiene. Esta información irá normalmente almacenada en un elemento del tipo CDATA, estos elementos no son analizados por el compilador de XML.

b) *Para definir el contenido de los mensajes*: Para que se puedan intercambiar información dos aplicaciones y puedan procesarla automáticamente.

c) *Para describir el contenido de los mensajes*: Si lo utilizamos con este rol, podemos describir el esquema de contenidos de los mensajes. Estamos definiendo recursos, lo utilizamos como meta datos. Podemos facilitar el acceso a la información a los agentes de software.
La idea del XML es tener la información, la estructura de la información, formato y procesos, todo ello separado.

Algunas de las ventajas de XML son:

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se acelera el desarrollo de la aplicación.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

Básicamente XML está naciendo no sólo para su aplicación en Internet, sino que se propone como lenguaje de bajo nivel para intercambio de información estructurada entre diferentes plataformas. Se puede utilizar en bases de datos, editores de texto, hojas de cálculo, etc.

Respecto a su aplicación en Internet, mejorará lo que HTML ha intentado y es establecer un estándar fijo al que todo el mundo pueda atenerse sin necesidad de depender del Navegador utilizado, del sistema de objetos o de cualquier otra cosa. [7]

1.4.7.1 Serialización

La serialización es el proceso de convertir un objeto de forma que pueda transportarse fácilmente. Por ejemplo, se puede serializar un objeto por internet utilizando HTTP entre un cliente y un servidor. Por el otro lado, la deserialización reconstruye el objeto a partir de la secuencia.
La clase central de la serialización XML es el XMLSerializer y sus métodos más importantes son "Serialize" y "Deserialize". XMLSerialiazer crea archivos C# y los compila en archivos .dll para realizar esta serialización.
La secuencia XML que genera XMLSerializer cumple con la recomendación 1.0 del W3C (www.w3.org) acerca del lenguaje de definición de esquemas XML. [10]

1.4.8 Plataformas que soportan los Web Services

- Axis y el servidor Jakarta Tom cat (de Apache).
- Cold Fusion MX de Macromedia.
- Java Web Services Development Pack (JWSDP) de Sun Microsystems (basado en Jakarta Tomcat).
- JOnAS (parte de Object Web una iniciativa de código abierto).
- Microsoft .NET.
- Novell Extend (Basado en la plataforma J2EE).
- Web Logic.
- Web Sphere.
- Zope es un servidor de aplicaciones Web orientado a objetos desarrollado en el lenguaje de programación Python.
- Verastream de Attachmate WRQ para modernizar o integrar aplicaciones host IBM y VT.

1.5 Arquitecturas N-Tiers

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto, es separar la capa de datos de la capa de presentación del usuario.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varias capas y en caso de algún cambio sólo se ataca a la capa requerida sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por capas, de este modo, cada grupo de trabajo está totalmente abstraído del resto de las capas.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas N-Tiers o programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

La arquitectura que es más usada en la actualidad es la de 3 capas (Véase figura 1.6); otras arquitecturas menos conocidas son: En pipeline, Entre pares, En pizarra, Orientada a servicios y Maquinas Virtuales. [12]

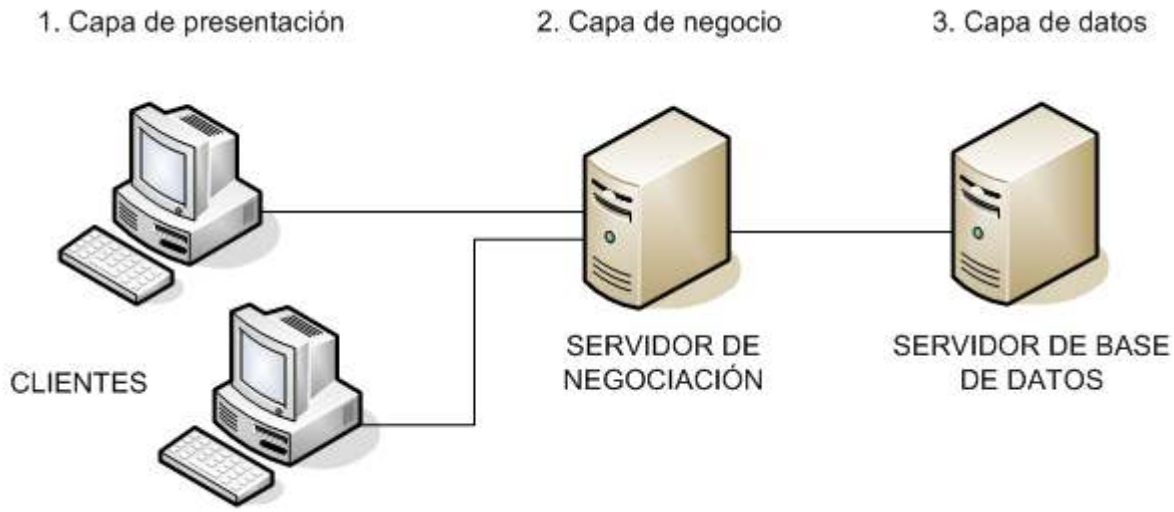

Figura 1.6 - Diseño de 3 capas

Capa de presentación: Es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato en los datos de entrada). Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio: Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

Capa de datos: Es donde residen los datos; está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares. El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/ Lógica de Negocio/ Datos.

En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice, que la arquitectura de la solución es de tres capas y *un nivel*.

Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *dos niveles*.

Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y *tres niveles*.[13]

1.6 Programación Web

1.6.1 Internet

Internet es un método de interconexión de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP y garantiza que redes físicas heterogéneas funcionen como una red (lógica) única. De ahí que Internet se conozca comúnmente con el nombre de "red de redes", pero es importante destacar que Internet no es un nuevo tipo de red física, sino un método de interconexión. Aparece por primera vez en 1969, cuando ARPAnet establece su primera conexión entre tres universidades en California y una en Utah.

Cuando se dice *red de redes* se hace referencia a que es una red formada por la interconexión de otras redes menores.

Al contrario de lo que se piensa comúnmente, Internet no es sinónimo de World Wide Web (WWW). Ésta es parte de Internet, siendo la World Wide Web uno de los muchos servicios ofertados en la red Internet. La Web es un sistema de información mucho más reciente,

desarrollado inicialmente por Tim Berners Lee en 1989. El WWW utiliza Internet como medio de transmisión.

Algunos de los servicios disponibles en Internet aparte de la Web son el acceso remoto a otras máquinas (SSH y telnet), transferencia de archivos (FTP), correo electrónico (SMTP), boletines electrónicos (news o grupos de noticias), conversaciones en línea (IRC y chats), mensajería instantánea, transmisión de archivos (P2P, P2M, Descarga Directa), etc.

1.6.2 Características de la Programación Web

Una aplicación Web es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet. Las aplicaciones Web son populares debido a la practicidad del navegador Web como cliente ligero. La habilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad. Aplicaciones como los Webmails, wikis, Weblogs, MMORPGs, tiendas en línea y la Wikipedia misma son ejemplos bien conocidos de aplicaciones Web.

1.6.3 Ventajas y Desventajas de la Programación Web

Ventajas de programar para Web

- Se tiene un entorno centralizado.
- Al hacer actualizaciones a los sistemas, solo se tendrá que en una computadora, en este caso, el servidor.
- No se debe instalar ningún ejecutable en las máquinas cliente.
- Se tiene un mejor control sobre los sistemas en tiempo de ejecución.
- Se tienen actualizaciones constantes de los sistemas y de la información que muestran.

Desventajas de programar para Web

- Los procesos son lentos.
- Se requiere conexión permanente a Internet.
- Pueden existir huecos en la seguridad de la información.
- Las versiones de los sistemas pueden no ser óptimas para todos los navegadores o resoluciones de pantallas.
- El sistema no logra tener completo control sobre la máquina cliente.

1.7 Programación de Aplicaciones de Escritorio

1.7.1 ¿Qué es una Aplicación de Escritorio?

Una aplicación de escritorio es un programa compilado, escrito en cualquier lenguaje, que le permite al usuario trabajar en una computadora.

Una aplicación de escritorio suele tener un único objetivo: navegar en la Web, revisar correo, explorar el disco duro, etc. Una aplicación que posee múltiples programas se considera un paquete.
Las aplicaciones pueden tener distintas licencias de distribución como ser freeware, shareware, trialware, etc. Todas las aplicaciones tienen un tipo de interfaz, que puede ser una interfaz de texto o una interfaz gráfica (o ambas).

La principal diferencia entre las aplicaciones de escritorio a las aplicaciones Web, son que las primeras están alojadas directamente en la máquina que el usuario está utilizando y se consumen los recursos de dicha máquina, contrario a las aplicaciones Web, que se ejecutan y consumen recursos del servidor que las aloja.

1.7.2 Ventajas y Desventajas de Aplicaciones de Escritorio

Ventajas de programar aplicaciones de escritorio

- Los procesos son rápidos.
- No requieren conexión a Internet, en caso de que accedan a bases de datos en una Intranet, solo se debe configurar la computadora a la Intranet.
- Hay mayor seguridad en los datos.
- Los sistemas no tienen problemas con las resoluciones de pantallas.
- Se tiene un mejor control sobre la máquina que ejecuta el sistema.

Desventajas de programar aplicaciones de escritorio

- Se tiene un entorno descentralizado.
- Las actualizaciones se deben instalar en cada una de las máquinas en donde se ejecuta el sistema.
- Se debe instalar el sistema localmente.
- En caso de falla del sistema, puede bloquear también otras aplicaciones de la misma máquina.
- Las actualizaciones al sistema y a su información pueden ser más lentas.

1.8 Sistemas distribuidos

1.8.1 Concepto de sistema distribuido

Se podría definir como una colección de elementos de cómputo autónomo que se encuentran físicamente separados y no comparten una memoria común, se comunican entre sí a través del intercambio de mensajes utilizando un medio de comunicación. De otra manera más sencilla, un sistema distribuido es aquel en el que dos o más máquinas colaboran para la obtención de un resultado. En todo sistema distribuido se establecen una o varias comunicaciones siguiendo un protocolo prefijado mediante un esquema cliente-servidor. [14]

1.8.2 Características de los sistemas distribuidos

- Cada elemento de cómputo tiene su propia memoria y su propio Sistema Operativo.
- Control de recursos locales y remotos.
- Sistemas Abiertos (Facilidades de cambio y crecimiento).
- Plataforma no Standard (Unix, NT, Intel, RISC, Etc.).
- Medios de comunicación (Redes, Protocolos, Dispositivos, Etc.).
- Capacidad de Procesamiento en Paralelo.
- Dispersion y parcialidad.

1.8.3 Ventajas y desventajas de los sistemas distribuidos

Ventajas:

- Procesadores más poderosos y a menos costos
- Desarrollo de estaciones con más capacidades.
- Las estaciones satisfacen las necesidades de los usuarios.
- Uso de nuevas interfaces.
- Disponibilidad de elementos de comunicación.
- Desarrollo de nuevas técnicas.
- Compartición de recursos (Hardware, Software).
- Eficiencia y flexibilidad (Respuesta rápida, ejecución concurrente de procesos, empleo de técnicas de procesamiento distribuido).
- Disponibilidad y confiabilidad

- Mayores servicios que elevan la funcionalidad (Monitoreo, Telecontrol, Correo Eléctrico, Etc.).
- Crecimiento Modular

Desventajas:

- Procesadores más poderosos y a menos costos
- Requerimientos de mayores controles de procesamiento
- Velocidad de propagación de información (Muy lenta a veces)
- Servicios de replicación de datos y servicios con posibilidades de fallas
- Mayores controles de acceso y proceso
- Administraciones complejas
- Costos altos (Dependiendo de la infraestructura)

1.8.4 Modelo cliente - servidor

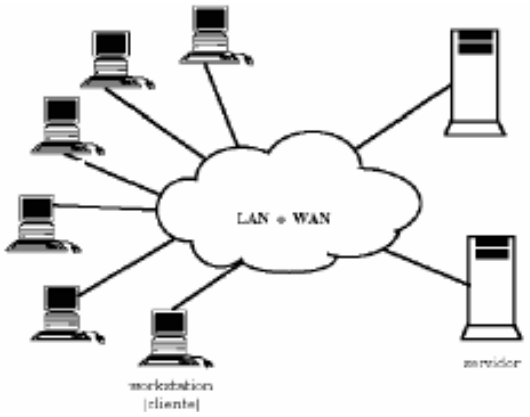

Fig. 1.6 - Modelo Cliente/Servidor (Lan o Wan)

De manera general un esquema cliente-servidor, se denomina cliente a la máquina que solicita un determinado servicio y se denomina servidor a la máquina que lo proporciona.
Un sistema cliente/servidor es aquel en el que uno o más clientes y uno o más servidores, conjuntamente con un sistema operativo subyacente y un sistema de comunicación entre procesos, forma un sistema compuesto que permite cómputo distribuido, análisis, y presentación de los datos.
El servicio puede ser la ejecución de un determinado algoritmo, el acceso a determinado banco de información o el acceso a un dispositivo hardware.

Se puede aplicar el esquema cliente-servidor dentro de una misma máquina, donde el proceso servidor y el proceso cliente son dos procesos independientes que corren dentro de la misma instancia de sistema operativo.
Es primordial la presencia de un medio físico de comunicación entre ambas máquinas, y será la naturaleza de este medio la que marque en muchos casos la viabilidad del sistema.
Se distinguen tres componentes básicos de software:

- Interacción con el usuario
- Lógica de Aplicación
- Repositorio de datos

La interacción con el usuario se realiza básicamente en el servidor. El cliente hace de pasarela, de sistema de acceso a los elementos hardware pantalla y teclado. Los datos se envían sin formatear, y es el cliente el responsable de formatear los datos y realizar las acciones de interacción con el usuario. En este caso, la aplicación y la base de datos se encuentran en el servidor
Tanto la interacción con el usuario como la aplicación residen en el cliente, siendo el servidor el depositario de los datos. Cliente-servidor. La aplicación se distribuye en los tres niveles:

- Aplicación
- Datos
- Interfaces de usuario

1.8.5 Aplicaciones distribuidas utilizando dispositivos móviles

El diseño de aplicaciones modernas involucra la división de una aplicación en múltiples capas; la interfaz de usuario, la capa media de objetos de negocios y la capa de acceso a datos. Es útil identificar los tipos de procesamiento que podemos esperar que una aplicación realice, muchas aplicaciones pueden, al menos, hacer lo siguiente:

- Cálculos u otros procesos de negocios.
- Ejecución de reglas de negocios.
- Validación de datos relacionados al negocio.
- Manipulación de datos.
- Ejecución de las reglas de datos relacional.
- Interactuar con aplicaciones externas o servicios.
- Interactuar con otros usuarios.

Nosotros podemos tomar estos tipos de servicios y generalizarlos dentro de los tres grupos o capas que a continuación se resumen:

- Interfase 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.

La división de estos procesos de aplicaciones y su distribución entre diferentes procesos cliente/servidor es conocido como Procesamiento Distribuido. Generalizando estos procesos dentro de estas tres categorías o capas es una distribución lógica y no refleja necesariamente alguna opción de diseño físico sobre computadoras, terminales u otros equipos. Es posible desarrollar una aplicación cliente/servidor distribuida basada sobre estas tres capas de Presentación, Lógica de Negocios y Servicios de Datos y tener la aplicación entera corriendo sobre una simple computadora. Alternativamente, se puede esparcir estas tres capas a través de un gran número de diferentes computadoras sobre una red. De cualquier forma se ha desarrollado una aplicación distribuida cliente/servidor de tres capas.

Con Dispositivos Móviles

Debido al hecho de que las aplicaciones distribuidas están tomando fuerza sobre el mundo de la informática y es el modelo más utilizado actualmente, es de gran importancia el combinar este modelo de aplicaciones con las nuevas tecnologías y nuevos dispositivos, estos son los dispositivos móviles.

Aunque en la actualidad esté tomando fuerza la programación para móviles, aún no se ha popularizado tanto al llegar al punto de que las mayorías de las empresas desarrollen o tengan en su tecnología desarrollos para dispositivos móviles, sin embargo, lo empiezan a considerar como nuevas opciones para su desempeño laboral. Esto no quiere decir que no existan aplicaciones móviles en las empresas, existen aunque aún son limitadas pues se debe considerar las funciones de la empresa y lo que se desea obtener de la aplicación.

Algunas aplicaciones distribuidas para dispositivos móviles existentes en la actualidad son:

- *wap.HotelDirect.de*

Este sitio alemán de reserva de habitaciones ofrece un buen ejemplo de aplicación distribuida para un teléfono móvil. Tiene como objetivo el poder buscar por ciudades y otras opciones, hoteles y poder realizar reservaciones de habitaciones. (Véase figura 1.7) [15]

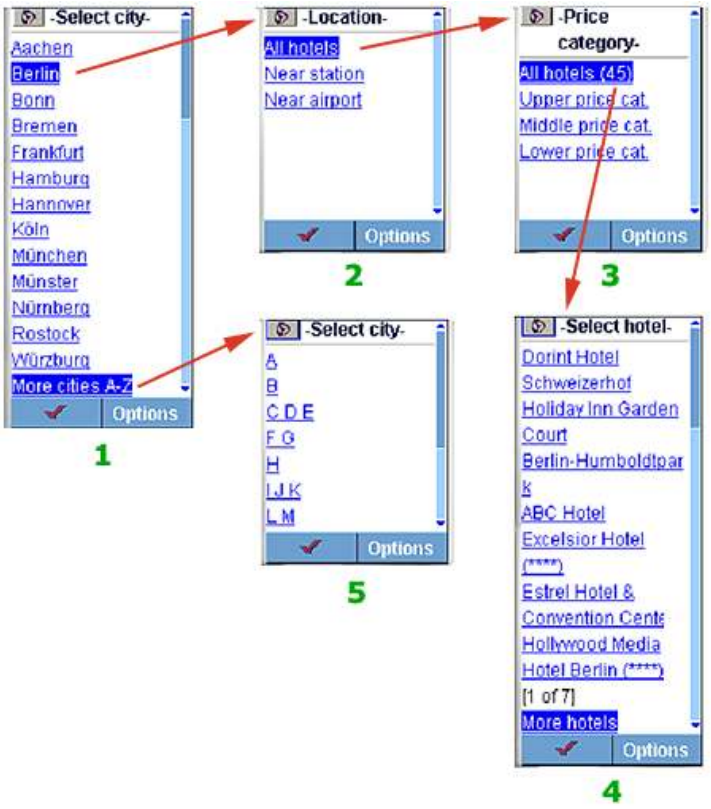

Figura 1.7 – Vista de aplicación HotelDirect

- La empresa española "Global Retail Systems - Soluciones Integrales para la distribución y la logística" (http://www.global-retail.com/res1/home.html) es una empresa de desarrollo de software y hardware, que ofrece en su catálogo de aplicaciones móviles las siguientes aplicaciones:
  - Inventario
  - Pedidos a Proveedor
  - Entrada de Mercancía
  - Facturación a Clientes
  - Autofacturación o Autocompra del Cliente
  - Chequeo de Precios
  - Traspasos de Local
  - Emisión de Etiquetas

- La empresa española "Vertiente – Mobile Technologies" (http://bertiente.com/ber/index.php) ha creado un software para PDA con la que se podrá disponer de la información Geográfica donde se encuentre el usuario.
  Conexión a través de GPRS, WiFi o Bluetooth.

  Si se dispone de un receptor GPS, se puede posicionar y obtener los mapas y la información que interese de la zona en la que el usuario se encuentre.

  Tratamiento Digital de ortofotos: Una vez descargadas las ortofotos se pueden realizar varias operaciones sobre ellas: Zoom más, Zoom menos, Desplazamiento.
  Búsqueda de ortofotos de un municipio: También se puede obtener la ortofoto de un determinado municipio junto con la información disponible en las diferentes capas que ofrezca el servicio de información geográfica.

  Búsqueda de ortofotos mediante introducción manual de coordenadas: En un campo de texto se pueden introducir las coordenadas geográficas que se desean para obtener el mapa correspondiente a ellas.

- Gusanito.com (http://www.nimanic.com/tarjetasmoviles/catalogo_movil.html) Gusanito.com comenzó a ofrecer a sus visitantes el primer sistema de tarjetas móviles del mundo, permitiéndoles elegir y personalizar las tarjetas en su sitio Web, y enviando las tarjetas al celular del destinatario. Además de ser un producto innovador, se trata de un modelo de negocios sin precedentes en Internet, porque el pago de la tarjeta se hace con el celular del remitente, permitiendo a Gusanito.com cobrar $13 pesos + IVA por envío, directamente en el recibo telefónico del usuario o en su tarjeta de prepago.

Capítulo 2

Análisis y Diseño del Sistema

2.1 Planteamiento del Problema

A partir de la década de los ochenta, pareciera gestarse una nueva sociedad en la que la información constituye un elemento determinante de su quehacer cotidiano que nos sumerge en un proceso de modificación de nuestra vida.

En la medida en que se ha ido acelerando el avance de la ciencia, sobre todo en el siglo XX, se ha visto también un avance acelerado en la tecnología; esta tecnología abarca desde la inclusión de computadoras en las cocinas de las casas hasta las grandes investigaciones interespaciales.

El crecimiento de la tecnología en el ámbito de los sistemas de cómputo presenta una tendencia generalizada a incrementar las capacidades de un sistema o dispositivo y decrementar su tamaño y costo, es por eso, que se ha visto el cambio de las grandes computadoras de uso limitado a empresas, a la computadora de escritorio, después a las computadoras portátiles y ahora a dispositivos móviles, los cuales muchos de ellos, actualmente tienen diversas funciones en un solo dispositivo.

Pero, ¿se pueden incrementar las funciones de estos dispositivos móviles según nuestras necesidades?, la teoría dice que si se puede hacer, mediante la adecuada programación de estos dispositivos, la cual, no es la misma que la programación de una computadora de escritorio, pues en un dispositivo se deben considerar sus características como: tamaño del

dispositivo, tamaño de la pantalla, capacidad de memoria, capacidad de interconexión a otros dispositivos, etc.

Este trabajo de tesis, pretende obtener una muestra de la programación de los dispositivos móviles bajo la plataforma .NET con un sistema que conecta el dispositivo móvil a internet para una comunicación con servidores y otros sistemas.

Se mostrará la forma en cómo se puede programar un sistema, destinado para ser utilizado sobre dispositivos móviles con el uso de Web Services, XML, ASP.NET y C#.

2.1.1 Objetivo General

Implementación de un Web Service en .NET para un ambiente distribuido.

2.1.2 Objetivos Específicos

- Analizar la forma de programación de los dispositivos móviles en .NET
- Analizar arquitectura de los Web Services
- Analizar la arquitectura N-Tiers
- Desarrollar una base de datos, que será destinada al albergue de la información referente a las aplicaciones.
- Implementar un Web Service que pueda ser utilizado por aplicaciones para acceder a la base de datos.
- Desarrollar una aplicación de prueba en una página Web diseñada para ser vista en un dispositivo móvil.
- Desarrollar una aplicación de prueba tipo windows para ser ejecutada en un dispositivo móvil.

2.2 Descripción General

El sistema que se implementará pretende ser un ejemplo de la programación de un dispositivo móvil utilizando .NET, en este caso se programará para ser ejecutado y visto en una Pocket PC o un Smartphone, con sistema operativo Windows Mobile.

El sistema será una pequeña aplicación que administrará el historial académico de un estudiante, permitiéndole tener un control del mismo vía Web, además de ser una valiosa fuente de consulta de sus materias aprobadas y reprobadas anteriormente, calificaciones, comentarios propios hechos a profesores y un control de tareas asignadas a cada materia inscrita, así como un acceso al sistema por medio de un nombre de usuario y una contraseña. El nombre del sistema será SIHAT, el cual es la unión de las iniciales del título "Sistema de Historial Académico y Tareas".

Se ha decidido que el sistema tenga la arquitectura de 3 capas, las cuales son de presentación, lógica y de acceso a datos, así como también se ha pensado en utilizar un Web Service y crear tipos de datos propios para el sistema por medio de objetos definidos. Los accesos a la base de datos serán por medio de objetos del ADO.NET utilizando procedimientos almacenados en la base de datos.

2.3 Plataforma

En el capítulo 1 de este trabajo de tesis, se habló referente a la plataforma .NET, los alcances y los servicios que ofrece, viendo que esta plataforma está tomando mucha fuerza en lo referente a la programación y comunicación de sistemas, se ha tomado como base para diseñar y crear el sistema que se desea, aprovechando lo que .NET tiene para ofrecer en cuanto a la programación de dispositivos móviles.

2.4 Herramientas de programación y de producción

A continuación se mencionarán las herramientas necesarias para el desarrollo y la puesta en producción de este sistema:

Herramientas de Programación:

- PC para el desarrollo, que también hará la función de servidor de bases de datos y servidor Web.
- Microsoft Visual Studio .NET 2005
- Microsoft SQL Server 2005.
- .NET Framework 2.0
- IIS 6.0
- Emulador de PocketPC y de Smartphone versión 8.0 de Microsoft que trae por default Visual Studio 2005.

- ActiveSync Versión 4.0

Herramientas de Producción:

- Microsoft SQL Server 2005.
- IIS 6.0
- .NET Framework 2.0
- Dispositivo Pocket PC
- Servidor de bases de datos
- Servidor de páginas Web.
- Servidor de Web Services.

2.5 Arquitectura del Sistema

El sistema está compuesto de varios componentes, juntos estos componentes o secciones crean la base del sistema y su funcionalidad; estos componentes se pueden ver en la figura 2.1.

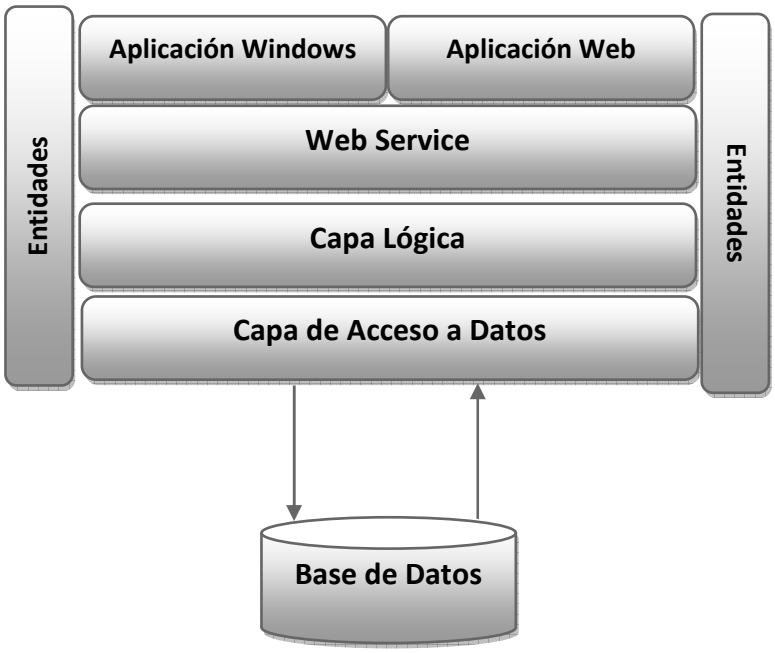

Figura 2.1 – Arquitectura del Sistema.

En la figura 2.1 se muestra que el diseño del sistema tiene en la parte inferior la base de datos, que es donde se almacenarán los datos y es donde se ejecutarán las consultas y los procedimientos almacenados.

En el siguiente nivel superior se tiene la capa de acceso a datos, la cual se encargará de realizar las conexiones y los llamados a los procedimientos almacenados y de llenar el resultado que regresen los objetos y se puedan enviar a la capa lógica; en otro sentido también es la capa responsable de guardar los cambios realizados en la base de datos.

En el siguiente nivel superior se tiene la capa lógica que es la encargada de realizar los métodos y acciones algorítmicas propias del sistema.

En el siguiente nivel superior se tiene el Web Service, que contendrá las funciones del sistema, las cuales podrán ser utilizadas por la aplicación creada o cualquier otra aplicación a la que se le permita el acceso y uso.

En el nivel siguiente se tiene la capa de presentación, la cual estará formada por dos aplicaciones, una es una aplicación Web y la otra es una aplicación windows o de escritorio; con estas dos versiones serán con las que interactuará el usuario.

Se puede ver que de modo general envuelve a todas las capas del sistema un componente de 'Entidades', las cuales son objetos creados específicamente para el sistema y que podrán ser accedidos desde cualquier parte del sistema. Se utilizarán para almacenar y transportar los datos de una capa a otra.

2.6 Diagrama de Componentes

El sistema estará compuesto por 3 componentes principales, los cuales están agrupando algunos de los elementos de la arquitectura del sistema (Véase figura 2.1). Cada uno de estos componentes tiene una funcionalidad específica dentro del sistema principal y cada uno de ellos se relaciona con los demás, formando la estructura y funcionamiento correcto del sistema que se realizará.

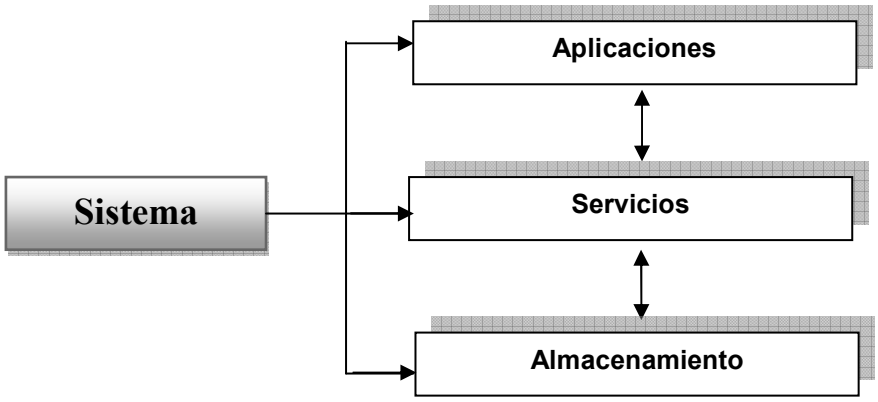

Figura 2.2 – Diagrama de Componentes de Sistema.

Almacenamiento:
Este componente incluye a los elementos 'capa de acceso a datos' y 'base de datos' de la arquitectura del sistema (véase figura 2.1), y estará dedicado al almacenamiento de la información que el sistema manejará.

En este componente se definirá la estructura de la base de datos, consultas SQL, funciones y se definirán los objetos en los que el sistema podrá guardar dicha información para que pueda ser manipulada, a estos objetos se le denominan 'Entidades'.

Servicios:
Este componente estará dedicado a la programación de los servicios que el sistema podrá ofrecer a los usuarios, teniendo en sí métodos y funciones que las aplicaciones podrán solicitar.

Dichos servicios contendrán la lógica del negocio específica para el sistema quue se desarrollará, realizando algoritmos, operaciones de consulta, mandando a llamar al componente de almacenamiento, etc.

Este componente estará compuesto por el 'Web Service' y la 'capa de negocios' de la arquitectura del sistema (véase figura 2.1).

Aplicaciones:
Este componente estará dedicado a la programación de las aplicaciones por las que el usuario final podrá acceder a la información contenida en la base de datos.

Este componente estará compuesto por una aplicación Windows y una aplicación Web, las cuales tendrán el objetivo de ser una pequeña agenda en la que el usuario pueda consultar la información de sus contactos por medio de Internet conectándose en su PocketPC o SmartPhone.

2.7 Proceso de Interacción del Sistema

La figura 2.3 muestra el proceso de interacción del sistema, en el cual se ve como los componentes del sistema interactúan mutuamente y con otros componentes externos.

El proceso inicia cuando alguno de los clientes móviles ingresa al sistema, pudiendo hacerlo por medio de dos opciones, las cuales tienen una interacción distinta (aunque similar) con el resto de los componentes del sistema.

La primera opción es que el usuario decida utilizar el sistema en su versión de aplicación Web, pasando por los siguientes pasos:

1- El dispositivo móvil deberá buscar en la red el servidor de aplicaciones Web que contiene dicha aplicación.
2- Una vez que el dispositivo lo encuentra y obtiene una respuesta por parte del servidor, el cliente realiza las peticiones correspondientes, accediendo al servidor Web.
3- El servidor Web buscará al servidor que aloja al Web Service del sistema.
4- Una vez localizado utiliza alguno de sus servicios.
5- Estos servicios se comunican con la capa lógica.
6- La capa lógica se comunica con la capa de acceso a datos.
7- La capa de acceso a datos se comunica con la base de datos, en donde se realizan las consultas de la petición hecha, retornando el resultado a la capa de acceso a datos.
8- Ésta a su vez, guarda esa información en estructuras de datos y las retorna a la capa lógica.
9- La capa lógica la retorna al Web Service.
10- El Web Service regresa las estructuras de datos mencionadas anteriormente al servidor que lo invocó, es decir, el servidor de aplicaciones Web convirtiendo las estructuras de datos en lenguaje XML.
11- La aplicación Web logra interpretar el XML resultante y envía la página al navegador del cliente (dispositivo móvil) que lo invocó.

Este proceso se repite tantas veces el usuario realice alguna petición al sistema.

La segunda opción es que el usuario decide utilizar el sistema en su versión de aplicación Windows la cual está directamente instalada sobre el dispositivo móvil, pasando por los siguientes pasos:

1- El dispositivo móvil deberá buscar en la red el servidor que aloja al Web Service del sistema.
2- Una vez localizado utiliza alguno de sus servicios, los cuales se comunican con la capa lógica.
3- La capa lógica se comunica posteriormente con la capa de acceso a datos.
4- La capa de acceso a datos se comunica finalmente a la base de datos, en donde se realizan las consultas de la petición hecha, retornando el resultado a la capa de acceso a datos.
5- Ésta a su vez, guarda esa información en estructuras de datos y las retorna a la capa lógica.
6- La capa lógica retorna al Web Service en donde éste regresa las estructuras de datos mencionadas anteriormente al servidor que lo invocó, es decir, a la aplicación Windows, convirtiendo las estructuras de datos en lenguaje XML.
7- La aplicación Windows logra interpretar el XML resultante, mostrando los resultados en la pantalla del dispositivo.

Este proceso se repite tantas veces el usuario realice alguna petición al sistema.

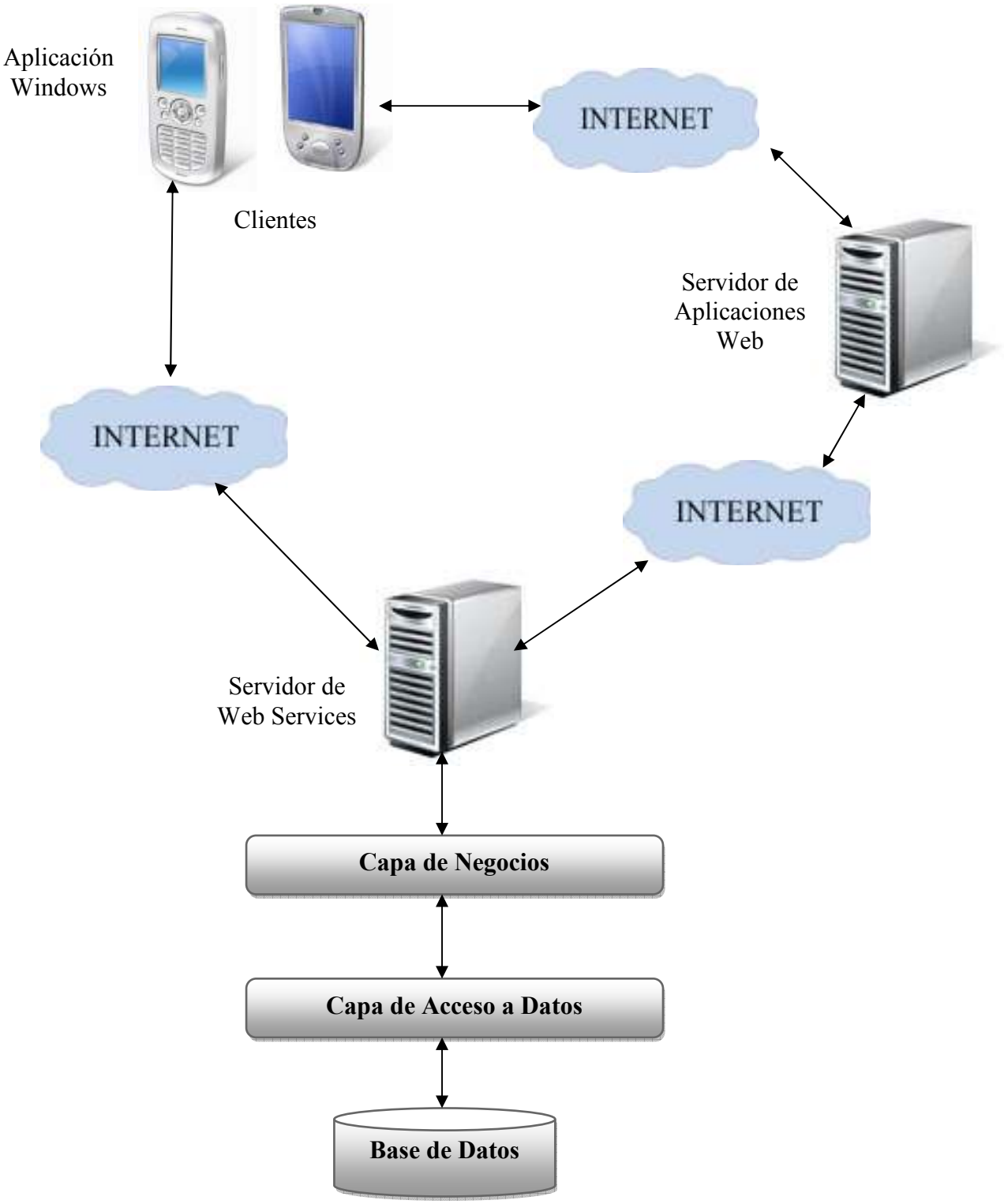

Figura 2.3 – Proceso de Interacción del Sistema.

2.8 Interoperabilidad de los Elementos

El sistema mostrará otra de las características de la tecnología .NET, la cual es la capacidad de unir muchos lenguajes de programación en un solo sistema, por medio de los elementos que lo integran.

La figura 2.4 muestra la interoperabilidad que tendrá el sistema sobre los lenguajes de programación que se utilizarán en cada uno de los elementos del sistema y la forma en cómo están comunicados unos con otros.

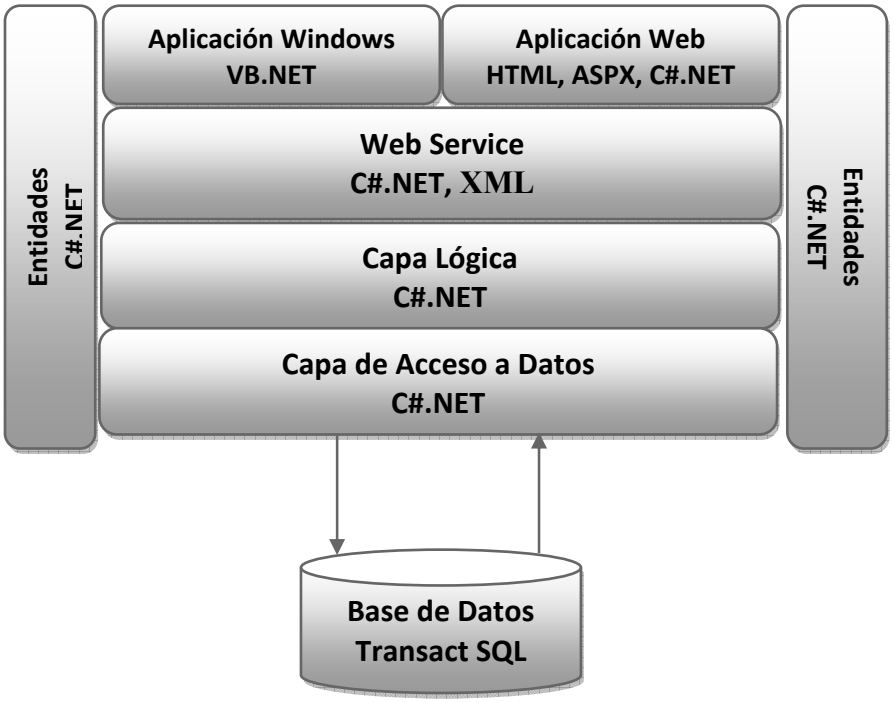

Figura 2.4 – Lenguajes de programación en el sistema.

Aplicación Web:

La aplicación Web estará implementada por medio de páginas Web, las cuales podrán ser vistas desde cualquier navegador Web. Contendrá controles ASPX y la programación será por medio de código en las clases que implementan las páginas.

- **HTML**: Creará la vista de las páginas Web.
- **ASPX**: Creará los controles en el diseño de las páginas.

- **C#.NET**: Creará la programación de la funcionalidad de las páginas y la comunicación con el Web Service.

Aplicación Windows:

La aplicación Windows estará implementada por código escrito en la herramienta de programación Visual Studio.NET, el cual dará un archivo ejecutable para dispositivos móviles.

- **VB.NET:** Creará la programación de la funcionalidad de las pantallas y la comunicación con el Web Service.

Web Service:

El Web Service estará implementado por código creado en la herramienta de programación Visual Studio.NET, el cual dará una carpeta con archivos Web; especialmente el Web Service queda creado en un archivo con la extensión ASMX.

- **C#.NET**: Creará el código de las funciones que ofrece el servicio.
- **XML**: Mostrará el resultado de las funciones que ofrece el servicio.

Capa Lógica:

La capa lógica del sistema estará implementada como una librería de clases creada con código en Visual Studio.NET, el cual dará como resultado un archivo compilado con extensión DLL.

- **C#.NET**: Creará el código de las funciones y comunicación con la capa de acceso a datos.

Capa de Acceso a Datos:

La capa de acceso a datos del sistema estará implementada como una librería de clases creada con código en Visual Studio.NET, el cual dará como resultado un archivo compilado con extensión DLL.

- **C#.NET**: Creará el código de las funciones que se comunican a la base de datos.

Entidades:

Las entidades del sistema estarán implementadas como una librería de clases creada con código en Visual Studio.NET, el cual dará como resultado un archivo compilado con extensión DLL.

- **C#.NET**: Creará las propiedades que tendrá cada una de las entidades.

Base de Datos:

La base de datos que el sistema utilizará, se creará en SQL Server 2005 utilizando las herramientas que dicho motor de base de datos ofrece.

- **Transact SQL:** Crea las consultas y procedimientos almacenados para el sistema.

2.9 Diseño de Interfaz con el Usuario

El funcionamiento de las aplicaciones que se realizarán es sencillo y se pretende que tengan interfaces agradables al usuario y que les sea sencillo de utilizar. Además que se ha analizado de tal forma que se han considerado las limitantes que puede tener una aplicación que será destinada para ser vista en un dispositivo móvil, de tal forma que el diseño de la interfaz con el usuario se presenta en la figura 2.5.

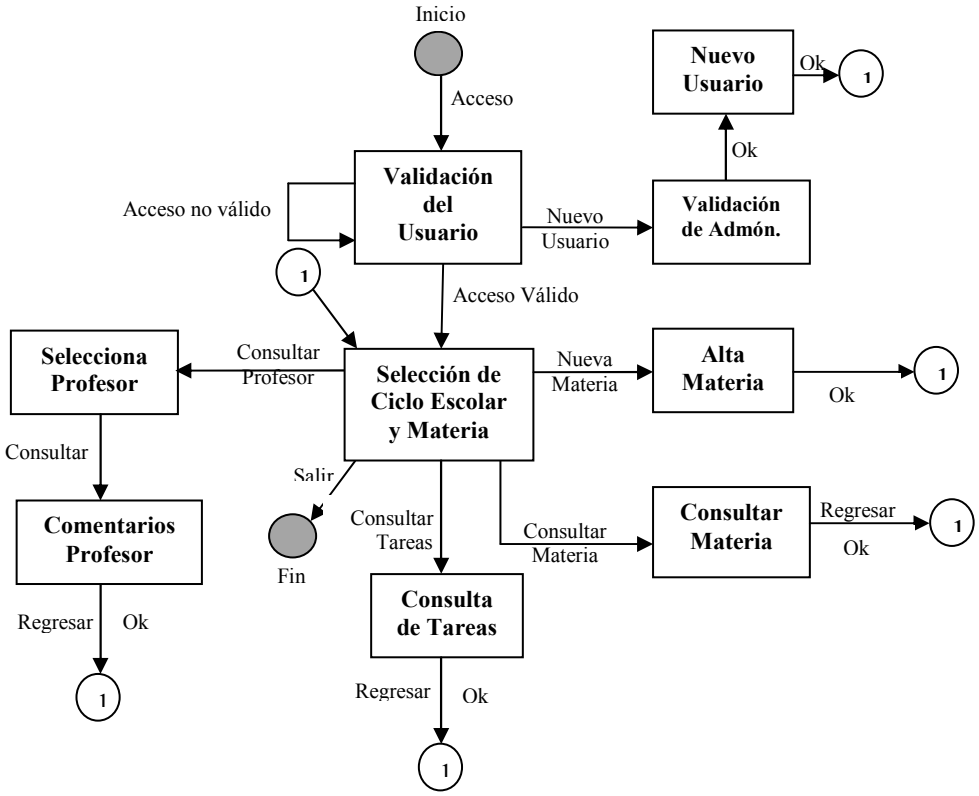

Figura 2.5 – Diseño de Interfaz con el Usuario

En la figura 2.5 se puede apreciar la secuencia de posibles acciones que las aplicaciones permitirán al usuario.
Se puede ver que se iniciará en la pantalla de *validación del usuario*, en donde el usuario deberá ingresar su nombre de usuario y su contraseña, el sistema validará que los datos ingresados sean correctos, de acuerdo a la información existente en la base de datos, en caso de que el usuario y la contraseña no sean válidos, el sistema mostrará un mensaje de alerta, informando al usuario que su nombre de usuario o contraseña son incorrectos, en otro caso, si el sistema encuentra correcta la validación, la aplicación mostrará la segunda pantalla, la 'selección de un ciclo escolar y una materia'.

Para los casos en que el usuario no esté aún dado de alta en el sistema, se podrá dar de alta por medio de la pantalla de '*Nuevo Usuario'*, siempre que haya sido validado como administrador del sistema por medio de la pantalla de '*Validación de Administrador*'.

En la segunda pantalla, '*selección de un ciclo escolar y una materia*' el usuario podrá realizar alguna de las siguientes acciones:

1) En caso de que no cuente, para el ciclo escolar seleccionado, con alguna materia añadida en el sistema, podrá añadirla por medio de la pantalla de 'Alta de Materia'.
    a. *Alta de Materia*: En esta pantalla se presentarán todas las materias existentes en el catálogo de materias, donde el usuario podrá seleccionar una de ellas y añadirla al ciclo escolar seleccionado en la pantalla anterior. En caso de que la materia seleccionada ya exista en el ciclo escolar, el sistema mostrará un mensaje de error, indicando que la materia ya existe en el ciclo escolar.
2) El usuario podrá seleccionar un ciclo escolar y una materia a la que desea consultar su información, entonces podrá realizarlo por medio de la pantalla de '*Consulta de Materia*'.
    a. *Consulta de Materia*: En esta pantalla se podrá revisar y modificar información importante de la materia, como lo es, el salón en donde se imparte, el profesor, la calificación, si es un curso o recurso, etc.
3) El usuario podrá seleccionar un ciclo escolar y una materia a la que desea consultar las tareas que le han sido asignadas, y esto lo hace entrando a la pantalla de '*Consulta de Tareas'*.
    a. *Consulta de Tareas*: La pantalla de consulta de tareas permitirá al usuario añadir nuevas tareas de la materia seleccionada, así como también consultar las anteriores y modificar su información. Contendrá información importante de cada tarea como es: Fecha de inicio, Fecha de fin, porcentaje de avance y estatus.
4) El usuario podrá seleccionar directamente la opción de '*Consulta Profesor'*.

a. *Consulta profesor*: Esta pantalla contiene un listado de todos los profesores existentes en el catálogo de profesores, de los que el usuario podrá seleccionar uno de ellos y consultar sus comentarios, sin necesidad de que ese profesor le haya dado clase o no, esto lo hará por medio de la pantalla '*Comentarios Profesor'*
Comentarios Profesor: Esta pantalla permite añadir comentarios hacia el profesor y consultar los comentarios anteriores, así como también se podrán editar y guardar cambios.

En el momento en que el usuario decida salir del sistema, deberá cerrar la aplicación y automáticamente terminará su sesión.

2.10 Diseño de la Base de Datos

Tablas

La base de datos guardará únicamente la información que es útil para el sistema y su correcto funcionamiento, por lo que analizando el objetivo del sistema, sus aplicaciones y el contenido de lo que se desea obtener en cada una de las pantallas de las aplicaciones, se diseñaron siete tablas para la base de datos del sistema, las cuales están normalizadas y relacionadas para obtener un correcto funcionamiento del sistema.
A continuación se describen las tablas que contendrá la base de datos.

Nombre: Cat_Alumnos
Objetivo: Este es una tabla que contendrá información del alumno que ingresará al sistema como estudiante de la universidad. Será un listado de todos los estudiantes con acceso al sistema.
Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
ClaveAlumno	varchar	10	Not Null	Primary Key	200023349
NombreAlumno	varchar	100	Null	No	Rebeca Maritza
PaternoAlumno	varchar	100	Null	No	Sánchez
MaternoAlumno	varchar	100	Null	No	Becerril
Contraseña	varchar	250	Null	No	202cb962ac5905584
CorreoElectronico	varchar	100	Null	No	beccas_82@hotmail.com

Nombre: Cat_CiclosEscolares
Objetivo: Esta tabla contendrá un listado de los ciclos escolares que han transcurrido en la historia del sistema o están vigentes.
Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
Sesion	varchar	10	Not Null	Primary Key	OTOÑO
Anio	varchar	4	Not Null	Primary Key	2008
Consecutivo	int		Null	No	1

Nombre: Cat_Materias
Objetivo: Esta tabla contendrá la información de las materias que se han impartido en la universidad y que estarán disponibles para que el estudiante pueda registrarlas en el sistema.
Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
ClaveMateria	varchar	10	Not Null	Primary Key	100
NombreMateria	varchar	200	Null	No	Programación

Nombre: Cat_Profesores
Objetivo: Esta tabla contendrá la información de los profesores de la universidad que imparten las materias en cada ciclo escolar.
Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
ClaveProfesor	varchar	10	Not Null	Primary Key	001
NombreProfesor	varchar	100	Null	No	Rafael
PaternoProfesor	varchar	100	Null	No	De la Rosa
MaternoProfesor	varchar	100	Null	No	Flores

Nombre: ComentariosProfesor
Objetivo: Esta tabla contendrá los comentarios hechos por parte de un alumno a un profesor.

Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
ClaveProfesor	varchar	10	Not Null	Primary Key	001
ClaveAlumno	varchar	10	Not Null	Primary Key	200023349
Consecutivo	int		Not Null	Primary Key	3
Comentario	varchar	Max	Null	No	Este es el comentario

Nombre: MateriasDelCiclo
Objetivo: Esta tabla contendrá la información de las materias que ha inscrito el alumno en cada ciclo escolar.
Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
Sesion	varchar	10	Not Null	Primary Key	OTOÑO
Anio	varchar	4	Not Null	Primary Key	2008
ClaveAlumno	varchar	10	Not Null	Primary Key	200023349
ClaveMateria	varchar	10	Not Null	Primary Key	001
Salon	varchar	100	Null	No	B-16
ClaveProfesor	varchar	10	Null	No	001
TipoCurso	int		Null	No	1
Calificación	decimal	(18,1)	Null	No	9.7
Estatus	bit	1	Null	No	0

Nombre: PendientesMateria
Objetivo: Esta tabla contendrá la información de las materias que ha inscrito el alumno en cada ciclo escolar.
Estructura:

Nombre Campo	*Tipo*	*Longitud*	*Valores Nulos*	*Llave*	*Ejemplo*
Sesion	varchar	10	Not Null	Primary Key	OTOÑO
Anio	varchar	4	Not Null	Primary Key	2008
ClaveAlumno	varchar	10	Not Null	Primary Key	200023349
ClaveMateria	varchar	10	Not Null	Primary Key	001

ClavePendiente	varchar	10	Not Null	Primary Key	1
DescribePendiente	varchar	250	Null	No	Proyecto final
FechaInicio	datetime		Null	No	10/09/2008
FechaFin	datetime		Null	No	01/12/2008
Estatus	int		Null	No	0
PorcentajeAvance	int		Null	No	50

2.11 Seguridad

Autenticación a la Base de Datos

Para poder acceder a la base de datos del sistema, no será directamente desde la aplicación Web o windows en el dispositivo móvil, será por medio de un Web Service, el cual estará controlado por un administrador.

Por lo mismo, se tomará como método de autenticación (Web Service → Base de Datos) un nombre de usuario y contraseña, pues se puede tener el control desde el servidor de base de datos, como en el servidor de Web Services.

Para realizar esto, se deberá asignar a una cuenta de usuario los permisos necesarios en el servidor de bases de datos a la base de datos del sistema, posteriormente, se definirá en el archivo Web.config del Web Service una cadena de conexión, asignando como propiedades el método de autenticación y escribiendo el nombre del usuario y la contraseña válida a los que ya se les dio permisos sobre la base de datos.

Paso de información entre páginas y formularios

Para poder mantener la usabilidad, control y funcionalidad del sistema que se creará, será necesario enviar datos e información de una página Web o un formulario windows a otros de su misma categoría.
Analizando las ventajas y desventajas de cada una de las implementaciones para lograrlo, se establece que el sistema en su versión Web trabajará con variables de sesión, por ser de uso común y general actualmente, además que provee al sistema tiempos límites de autentificación sin movimiento por parte del usuario, dando así seguridad de acceso al mismo, además tomando la ventaja de que al ser variables que físicamente existen en el servidor, se puede así optimizar los recursos del cliente.

El sistema, en su versión windows, trabajará por medio de una clase estática, la cual contendrá las variables que son necesarias pasar entre los formularios. Se decidió trabajar de esta forma pues es la manera que recomiendan los expertos, se puede obtener un mejor control sobre las variables al estar todas en una sola clase y se ahorra memoria del cliente al no tener que instanciarse la clase que las contiene.

2.12 Diagrama de Clases

En el sistema se tendrá la información muy clara acerca de qué datos son lo que se requieren y qué datos deben de mostrarse en pantalla o guardarse en la base de datos, por lo que se ha analizado y llegado a la conclusión de que el utilizar alguna estructura compleja de Ado.NET (como puede ser un Data Set) puede llegar a ser muy pesado para el sistema y un desperdicio de memoria, pues dichas estructuras contienen espacio suficiente para guardar cantidades bastante grandes de información y contienen métodos muy útiles para manejar dicha información, sin embargo, este sistema manejará una información moderada a comparación de otros sistemas, por lo que se ha concluido que la mejor forma de optimizar el manejo de la información almacenada y buscando la optimización de los recursos, es creando objetos específicos para este sistema, con las que se podrá almacenar la información que requerimos, creando un espacio de memoria específico para cada uno de los datos que tenemos, reduciendo así pérdida de espacio en memoria y lentitud en el sistema.

Se crearán siete clases distintas para poder manipular la información dentro del sistema, las cuales se ilustran a continuación:

Alumno

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar un alumno con todos los datos que se le encuentren en la base de datos.

Estructura:

ALUMNO
ClaveAlumno
NombreAlumno
PaternoAlumno
MaternoAlumno
Contraseña
CorreoElectronico

CicloEscolar

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar un ciclo escolar específico.

Estructura:

CICLO ESCOLAR
Sesion
Anio
Consecutivo

Comentarios

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar un comentario que cierto estudiante haya hecho sobre alguno de los profesores.

Estructura:

COMENTARIO
ClaveAlumno
ClaveProfesor
Consecutivo
Comentario

InfoMateria

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar la información específica de cierta materia que un alumno desee consultar por medio del sistema.

Estructura:

INFOMATERIA
Salon
ClaveProfesor
TipoCurso
Calificacion
Estatus

Materia

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar la información básica de una materia existente en el catálogo de materias.

Estructura:

MATERIA
ClaveMateria
NombreMateria

Profesor

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar la información básica de profesor existente en la base de datos del sistema.

Estructura:

PROFESOR
ClaveProfesor
NombreProfesor
PaternoProfesor
MaternoProfesor

Tarea

Objetivo: El objetivo de este objeto es crear espacio de memoria específico para almacenar la información de una tarea que haya sido asignada por un alumno a una materia en un ciclo escolar.

Estructura:

TAREA
ClaveAlumno
Anio
ClaveMateria
ClavePendiente
DescribePendiente
Estatus
FechaInicio
FechaFin
PorcentajeAvance
Sesion

Diagrama de Objetos

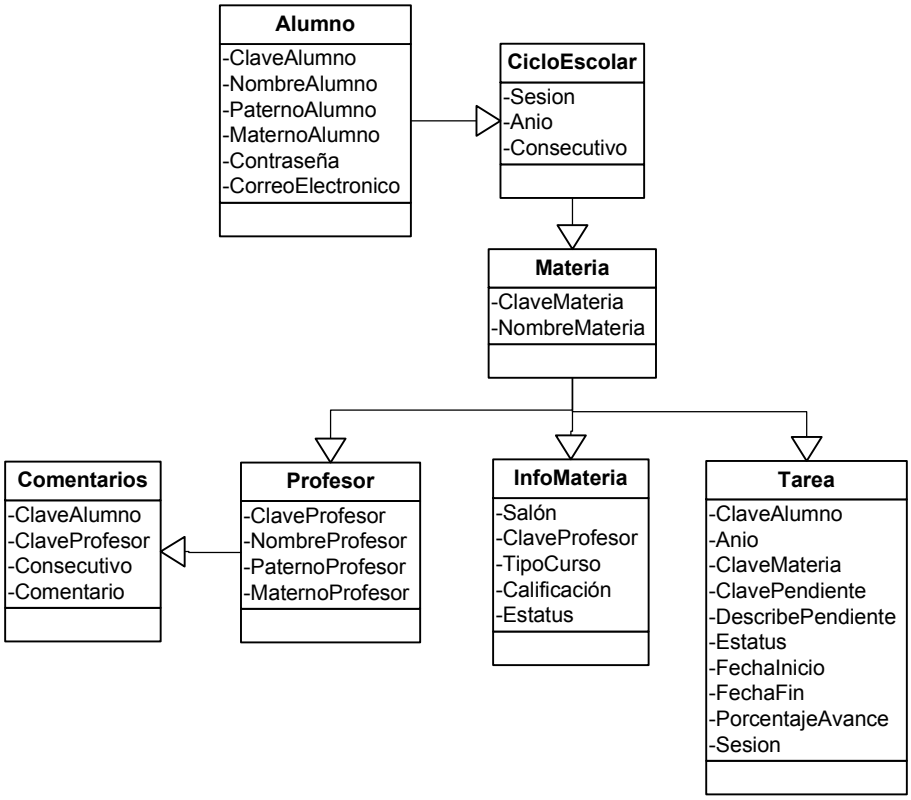

Figura 2.6 – Diagrama de Objetos

Capítulo 3

Implementación y Pruebas al Sistema

En esta sección, se verá la implementación del sistema, se describirán las herramientas de software que se utilizaron, así como también la forma en la que el sistema se ha programado.

También se hablará en este capítulo de las pruebas a las que el sistema ha sido sometido y sus resultados, dando así una vista al lector del sistema completo y funcionando.

3.1 Software

Para la creación del sistema, se han utilizado algunas herramientas de software, las cuales se presentan a continuación:

Microsoft Visual Studio .Net

Visual Studio .Net es un entorno de desarrollo integrado (IDE) creado por Microsoft a partir del año 2002. Es para el sistema operativo Microsoft Windows y está pensado para desarrollar para plataformas Win32.

La última versión del IDE Visual Studio .Net soporta los nuevos lenguajes .Net: C#, Visual Basic.Net y C++. Visual Studio.Net puede utilizarse para construir aplicaciones dirigidas a Windows (utilizando Windows Forms), Web (utilizando ASP.Net y Web Services) y dispositivos móviles (utilizando .Net Compact Framework).

El aspecto de Visual Studio.Net 2005 es casi idéntico a las versiones anteriores del IDE. Algunas excepciones destacables son la interfaz más limpia y mayor cohesión. También es más personalizable con ventanas informativas de estado que automáticamente se ocultan cuando no se usan. Todas las versiones del Visual Studio.Net incluyen un depurador integrado en el entorno de edición.

Microsoft SQL Server

Microsoft SQL Server Management Studio, es un entorno integrado para obtener acceso a todos los componentes de SQL Server, configurarlos, administrarlos y desarrollarlos. SQL Server Management Studio combina un amplio grupo de herramientas gráficas con una serie de editores de secuencia de comandos para ofrecer acceso a SQL Server a programadores y administradores de todos los niveles de especialización.

SQL Server Management Studio combina las características del Administrador Corporativo, el Analizador de Consultas y Analysis Manager, herramientas incluidas en versiones anteriores a la 2005 del SQL Server, en un único entorno. Además, SQL Server Management Studio funciona con todos los componentes de SQL Server, como Reporting Services, Integration Services, SQL Server Compact Edition y Notification Services. Los programadores obtenemos un ambiente conocido y los administradores de bases de datos una única herramienta completa que combina herramientas gráficas fáciles de usar con funciones de secuencias de comandos enriquecidas.

Microsoft ActiveSync

ActiveSync es un programa de sincronización entre dispositivos móviles y computadoras de escritorio hecho por Microsoft. ActiveSync permite gestionar la instalación de nuevos programas o aplicaciones, explorar la estructura de directorios, transferir todo tipo de archivos, sincronizar con bandeja de entrada, contactos y tareas de Microsoft Outlook, navegar por Internet, etc.

Los dispositivos móviles que pueden ser sincronizados con ActiveSync son las Pocket PC o Smartphone que tienen sistema operativo Windows; también puede ocuparse con otros dispositivos que no utilizan un sistema operativo de Microsoft, por ejemplo los que utilizan una plataforma de Symbian.

ActiveSync también considera la transferencia manual de archivos a un dispositivo móvil, junto con funcionalidad limitada de restaurar y la capacidad de instalar y de desinstalar.

Comenzando con Windows Vista, ActiveSync se ha substituido por el centro móvil del dispositivo de Windows, el cuál es un componente que viene incluido en Windows Vista.

Emulador de Pocket PC

Un emulador es un programa que funciona como un verdadero dispositivo móvil. Un emulador trata de modelar de forma precisa el dispositivo que esta emulando, lo que hace la diferencia con los simuladores, los cuales sólo tratan de reproducir el funcionamiento del programa.

Esta aplicación puede servir para practicar en una PC el funcionamiento de una pocket PC y hacer las mismas cosas que haría en una verdadera, como instalar aplicaciones, llevar una agenda, tener contactos, tareas, documentos, conectarse a distintas redes, etc.

Un emulador se refiere tanto a un equipo de hardware como a los programas de software.

3.2 Desarrollo

3.2.1 Componente de Almacenamiento

Base de Datos

En primera instancia se ha creado la base de datos en SQL Server 2005, utilizando el programa Microsoft SQL Server Management Studio que permite la administración de las bases de datos y la creación de nuevas bases por medio la opción de creación de base de datos nueva, en donde sólo hay que especificar el nombre, el servidor, el propietario y algunas opciones de configuración y seguridad. Automáticamente el servidor crea y aloja la base de datos con las opciones configuradas.

La vista del diseñador de la base de datos está mostrada en la figura 3.1.

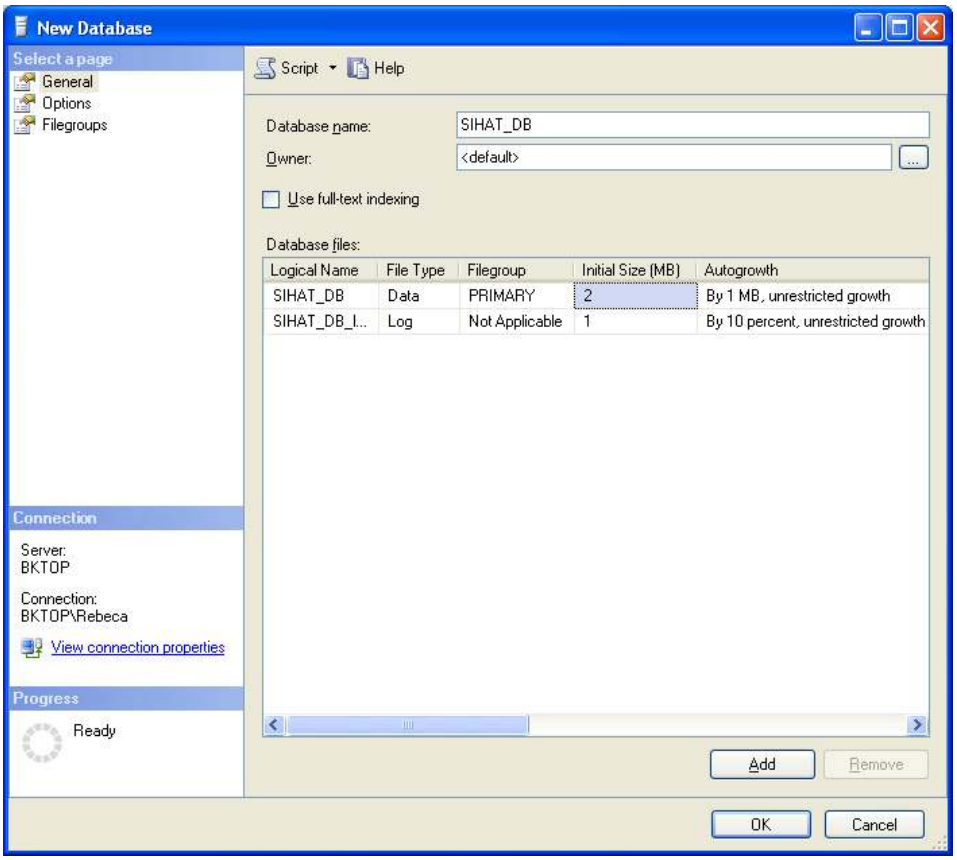

Figura 3.1 – Vista de diseñador para crear la base de datos.

Posteriormente se han creado las siete tablas que la base de datos contendrá; la creación de nuevas tablas es por medio de dos opciones, ejecutando un script que contenga la orden de creación de cada tabla con la estructura específica, o creándola por medio de la vista de diseño.

Las tablas que se crearon para el sistema fueron creadas por medio de la vista de diseño, siendo estas más amigables, las cuales se presentan en la figura 3.2. Se han incluido en el CD del sistema, el script necesario para la creación de estas tablas por medio de un query para SQL Server.

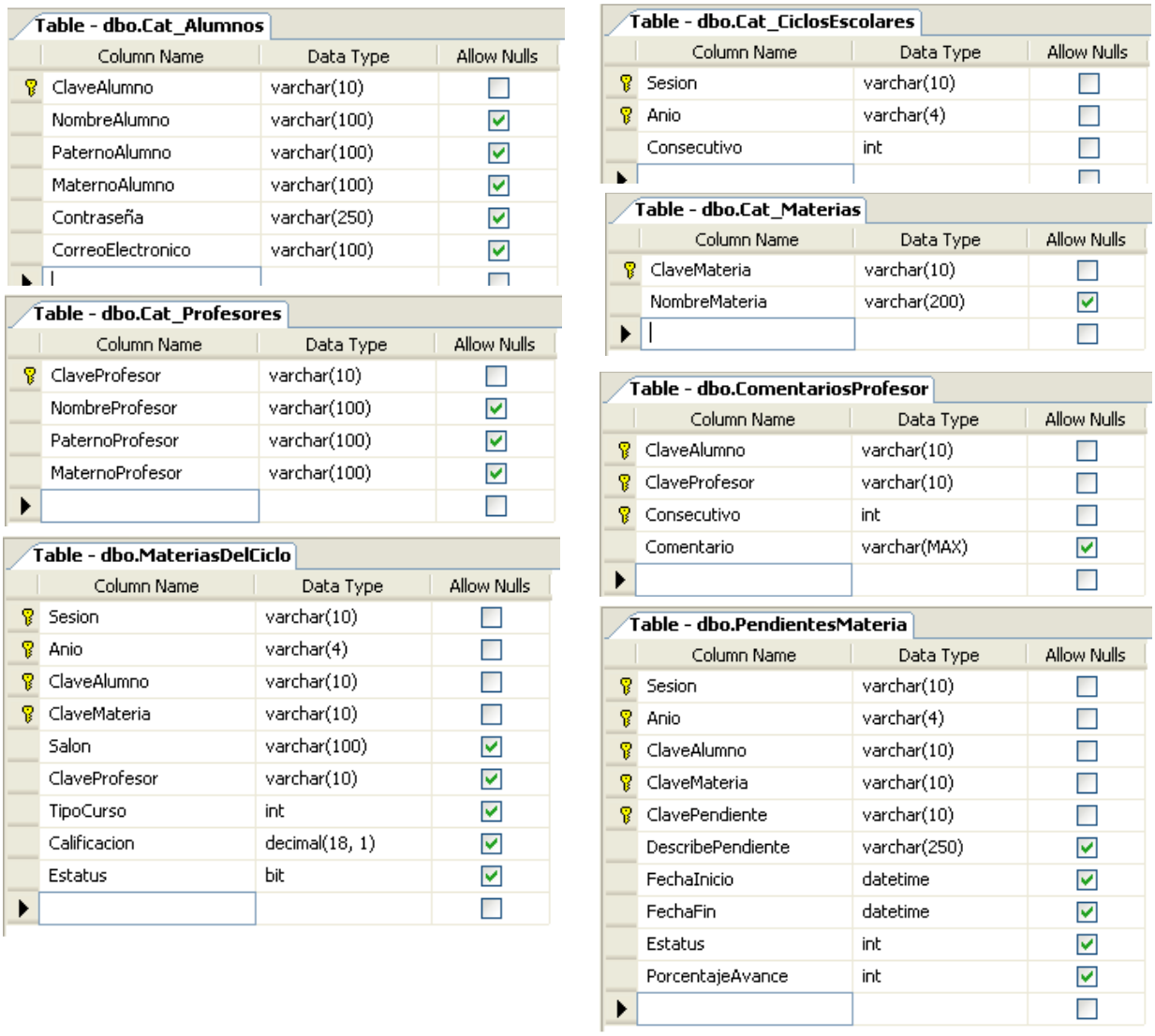

Table - dbo.Cat_Alumnos

Column Name	Data Type	Allow Nulls
ClaveAlumno	varchar(10)	☐
NombreAlumno	varchar(100)	☑
PaternoAlumno	varchar(100)	☑
MaternoAlumno	varchar(100)	☑
Contraseña	varchar(250)	☑
CorreoElectronico	varchar(100)	☑

Table - dbo.Cat_Profesores

Column Name	Data Type	Allow Nulls
ClaveProfesor	varchar(10)	☐
NombreProfesor	varchar(100)	☑
PaternoProfesor	varchar(100)	☑
MaternoProfesor	varchar(100)	☑

Table - dbo.MateriasDelCiclo

Column Name	Data Type	Allow Nulls
Sesion	varchar(10)	☐
Anio	varchar(4)	☐
ClaveAlumno	varchar(10)	☐
ClaveMateria	varchar(10)	☐
Salon	varchar(100)	☑
ClaveProfesor	varchar(10)	☑
TipoCurso	int	☑
Calificacion	decimal(18, 1)	☑
Estatus	bit	☑

Table - dbo.Cat_CiclosEscolares

Column Name	Data Type	Allow Nulls
Sesion	varchar(10)	☐
Anio	varchar(4)	☐
Consecutivo	int	☐

Table - dbo.Cat_Materias

Column Name	Data Type	Allow Nulls
ClaveMateria	varchar(10)	☐
NombreMateria	varchar(200)	☑

Table - dbo.ComentariosProfesor

Column Name	Data Type	Allow Nulls
ClaveAlumno	varchar(10)	☐
ClaveProfesor	varchar(10)	☐
Consecutivo	int	☐
Comentario	varchar(MAX)	☑

Table - dbo.PendientesMateria

Column Name	Data Type	Allow Nulls
Sesion	varchar(10)	☐
Anio	varchar(4)	☐
ClaveAlumno	varchar(10)	☐
ClaveMateria	varchar(10)	☐
ClavePendiente	varchar(10)	☐
DescribePendiente	varchar(250)	☑
FechaInicio	datetime	☑
FechaFin	datetime	☑
Estatus	int	☑
PorcentajeAvance	int	☑

Figura 3.2 – Vista de diseñador para crear las tablas del sistema.

Una vez que se han creado la base de datos y las tablas que tendrá dicha base, se continuó con la unión de todas estas tablas en un diagrama principal de la base de datos, el cual presenta relaciones entre estas tablas por medio de las llaves de cada tabla (véase figura 3.3).

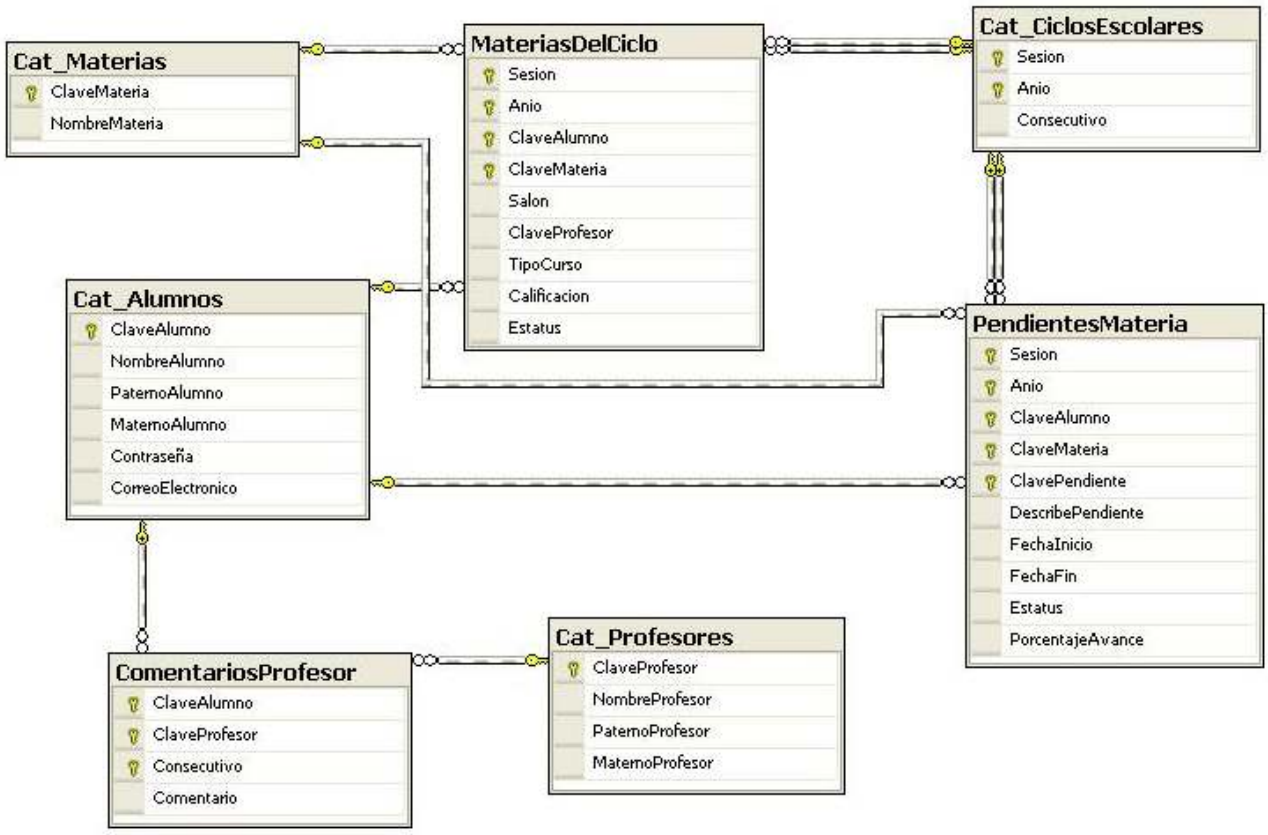

Figura 3.3 – Vista del diagrama relacional de la base de datos

Después de la creación del diagrama relacional de la base de datos, se procedió con la creación de los procedimientos almacenados, los cuales contienen las consultas específicas que el sistema requiere ser aplicadas sobre las tablas que ya fueron creadas.

Se crearon 14 procedimientos almacenados, los cuales se numeran y explican brevemente a continuación: (El contenido de estos procedimientos almacenados se han incluido en el CD del sistema)

1- **SP_GUARDACOMENTARIOSPROFESOR**: Con este procedimiento almacenado se puede modificar algun comentario hecho por un estudiante a un profesor y también se puede añadir nuevos comentarios y el procedimiento le asigna un número consecutivo al nuevo comentario automáticamente.

2- **SP_GUARDAINFORMACIONMATERIA**: Con este procedimiento almacenado se puede actualizar la información de una materia que ya existe en un periodo y alumno específico, pudiendo con esto modificar información como salón, profesor, tipo curso, calificación y estatus.

3- **SP_GUARDANUEVAMATERIAENCICLO**: Con este procedimiento almacenado se puede verificar que una materia exista o no en un ciclo escolar especificado para cierto alumno y en caso de que la materia no exista, entonces se podrá dar de alta.

4- **SP_GUARDATAREASMATERIA**: Con este procedimiento almacenado se puede modificar alguna tarea hecha por un estudiante para cierta materia y también se puede añadir nuevas tareas y el procedimiento le asigna un número consecutivo a la nueva tarea automáticamente.

5- **SP_INSERTAALUMNO**: Con este procedimiento almacenado se puede dar de alta un nuevo alumno y usuario al sistema.

6- **SP_REGRESACICLOSESCOLARES**: Con este procedimiento almacenado se puede obtener una lista de los ciclos escolares que han sido dados de alta en la base de datos, específicamente, sobre la tabla 'Cat_CiclosEscolares'.

7- **SP_REGRESACOMENTARIOSPROFESOR**: Con este procedimiento almacenado se puede obtener una lista de los comentarios que cierto alumno ha ingresado en el sistema acerca de algún profesor específico.

8- **SP_REGRESAINFORMACIONMATERIA**: Con este procedimiento almacenado se obtiene toda la información de una materia para un ciclo y alumno especificado.

9- **SP_REGRESAMATERIAS**: Con este procedimiento almacenado se puede obtener una lista de las materias que han sido dadas de alta en la base de datos, específicamente, sobre la tabla 'Cat_Materias'.

10- **SP_REGRESAMATERIASCICLOESCOLAR**: Con este procedimiento almacenado se puede obtener una lista de las materias que un alumno específico tiene dadas de alta en determinado ciclo escolar.

11- **SP_REGRESAPROFESORES**: Con este procedimiento almacenado se puede obtener una lista de los profesores que han sido dados de alta en la base de datos, específicamente, sobre la tabla 'Cat_Profesores'.

12- **SP_REGRESATAREASMATERIA**: Este procedimiento almacenado obtiene una lista de las tareas que han sido dadas de alta en la base de datos para un alumno, materia y ciclo escolar específico.

13- **SP_VALIDAALUMNO**: Este procedimiento almacenado revisa si una clave de alumno y contraseña son válidos para el sistema, revisando esta información contra la tabla 'Cat_Alumnos'.

14- **SP_VALIDAEXISTEALUMNO**: Este procedimiento almacenado revisa si existe en la tabla 'Cat_Alumnos' una clave de alumno y una contraseña especificada.

Entidades

Al terminar el desarrollo de la base de datos, se comenzó con la programación de las clases, iniciando con un proyecto tipo Librería de clases llamado 'SIHAT_Ent', donde se crearon las siguientes clases públicas:

- Alumno.cs
- CicloEscolar.cs
- Comentarios.cs
- InfoMateria.cs
- Materia.cs
- Profesor.cs
- Tarea.cs

En todas las clases ya mencionadas, se maneja el código de cada una de las propiedades que se quieren definir, de la siguiente forma:

1) Escribiendo un encabezado XML (con el objetivo de que todas las propiedades de la estructura que crearemos sean serializables al momento de llegar al Web Service) con la especificación de que la propiedad puede contener o no valores nulos y especificando el tipo de dato que contendrá la propiedad.
2) Se declaran las variables privadas del tipo que deseemos que sean y se especifica un nombre. Estas variables son las que contendrán los datos que se le ingresen a cada propiedad.
3) Se establece el contenido de la propiedad mediante un método que tendrá el mismo nombre que la propiedad (exceptuando el primer carácter) en el que el cuerpo tendrá un get y un set a la variable previamente declarada.

Ejemplo:

1)
```
[XmlElement(IsNullable = true, DataType = "String")]
```

2)
```
private string _ClaveMateria;
```

3)
```
public string ClaveMateria
    {
    get { return _ClaveMateria; }
    set { _ClaveMateria = value; }
    }
```

Una vez que se ha concluido las clases, se ha generado la interfaz de cada una de las clases, la cual se hace pública y será la que se instanciará al momento de crear un objeto de la clase respectiva.

Al compilar el proyecto se crea un archivo llamado 'SIHAT_Ent.dll' el cual se añadirá como referencia a los proyectos siguientes que utilizarán estas estructuras.

Capa de Acceso a Datos

Se procedió a crear un nuevo proyecto tipo Librería de Clases llamado 'SIHAT_DAO' (SIHAT – Data Access Object), en donde están escritas 14 funciones, las cuales contienen el código necesario para poder conectar a la base de datos, hacer una solicitud a algún procedimiento almacenado y enviar los parámetros de entrada necesarios, solicitar la ejecución del procedimiento almacenado y recibir la respuesta de la base de datos, analizarla, guardarla en alguna de las estructuras creadas y enviar los resultados de regreso a la siguiente capa.

3.2.2 Componente de Servicios

Para el componente de servicios se construyó la capa lógica y el Web Service.

Para la capa Lógica, se creó un proyecto de tipo librería de clases llamado 'SIHAT_Log', donde se ha creado una clase llamada 'SIHAT_Log.cs' en donde están escritas, también, 14 funciones, las cuales contienen el código necesario para invocar a las funciones de la capa DAO, obtenerlas en una estructura y enviarlas al servicio que lo invocó.

El Web Service, para su creación se ha añadido a la solución como un nuevo Web Service con nombre 'WS_SIHAT', en el que se renombró al archivo con extensión .asmx como 'Service.asmx' y su clase correspondiente como 'Service.cs' en la que se escribieron 14 funciones, las cuales contienen el código necesario para llamar a las funciones de la capa lógica, tomando los valores que regresan y regresándolos a la aplicación que las haya invocado (véase figura 3.4).

A continuación, se presenta el código y la forma correcta de escribir una función en un Web Service.

```
[WebMethod]
    [System.Xml.Serialization.XmlInclude(typeof(Alumno))]
        [return: XmlElement(typeof(Alumno))]
        public Alumno ValidaAlumnoWS(string strClaveAlumno, string     strContraseña)
  {
    SIHAT_Log.SIHAT_Log ObjLog = new SIHAT_Log.SIHAT_Log(strCadenaConexion);
    Alumno Alumno = new Alumno();
    Alumno = ObjLog.ValidaAlumnoLOG(strClaveAlumno, strContraseña);
    ObjLog = null;
    return Alumno;
  }
```

El Web Service contiene también un archivo de configuración llamado 'Web.config' en el que se especifica la configuración del servicio, incluye configuraciones default a las que se le puede añadir configuraciones específicas de cada sistema.

Para este sistema, se añadió la cadena de conexión a la base de datos por medio de la variable 'Cadena_Conexion' y como se muestra un poco más adelante, y en la clase del Web Service (Service.cs) se lee esa cadena del Web.Config para que todos los métodos puedan usarla.

```
<connectionStrings>
        <add name="Cadena_Conexion" connectionString="Data Source = BKTOP; Initial Catalog =
        SIHAT_DB; user id = SIHAT; password = 123456SIHAT;"/>
</connectionStrings>
```

Estos tres proyectos tienen en cada una de sus clases funciones que mandan a llamar a otra función de la siguiente capa en cascada (véase figura 3.4), de hecho, cada proyecto coincide teniendo 14 funciones cada uno, donde, cada una de esas funciones en los diferentes proyectos, tiene un nombre descriptivo de su objetivo más el sufijo DAO, Log, WS para los proyectos de DAO, Logica y Web Service respectivamente, de tal forma que si en el proyecto SIHAT_DAO existe una función llamada 'ValidaAlumnoDAO', esa función será invocada por el proyecto de SIHAT_Log por la función llamada 'ValidaAlumnoLogica', la cual a su vez, será llamada por el Web Service por medio de la función 'ValidaAlumnoWS' donde las tres funciones reciben los mismos parámetros de entrada y regresan el mismo tipo de dato.

Es por esta razón que no se hablará de las 42 funciones que están en estos tres proyectos, si no que se hablará genéricamente de solo las 14 básicas, especificando en las figuras 3.4 y 3.5 la forma en cómo se comunican las funciones entre los proyectos SIHAT_DAO, SIHAT_Log y WS_SIHAT.

Nombre: *ValidaAlumno.*
Objetivo: Validar que el alumno que desea acceder al sistema, esté registrado en la base de datos. **Entrada:** Clave del alumno y la contraseña a validar. **Procedimiento almacenado que ejecuta:** SP_VALIDAALUMNO. **Salida:** La función regresa un objeto del tipo Alumno de la librería SIHAT_Ent, el cual ya contendrá la información del alumno que se está validando llenando en su propiedad 'Contraseña' el texto 'Alumno_Valido', lo cual indica que el alumno es válido en el sistema.

Nombre: *InsertaAlumno.*
Objetivo: Función que inserta un nuevo alumno en la base de datos, ingresando su información básica y su contraseña dentro del sistema. **Entrada:** Objeto de tipo 'Alumno' con la información del alumno que se insertará. **Procedimiento almacenado que ejecuta:** SP_INSERTAALUMNO. **Salida:** La función regresa una variable de tipo booleana, lo que indica si el proceso de inserción fue exitoso o no.

Nombre: *ValidaAlumno.*
Objetivo: Función que revisa si una clave de alumno ya existe en el sistema. **Entrada:** string con la clave del alumno. **Procedimiento almacenado que ejecuta:** SP_VALIDAEXISTEALUMNO. **Salida:** La función regresa un objeto del tipo Alumno de la librería SIHAT_Ent, el cual ya contendrá la información del alumno, llenando en su propiedad 'Contraseña' el texto 'Alumno_Valido', lo cual indica que el alumno es válido en el sistema.

Nombre: *RegresaCiclosEscolares.*
Objetivo: Función que regresa los ciclos escolares existentes en el sistema. **Entrada:** **Procedimiento almacenado que ejecuta:** SP_REGRESACICLOSESCOLARES**.** **Salida:** La función regresa un objeto del tipo ArrayList, el cual contendrá un objeto de tipo 'CicloEscolar' con la información de 'Sesion', 'Anio' y 'Consecutivo' asignada del registro que coincide en la búsqueda en la base de datos.

Nombre: *RegresaMateriasDelCicloEscolar.*
Objetivo: Función que regresa las materias de un alumno según el ciclo escolar especificado. **Entrada:** Variables tipo string con la información de clave alumno, sesion y año **Procedimiento almacenado que ejecuta:** SP_REGRESAMATERIASCICLOESCOLAR**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Materia' los cuales fueron llenados con la información que coincide de la base de datos.

Nombre: *RegresaMaterias.*
Objetivo: Función que regresa las materias de todo el catálogo. **Entrada:** Variables tipo string con la información de clave alumno, sesion y año. **Procedimiento almacenado que ejecuta:** SP_REGRESAMATERIAS**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Materia' los cuales fueron llenados con la información que coincide de la base de datos.

Nombre: *GuardaNuevaMateriaEnCiclo.*
Objetivo: Función que guarda una nueva materia en el ciclo escolar del estudiante. **Entrada:** Variables de tipo string con la información de Clave alumno, Sesion, Anio y clave materia. **Procedimiento almacenado que ejecuta:** SP_ GUARDANUEVAMATERIAENCICLO**.** **Salida:** Valor de tipo 'string' que indica si el procedimiento de inserción se llevo a cabo correctamente o no.

Nombre: *RegresaTodosProfesores.*
Objetivo: Función que regresa a todos los profesores. **Entrada:** **Procedimiento almacenado que ejecuta:** SP_ REGRESAPROFESORES**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Profesor' los cuales fueron llenados con la información que coincide de la base de datos.

Nombre: *RegresaInformacionMateria.*
Objetivo: Funcion que regresa la informacion de una materia especifica. **Entrada:** Variables de tipo string con la información de Clave alumno, Sesion, Anio y clave materia. **Procedimiento almacenado que ejecuta:** SP_ REGRESAINFORMACIONMATERIA**.** **Salida:** Objeto de tipo 'InfoMateria' que contiene en sus propiedades, la información de la

base de datos que coincide con los parámetros de búsqueda.

Nombre: *ActualizaInformacionMateria.*
Objetivo: Funcion que guarda la información de la materia especifica. **Entrada:** Variables de tipo string con la información de Clave alumno, Sesion, Anio, clave materia, salon, clave profesor, tipo, calificacion y estatus. **Procedimiento almacenado que ejecuta:** SP_ GUARDAINFORMACIONMATERIA**.** **Salida:** Una variable de tipo booleano que indica si el proceso de actualización se llevó a cabo correctamente o no.

Nombre: *RegresaComentariosProfesorAlumno.*
Objetivo: Función que regresa a todos los comentarios de un profesor respecto un alumno. **Entrada:** Variables de tipo string con la información de Clave alumno y clave profesor. **Procedimiento almacenado que ejecuta:** SP_ REGRESACOMENTARIOSPROFESOR**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Comentarios' los cuales fueron llenados con la información que coincide de la base de datos.

Nombre: *GuardaComentariosProfesor.*
Objetivo: Funcion que actualiza o guarda comentarios del profesor respecto a un alumno. **Entrada:** Variables de tipo string con la información de Clave alumno, clave profesor, comentarios y la clave del comentario. **Procedimiento almacenado que ejecuta:** SP_ REGRESACOMENTARIOSPROFESOR**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Comentarios' los cuales fueron llenados con la información que coincide de la base de datos.

Nombre: *RegresaTareasMateriaAlumno.*
Objetivo: Función que regresa a todos las tareas de una materia respecto un alumno. **Entrada:** Variables de tipo string con la información de Clave alumno, clave materia, sesion y año. **Procedimiento almacenado que ejecuta:** SP_ REGRESATAREASMATERIA**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Tarea' las cuales fueron llenados con la información que coincide de la base de datos.

Nombre: *GuardaTareasMateria.*
Objetivo: Funcion que actualiza o guarda tareas de una materia respecto a un alumno. **Entrada:** Variables de tipo string con la información de Clave alumno, clave materia, sesion, año, descripcion de la tarea, clave de la tarea, fecha de inicio, fecha de fin, porcentaje de avance y estatus. **Procedimiento almacenado que ejecuta:** SP_ GUARDATAREASMATERIA**.** **Salida:** La función regresa un valor de tipo ArrayList que contiene objetos de tipo 'Tarea' las cuales fueron llenados con la información que coincide de la base de datos.

Diagrama de secuencia

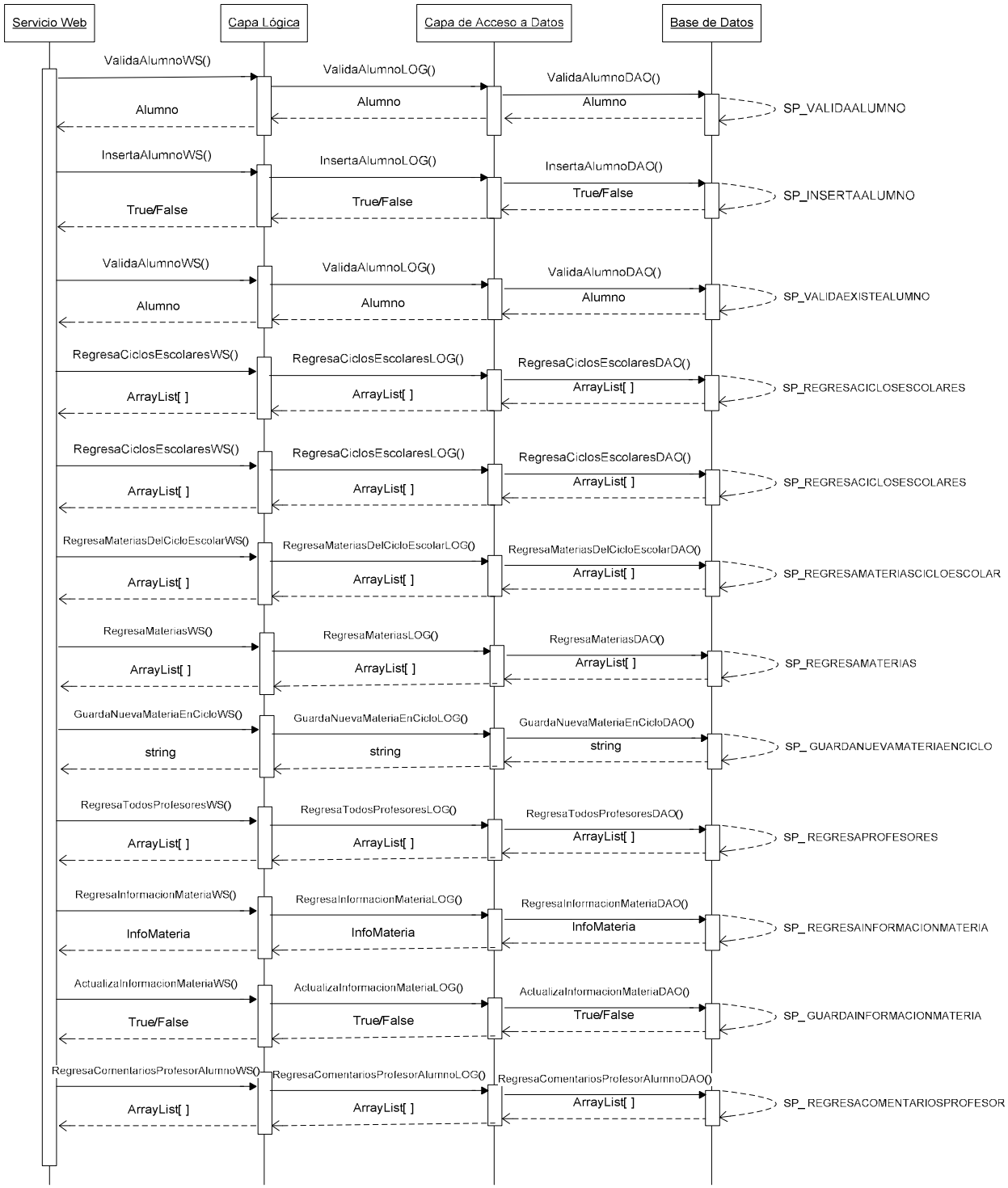

Figura 3.4 – Diagrama de secuencia de los métodos del sistema (Parte 1)

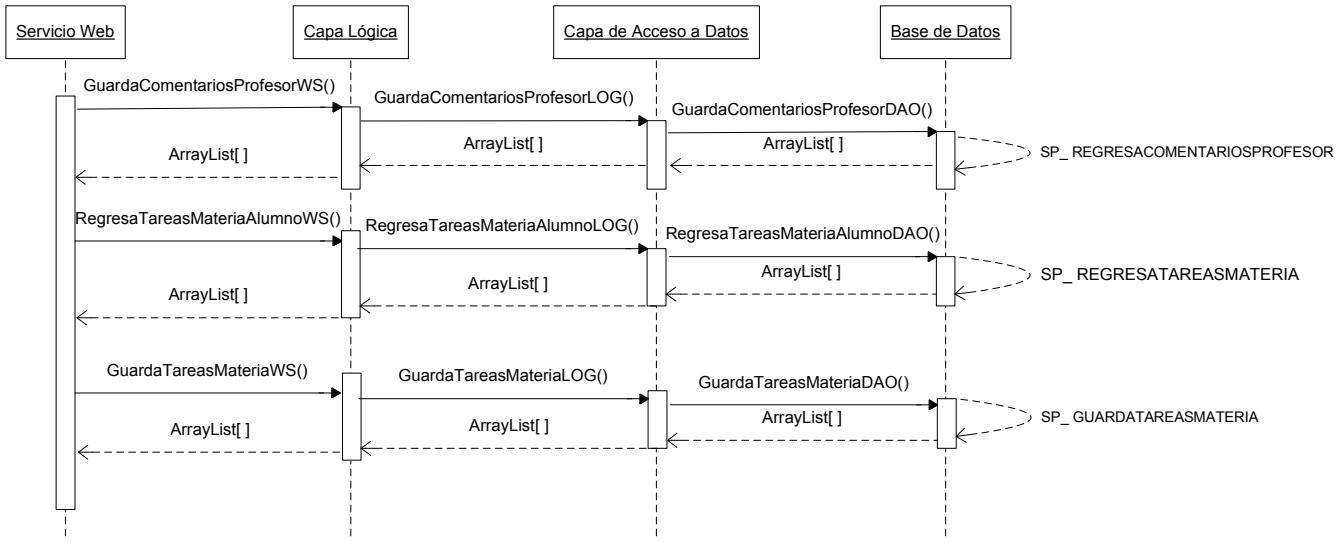

Figura 3.5 – Diagrama de secuencia de los métodos del sistema (Parte 2)

3.2.3 Componente de Aplicaciones

El componente de aplicaciones del sistema, está formado por la aplicación en su versión para Web y su versión Windows, las cuales se detallan a continuación:

Aplicación Web y Aplicación Windows:

La aplicación Web está formada por ocho páginas ASPX, las cuales están hechas con código C#; la aplicación Windows está formada por ocho Forms Windows, las cuales fueron añadidas en un proyecto para un dispositivo móvil y están hechas con código visual Basic. Tanto las páginas ASPX de la aplicación Web como las Forms de la aplicación Windows, según su nombre, tienen la misma funcionalidad, controles y objetivo.

Default

Nombres:
Aplicación Web: Default.aspx
Aplicación Windows: Default.vb

Descripción:
Es la página/forma de validación del alumno que quiere entrar al sistema, contiene dos controles TextBox en donde se ingresará el número de alumno y contraseña, y un botón con la leyenda "Aceptar", el cual al ser oprimido llama al método ValidaAlumnoWS del Web Service pasando como parámetros la matricula del alumno y contraseña que han sido ingresados en las cajas de texto, posteriormente compara el valor que el método ValidaAlumnoWS le regresa y niega o permite el acceso al sistema. En caso de que el acceso sea negado, se muestra un mensaje en pantalla y limpia los controles, en otro caso, el sistema pasa a la página/forma 'SelCicloEscolar.aspx'/'SelecCicloEscolar.vb'.

Figura 3.6 – Interfaz en tiempo de diseño (Default.aspx y Default.vb).

Selección de Ciclo Escolar y Materia

Nombres:
Aplicación Web: SelCicloEscolar.aspx
Aplicación Windows: SelecCicloEscolar.vb

Descripción:
Es la página/forma principal del sistema, pues es en esta, donde el usuario puede seleccionar las diferentes opciones del menú:

- Seleccionando un ciclo escolar, puede dar de alta una nueva materia → pasa a la página/forma 'NuevaMateria.aspx'/'NuevaMateria.vb'
- Seleccionando un ciclo escolar y una materia, puede dar de alta tareas → pasa a la página/forma 'ConsultarTareas.aspx'/'ConsultarTareas.vb'.
- Seleccionando un ciclo escolar y una materia, puede consultar el detalle de la materia → pasa a la página/forma 'ConsultarMateria.aspx'/'ConsultarMateria.vb'.
- Puede consultar a un profesor → 'ConsultarProfesor.aspx'/'ConsultarProfesor.vb'.

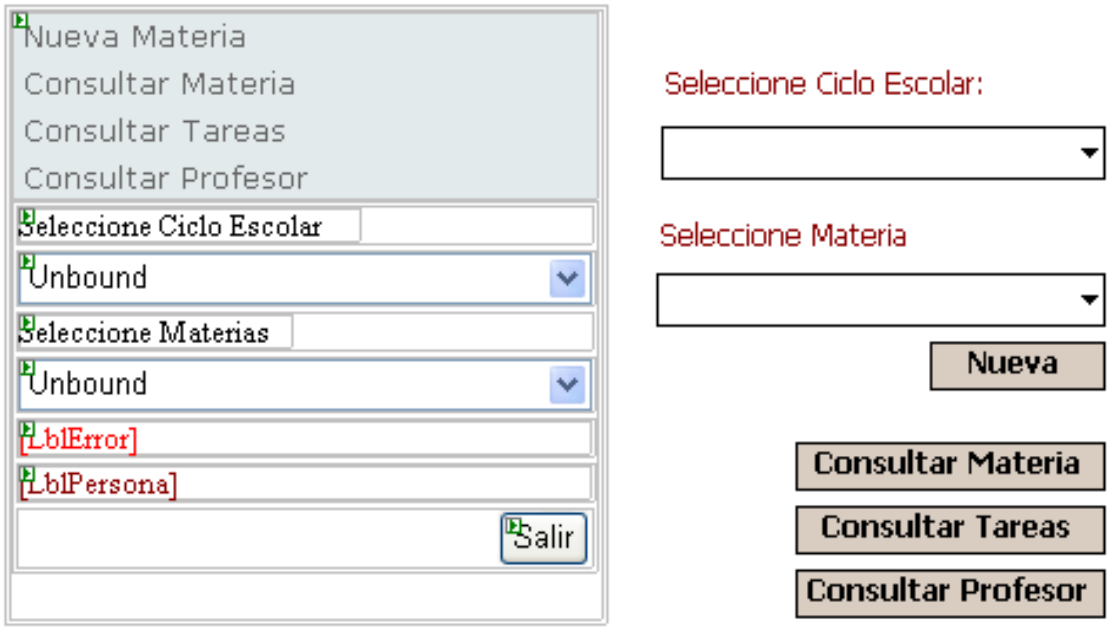

Figura 3.7 – Interfaz en tiempo de diseño (SelCicloEscolar.aspx y SelecCicloEscolar.vb).

Nueva Materia

Nombres:
Aplicación Web: NuevaMateria.aspx
Aplicación Windows: NuevaMateria.vb

Descripción:
Es la página/forma en donde se puede dar de alta una nueva materia en el sistema para el alumno en un ciclo escolar.
La página/forma muestra controles donde se escribirá el nombre de la persona que se ha logeado en el sistema y el nombre del ciclo escolar que previamente se ha seleccionado, así como también presenta un control tipo lista desplegable en el que se seleccionará la materia que se desea añadir al ciclo escolar; presionando el botón 'Añadir' el sistema mandará llamar al método 'GuardaNuevaMateriaEnCicloWS' del Web Service, pasando como parámetros la clave del alumno, la sesión, el año y la clave de la materia, ejecutándose el método, entonces se habrá dado de alta en el sistema la nueva materia, para el alumno en el ciclo escolar seleccionado.
Para salir de la pantalla, se deberá oprimir el botón 'Cancelar'.

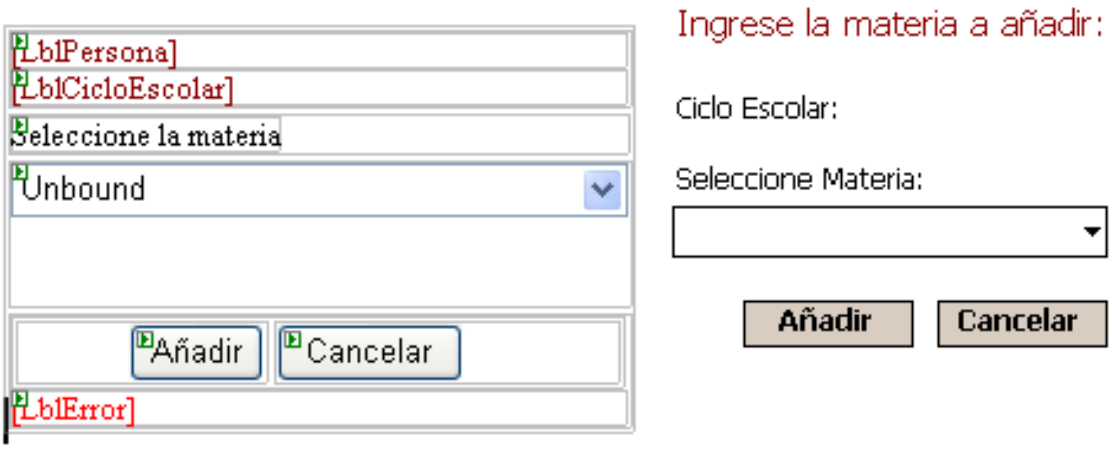

Figura 3.8 – Interfaz en tiempo de diseño (NuevaMateria.aspx y NuevaMateria.vb).

Consultar Materia

Nombres:
Aplicación Web: ConsultarMateria.aspx
Aplicación Windows: ConsultarMateria.vb

Descripción:
Es la página/forma en donde se puede consultar el detalle de una materia ingresada en el sistema.
La página/forma muestra controles donde se escribirá el nombre de la persona que se ha logeado en el sistema, el nombre del ciclo escolar y el nombre de la materia que previamente se ha seleccionado, así como también presenta controles para guardar el salón en el que se impartió el curso, una lista desplegable en donde se puede seleccionar el profesor que ha impartido el curso, la opción de seleccionar si es un curso o un recurso, una caja de texto para guardar la calificación del curso y una lista desplegable para seleccionar el estado actual del curso.
Cuando se desean guardar los cambios marcados en los controles de la pantalla, se deberá oprimir el botón 'Guardar', con lo que el sistema manda a llamar al método 'ActualizaInformacionMateriaWS' del Web Service.
Para salir de la pantalla, se deberá oprimir el botón 'Regresar'.

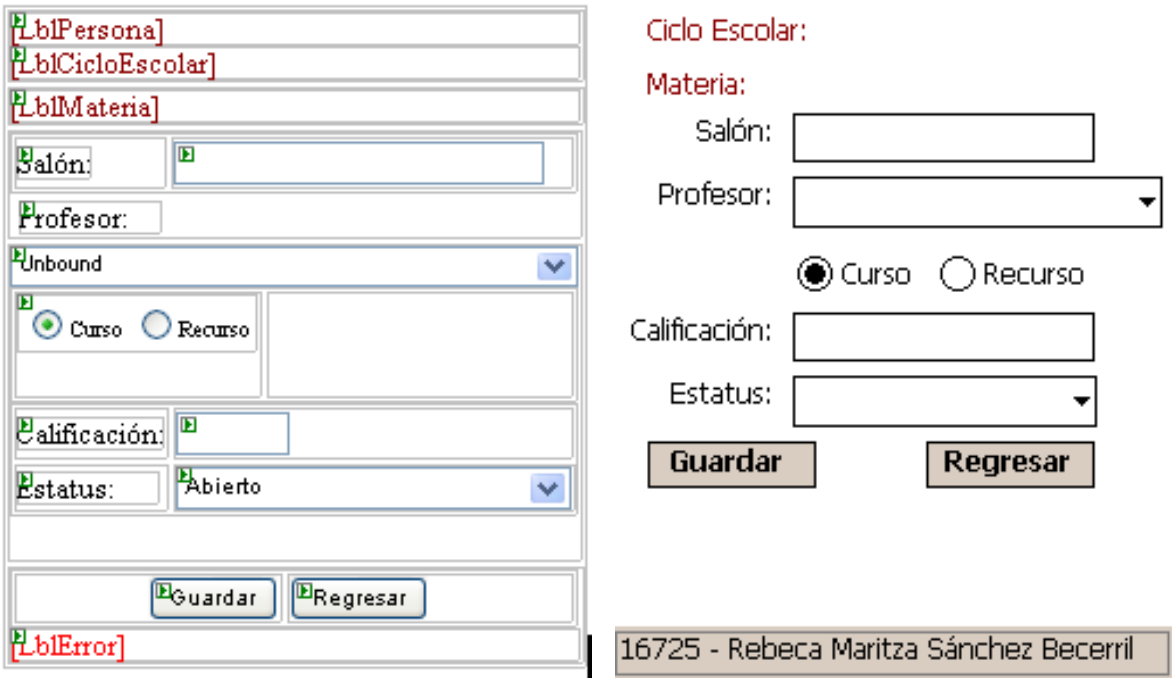

Figura 3.9 – Interfaz en tiempo de diseño (ConsultarMateria.aspx y ConsultarMateria.vb).

Consultar Profesor:

Nombres:
Aplicación Web: ConsultarProfesor.aspx
Aplicación Windows: ConsultarProfesor.vb

Descripción:
Es la página/forma en donde se puede seleccionar del catálogo de profesores del sistema, un profesor con el objetivo de consultar o añadir comentarios acerca del profesor.
La página/forma muestra controles donde se escribirá el nombre de la persona que se ha logeado en el sistema y una lista desplegable donde se podrá seleccionar a algún profesor.
Una vez que se ha seleccionado un profesor, se deberá oprimir el botón 'Consultar' con lo que el sistema manda a llamar a la página/forma 'ComentariosProfesor.aspx' / 'ComentariosProfesor.vb'.
Para salir de la pantalla, se deberá oprimir el botón 'Regresar'.

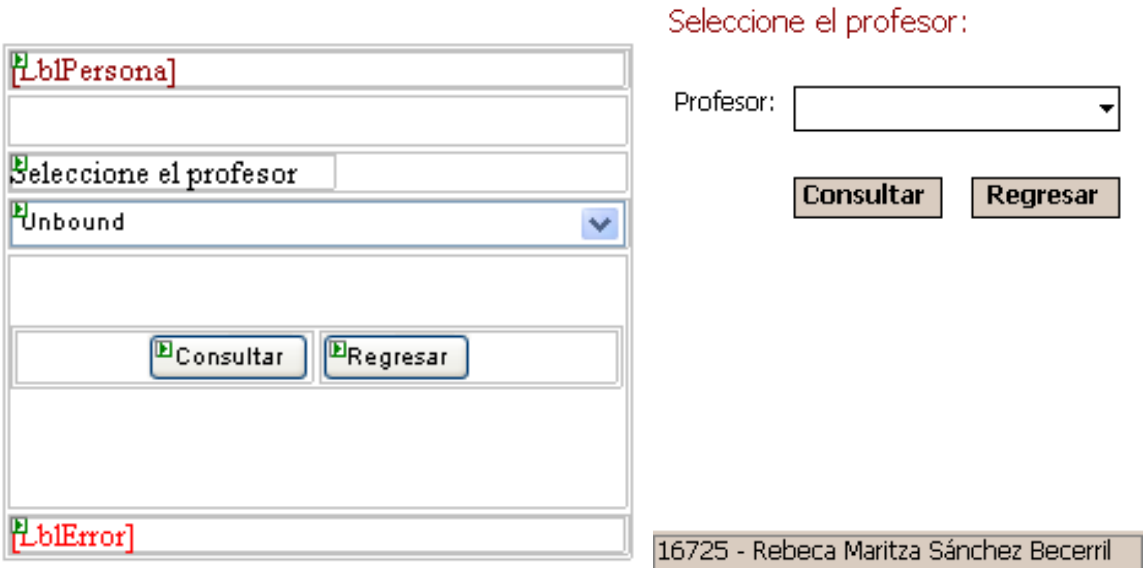

Figura 3.10 – Interfaz en tiempo de diseño (ConsultarProfesor.aspx y ConsultarProfesor.vb).

Añadir Comentarios:

Nombres:
Aplicación Web: AnadirComentarios.aspx
Aplicación Windows: ComentariosProfesor.vb

Descripción:
Es la página/forma en donde se pueden consultar y añadir comentarios a un profesor específico.
La página/forma muestra controles donde se escribirá el nombre de la persona que se ha logeado en el sistema, el nombre del profesor que se trata, así como también presenta un control para escribir el contenido del comentario, y botones de anterior y siguiente, con los que se pueden consultar comentarios hechos previamente acerca del mismo profesor.
En caso de que se desee guardar un nuevo comentario, se debe oprimir el botón 'Nuevo', escribir el comentario y finalmente oprimir el botón 'Guardar', con lo que el sistema manda a llamar al método 'GuardaComentariosProfesorWS' del Web Service.
Para salir de la pantalla, se deberá oprimir el botón 'Regresar'.

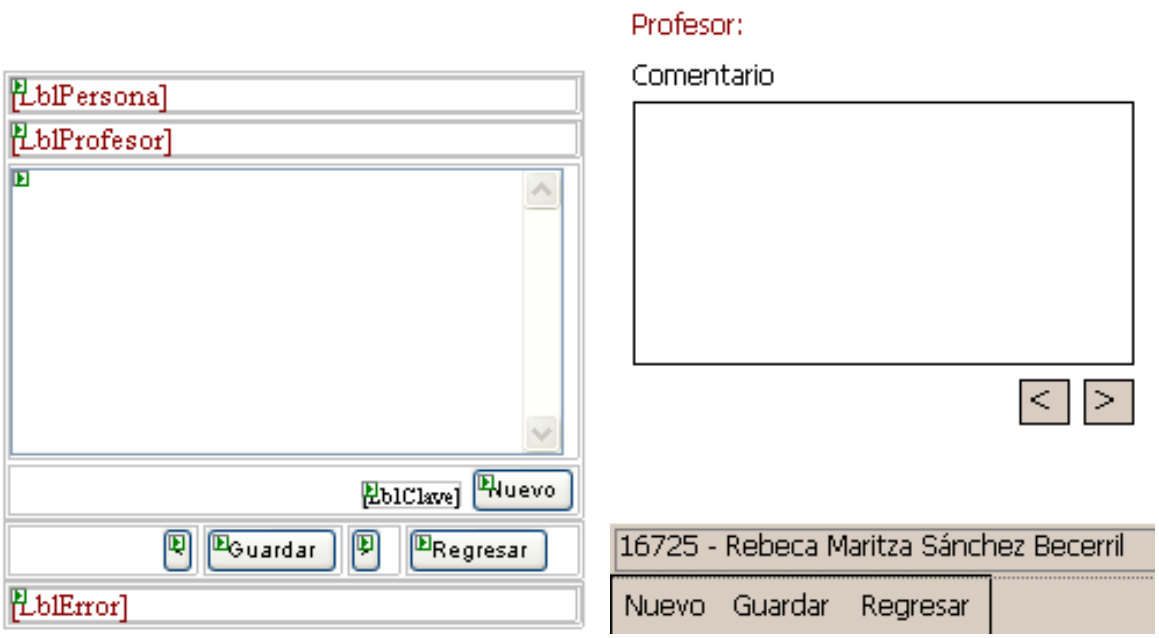

Figura 3.11 – Interfaz en tiempo de diseño (AñadirComentarios.aspx y ComentariosProfesor.vb).

Consultar Tareas:

Nombres:
Aplicación Web: ConsultarTareas.aspx
Aplicación Windows: ConsultarTareas.vb

Descripción:
Es la página/forma en donde se puede dar de alta o consultar el detalle de una tarea ingresada en el sistema.
La página/forma muestra controles donde se escribirá el nombre de la persona que se ha logeado en el sistema, el nombre del ciclo escolar y el nombre de la materia que previamente se ha seleccionado, así como también presenta controles para guardar la descripción de la tarea, la fecha de inicio y fin de la tarea, el porcentaje de avance y el estatus actual de la tarea. La pantalla también presenta botones de anterior y siguiente, con los que se pueden consultar las tareas hechos previamente.
En caso de que se desee guardar una nueva tarea, se debe oprimir el botón 'Nueva', escribir el detalle de la tarea y finalmente oprimir el botón 'Guardar', con lo que el sistema manda a llamar al método 'GuardaTareasMateriaWS' del Web Service.
Para salir de la pantalla, se deberá oprimir el botón 'Regresar'.

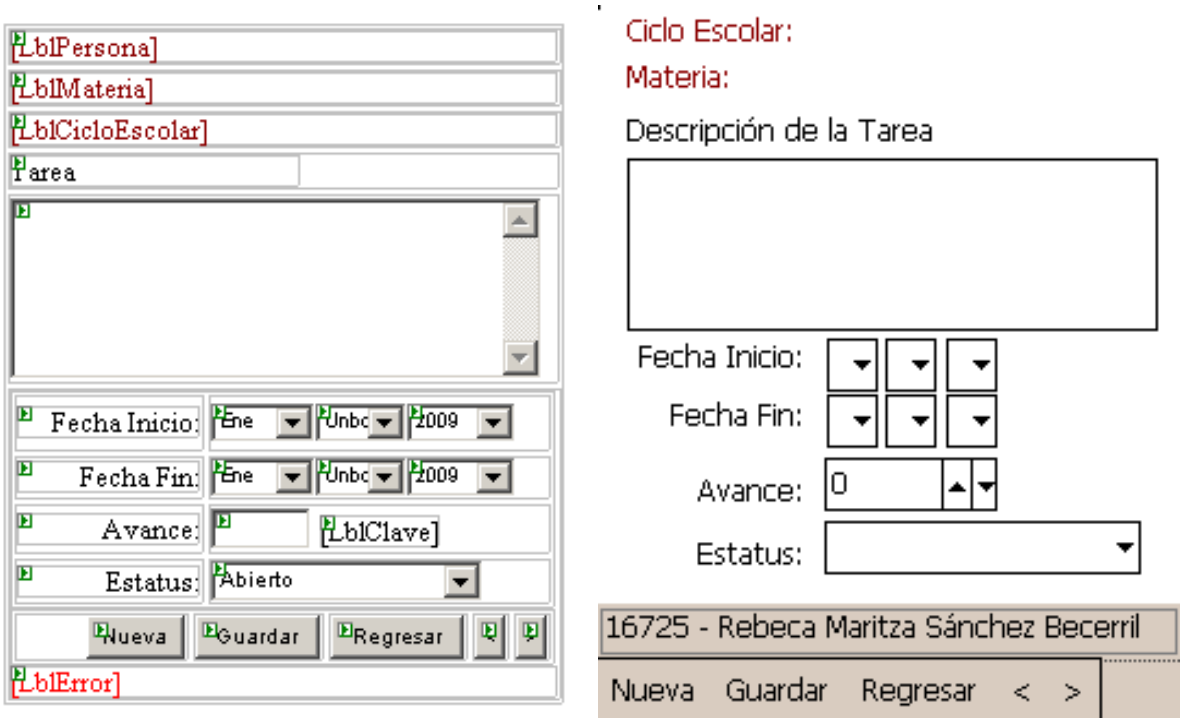

Figura 3.12 – Interfaz en tiempo de diseño (ConsultarTareas.aspx y ConsultarTareas.vb).

Validación de Administrador:

Nombres:
Aplicación Web: ContraAltaEstudiante.aspx
Aplicación Windows: AltaEstudiante.vb

Descripción:
Es la página/forma en donde se debe validar el usuario del sistema como administrador del mismo, de forma que sea de una manera segura el ingreso a la pantalla de alta de estudiantes nuevos.

La página/forma muestra un TextBox donde se solicita el ingreso del password de administrador.

En caso de que se desee continuar con el flujo del sistema, se debe oprimir el botón 'Aceptar', con lo que el sistema valida la contraseña ingresada contra la del administrador del sistema, si es correcta, el sistema pasará a la pantalla de 'Alta Estudiante'. En caso de que no se desee continuar, se debe oprimir el botón 'Regresar', con lo que el sistema regresará a la pantalla de 'Validación de Usuario'.

Figura 3.13 – Interfaz en tiempo de diseño (AltaEstudiante.aspx y AltaEstudiante.vb).

Alta Estudiante:

Nombres:
Aplicación Web: AltaEstudiante.aspx
Aplicación Windows: AltaEstudiante.vb

Descripción:
Es la página/forma en donde se puede dar de alta nuevos estudiantes al sistema, creando su perfil con sus datos y una contraseña.
La página/forma muestra controles donde se escribirán los datos de la persona que se dará de alta en el sistema, los cuales son:

- Matricula
- Nombre
- Apellido Paterno
- Apellido Materno
- Correo Electrónico
- Contraseña
- Confirmación de contraseña

En caso de que se desee confirmar la creación del nuevo usuario en el sistema, se debe oprimir el botón 'Aceptar', con lo que el sistema manda a llamar al método 'ValidaExisteAlumnoWS' del Web Service.
Para salir de la pantalla, se deberá oprimir el botón 'Regresar'.

Figura 3.14 – Interfaz en tiempo de diseño (AltaEstudiante.aspx y AltaEstudiante.vb).

Todas las páginas ya descritas conforman a la aplicación Web, la cual deberá alojarse en un servidor Web con conexión a internet y al servidor en donde esté alojado el Web Service, dando salida a que cualquier persona pueda conectarse a la aplicación Web desde sus dispositivos móviles.

Para la aplicación Windows se han creado todas las formas ya descritas, las cuales deberán ser compiladas por Visual Studio.Net y creando un archivo ejecutable, el cual con la ayuda de Active Sync se podrá sincronizar el dispositivo y la PC e instalar el programa en el dispositivo móvil. Este dispositivo necesita contar con los recursos de hardware y software suficientes para soportar la aplicación, además que necesita contar con una conexión a internet y al servidor que aloja el Web Service que la aplicación utilizará.

3.3 Pruebas al Sistema

Una vez que se ha desarrollado el sistema, se procede a realizar las pruebas, para lo que se necesita configurar un poco la aplicación así como sus conexiones.

Primeramente se deberá colocar la base de datos en un servidor con SQL Server, posteriormente se deberá montar sobre un servidor Web el Web Service, teniendo salida a

internet (o a red interna donde corra la aplicación) y conexión con la base de datos (autorizada por SQL), colocando en el IIS local del servidor (véase figura 3.15), la carpeta del Web Service y probando el Web Service probándolo en una ventana de algún navegador de internet, tal como lo muestra la figura 3.16 y 3.17.

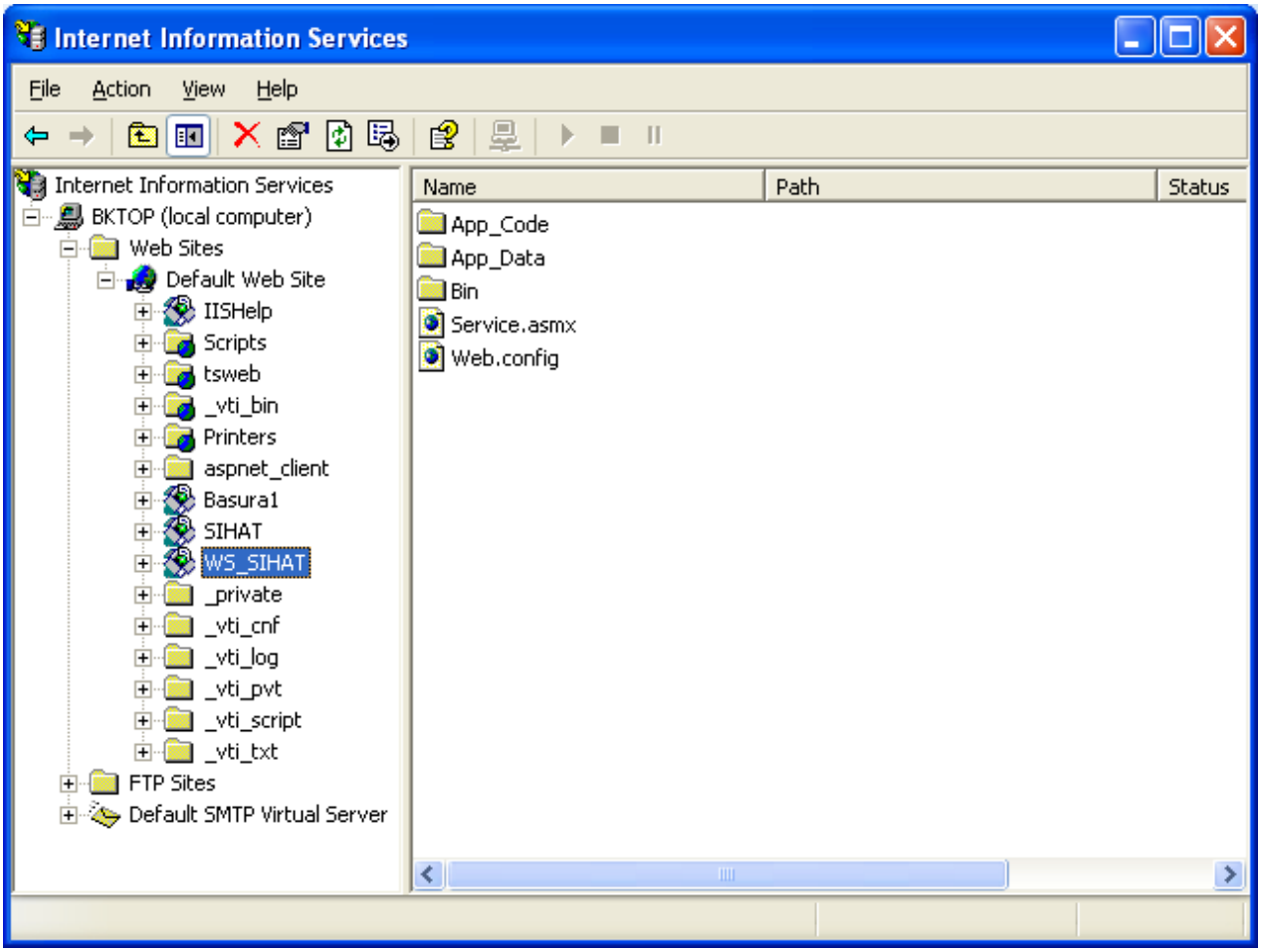

Figura 3.15 – Vista del IIS alojando el Web Service.

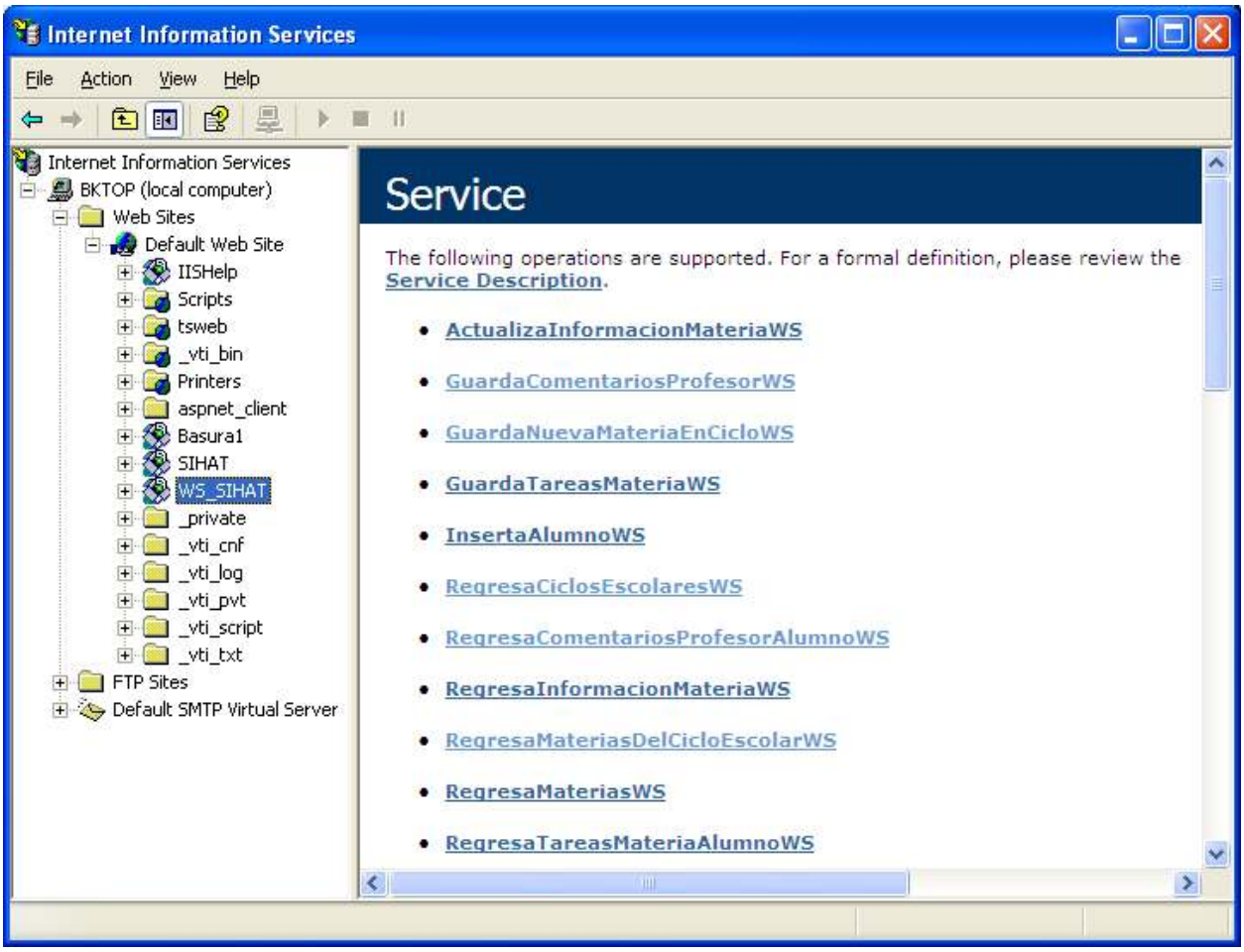

Figura 3.16 – Funciones del Web Service.

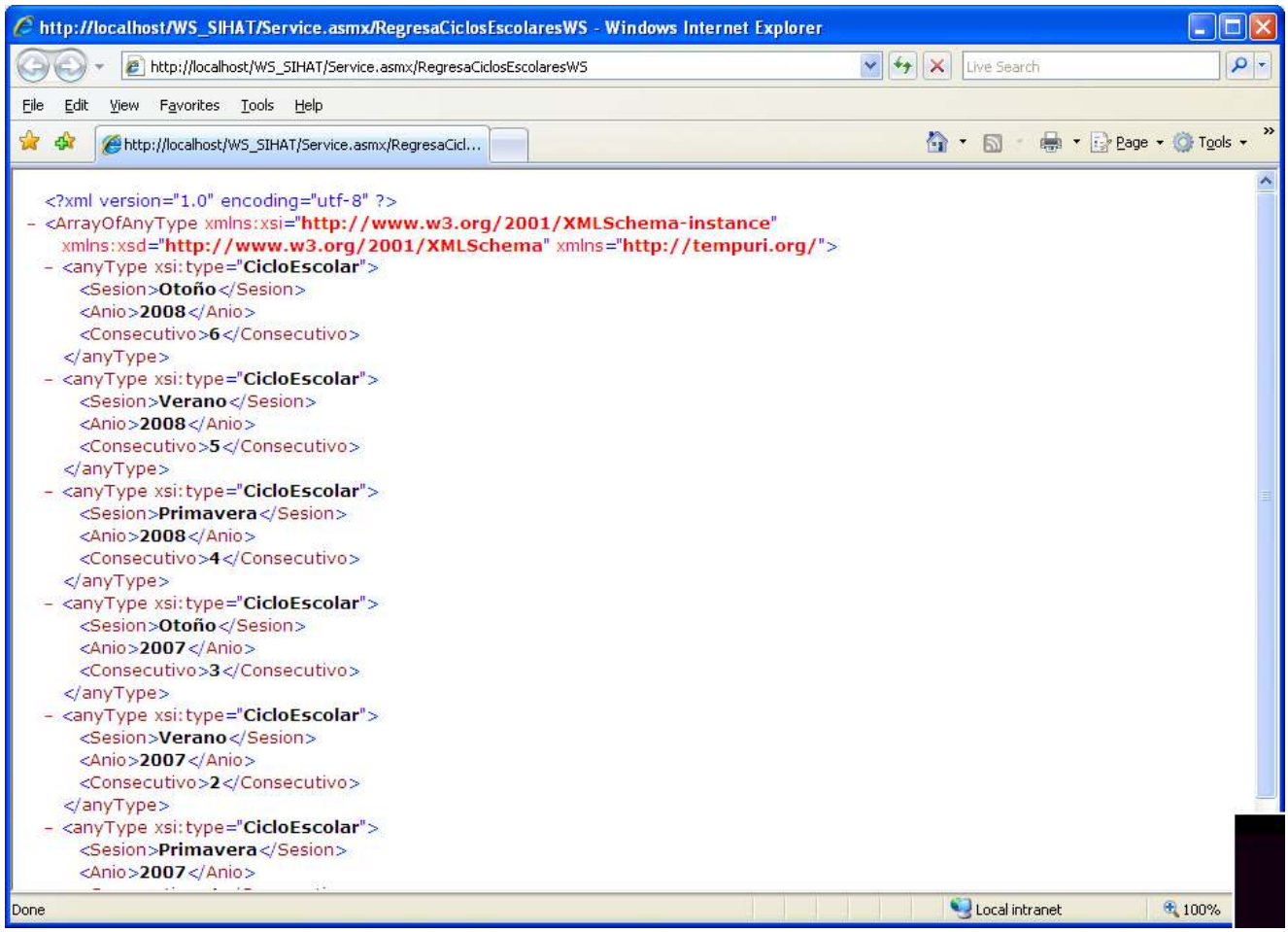

Figura 3.17 – Ejecución del Web Service mostrando resultados en XML.

De la pantalla que muestra la figura 3.16 se debe tomar la dirección que el Web Service toma, cambiando (si es necesario) localhost por el nombre o la dirección IP del servidor, de modo que se pueda acceder desde cualquier otra computadora. A esta dirección se le llamará de ahora en adelante 'Dirección del Web Service'.

Posteriormente se debe colocar en el servidor (puede ser el mismo u otro) la aplicación Web, colocándola en el IIS de la misma forma que se puso el Web Service (véase figura 3.18) y cambiando en el código del archivo Web.Config, la dirección de la referencia Web que se está añadiendo, escribiendo la dirección del Web Service que ya fue montado sobre el servidor (véase figura 3.19). Con esto se está indicando la dirección en donde la aplicación Web debe buscar al Web Service y hacer uso de sus funciones.

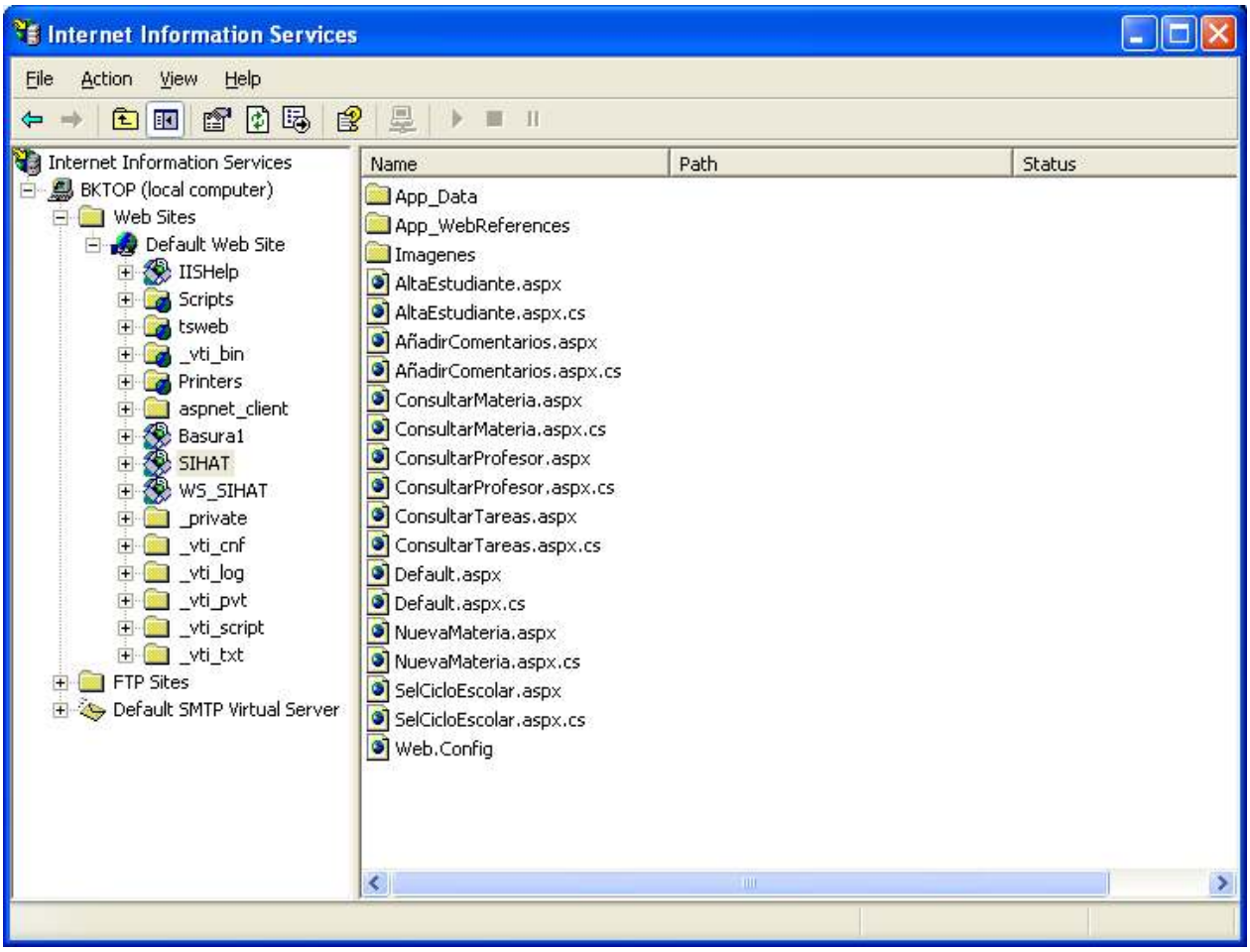

Figura 3.18 – Vista del IIS alojando la aplicación Web.

```
<configuration>
    <appSettings>
        <add key="WSRef.Service" value="http://localhost/WS_SIHAT/Service.asmx"/>
    </appSettings>
    <connectionStrings/>
```

Figura 3.19 – Configuración del Web.Config para indicar la dirección del Web Service.

Posteriormente, se procede a configurar la aplicación Windows, señalando la referencia Web a la dirección del Web Service que ya está montado en el servidor.

Ahora, ya que se tiene configurado el Web Service, la aplicación Web y la aplicación Windows, se procederá a configurar de manera correcta el emulador de PocketPC en el que se probarán las aplicaciones. Para esto, se procede a abrir el ActiveSync y el Visual Studio, abriendo en Tools → Device Emulator Manager el manejador de emuladores, que nos presenta una lista de emuladores que contiene Visual Studio (véase figura 3.20); para estas aplicaciones se utilizará el emulador Pocket PC 2003 SE Emulator.

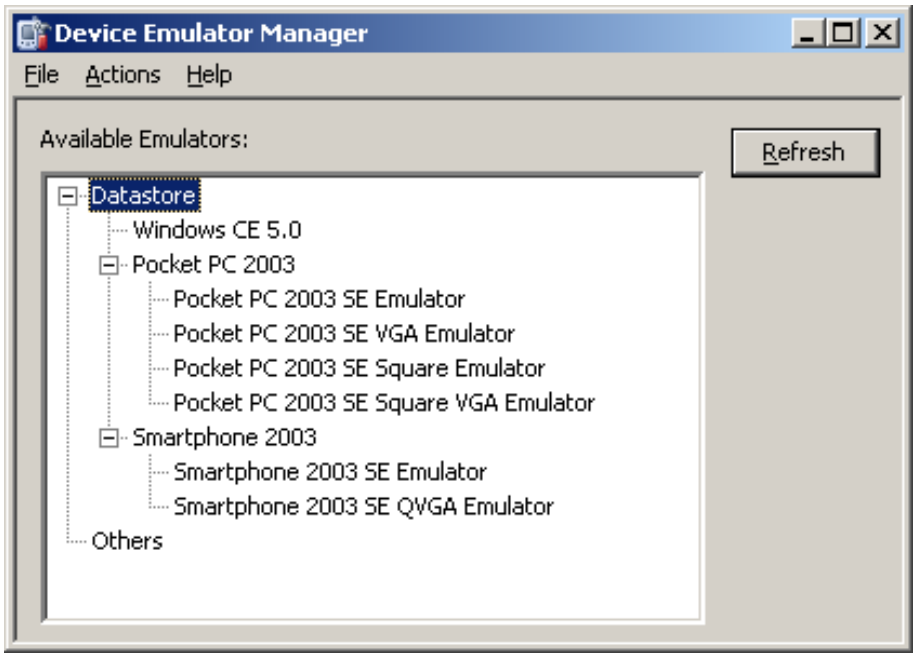

Figura 3.20 – Vista del Device Emulator Manager.

Se procede a conectar el Pocket PC 2003 SE Emulator, realizando las siguientes acciones:

1) Teniendo ActiveSync abierto y sin estar conectado a algún otro punto, se hace clic derecho sobre el emulador, seleccionando la opción 'Connect' automáticamente se abre una ventana con el emulador seleccionado listo para usarse, exceptuando que no está conectado a ninguna red. (Figura 3.21)

2) El emulador se debe conectar a la red para poder consultar la aplicación Web y el Web Service, por lo que se procede a hacer clic derecho en la ventana de 'Device Emulator Manager' sobre el dispositivo que tenemos conectado y seleccionamos la opción 'Cradle' con lo que automáticamente se activa ActiveSync y se selecciona el tipo de conexión a la red que deseamos que tenga el dispositivo. Cuando ActiveSync termina de configurar está listo para usar y podemos probar las aplicaciones. (Figura 3.22)

Figura 3.21 – Vista del proceso de conexión del emulador (Emulador no conectado).

Figura 3.22 – Vista del proceso de conexión del emulador (Emulador conectado).

Ahora, manejando únicamente el emulador, se abre una ventana del navegador de internet y se escribe en la barra de direcciones la dirección del Web Service que ya se tiene configurado, esto, con el objetivo de comprobar la conexión a la red y al servidor. (Figura 3.23).

Figura 3.23 – Consulta del Web Service desde el emulador.

Ahora, se procede a ejecutar la aplicación Web, por lo que se tendrá que escribir en la barra de direcciones, la dirección de la aplicación Web que está alojada en el servidor. Al mostrarse la página, se procederá con el uso del sistema. (Figura 3.24 a la 3.28)

Figura 3.24 – Interfaz de validación de usuario (Default.aspx) y selección del ciclo escolar (SelCicloEscolar.aspx) de la aplicación Web en ejecución.

Figura 3.25 – Interfaz de añadir materia (NuevaMateria.aspx) y consultar materia (ConsultarMateria.aspx) de la aplicación Web en ejecución.

Figura 3.26 – Interfaz de consulta de tarea (ConsultarTareas.aspx) y consultar profesor (ConsultarProfesor.aspx) de la aplicación Web en ejecución.

Figura 3.27 – Interfaz de consultar comentario (AñadirComentarios.aspx) y Alta de estudiante (AltaEstudiante.aspx) de la aplicación Web en ejecución.

Figura 3.28 – Interfaz de validar Administrador (ContraAltaEstudiante.aspx)
de la aplicación Web en ejecución.

Después de que se ha probado el sistema en la aplicación Web, revisando todas sus funcionalidades, se procede a ejecutar la aplicación Windows, por lo que para efectos de pruebas, esta última se probará en el mismo emulador, teniendo la misma funcionalidad que la aplicación Web, pero ahora como un ejecutable dentro del dispositivo móvil.

Para realizar la ejecución de la aplicación Windows en el emulador de PocketPC, se deberá tener la aplicación abierta en Visual Studio y oprimir el botón que ejecuta las aplicaciones, con esto Visual Studio muestra la venta de 'Deploy' (Figura 3.29) en donde podemos seleccionar el emulador deseado.

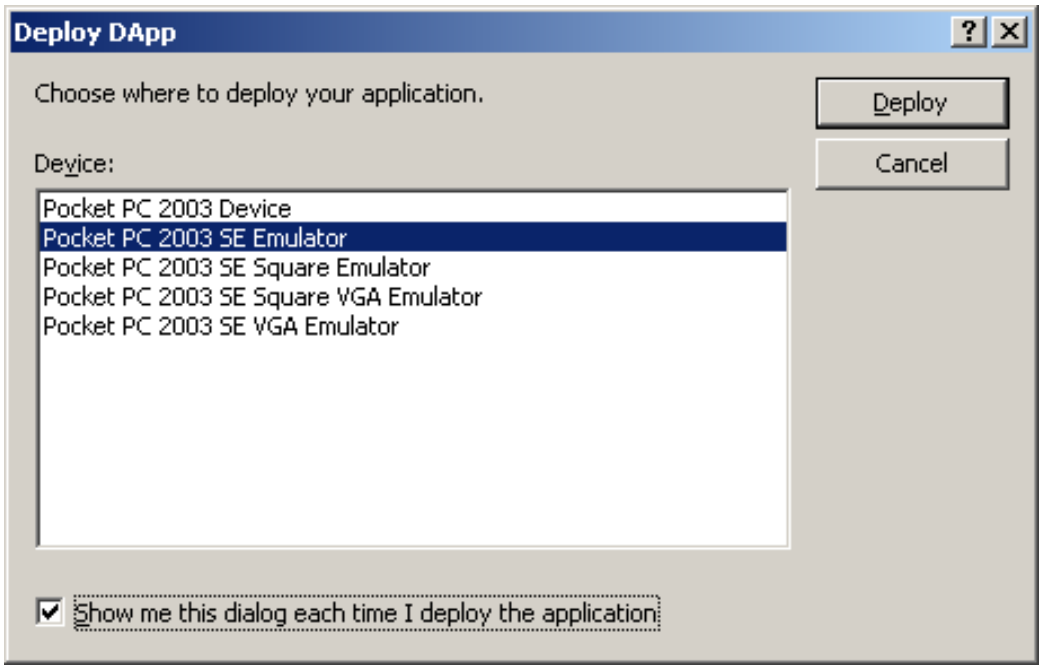

Figura 3.29 – Pantalla de Deploy donde se seleccionará el emulador para la aplicación Windows.

Una vez que se ha seleccionado el emulador, Visual Studio procede a abrirlo (si es que ya está abierto, ejecutará el programa sobre el ya abierto) y manda el archivo .exe al emulador del dispositivo, abriendo la aplicación Windows, para esto, tenemos que tomar en cuenta que también se tiene que realizar la misma configuración para la conexión a la red que se hiso con el emulador para probar la aplicación Web. Una vez realizada la conexión y cuando Visual Studio cargó la aplicación en el dispositivo móvil, se procederá a probar el funcionamiento de la aplicación dentro del emulador de PocketPC, tal como lo muestran las figuras 3.30 a la 3.34.

Figura 3.30 – Interfaz de validación de usuario (Default.vb) y selección de ciclo escolar (SelecCicloEscolar.vb) de la aplicación Windows en ejecución.

Figura 3.31 – Interfaz de añadir nueva materia (NuevaMateria.vb) y pantalla para la consulta de la información de una materia (ConsultarMateria.vb) de la aplicación Windows en ejecución.

Figura 3.32 – Interfaz de consulta de tareas (ConsultarTareas.vb) y consulta de profesor (ConsultarProfesor.vb) de la aplicación Windows en ejecución.

Figura 3.33 – Interfaz de captura de un nuevo alumno (AltaEstudiante.vb) y consulta de comentarios del profesor (ComentariosProfesor.vb) de la aplicación Windows en ejecución.

Figura 3.34 – Interfaz de validación del Administrador (ContraAltaEstudiante.vb) de la aplicación Windows en ejecución.

Con esto, se puedo comprobar el funcionamiento de las aplicaciones dentro de un dispositivo móvil, en este caso se implementó sobre un emulador de PocketPC, mostrando cómo ambas aplicaciones funcionan correctamente, conectándose a un Web Service y trabajando con una arquitectura N-Tiers y almacenando la información en una base de datos SQL.

Capítulo 4

Conclusiones y Perspectivas

4.1 Conclusiones

El desarrollo de aplicaciones y sistemas sobre dispositivos móviles es cada vez más valioso en el ámbito laboral y es por ello que esta tesis se propone como aportación de muchas investigaciones posteriores para los estudiantes de la Facultad de Ciencias de la Computación sobre el desarrollo para móviles o las herramientas que se utilizaron, como Web Services XML, arquitecturas N-Tiers, bases de datos, etc.

Este trabajo presenta los resultados de una investigación sobre los lenguajes, tecnologías y herramientas utilizados para la programación de dispositivos móviles, en particular se presentaron las tecnologías que actualmente ofrece Microsoft para el desarrollo sobre estos dispositivos, así como también se presentó la arquitectura N-Tiers, los lenguajes de programación de .Net, la creación y uso de los Web Services XML así como también el uso de las base de datos utilizando SQL Server, todo con el objetivo de mostrar la forma en cómo cualquier programador puede realizar un sistema distribuido destinado a dispositivos móviles.

Se realizaron dos aplicaciones de diferentes tipos, con la misma funcionalidad y para los mismos clientes (dispositivos móviles), la primera de estas aplicaciones fue un sistema de administración personal de historial académico destinado a correr sobre dispositivos móviles, el cual se puede consultar vía Web y la segunda es el mismo sistema pero en versión de aplicación Windows,

instalable y ejecutable sobre el mismo dispositivo móvil. Esto básicamente se logró a partir de la creación de un Web Service XML, que es el centro de este sistema, pues actúa como intermediario entre los clientes (las aplicaciones) y la base de datos, pasando también por la arquitectura de N-Tiers, hablando en lenguaje XML que es entendible por otros muchos sistemas.

Finalmente se puede indicar que esta investigación y proyecto conllevará a la realización de sistemas más complejos, los cuales se podrán utilizar ampliamente por empresas y personas, además que también se considerará la creación de aplicaciones para móviles como complemento para cualquier otro sistema creado para computadoras de escritorio, logrando así sistemas más interesantes, más útiles y que den tanto al usuario como al desarrollador satisfacción por el trabajo hecho.

4.2 Perspectivas

Se ha desarrollado este trabajo de tesis proyectándolo a un futuro en la investigación y desarrollo de tecnologías, sistemas y aplicaciones de móviles, especialmente para los estudiantes de la B.U.A.P., para que puedan obtener más conocimiento del área de desarrollo y las oportunidades de trabajo aumenten, así como también se pueda tomar el desafío de crear una empresa de desarrollo en donde se consideren este tipo de dispositivos, los cuales van en aumento y el desarrollo enfocado a móviles, aun es muy poco, pues son pocas las personas que lo hacen.

Hablando ahora de las aplicaciones que este proyecto de tesis presenta, se pueden mejorar para añadir un mayor interés al usuario en utilizarlas.

Algunas de las posibles mejoras son las siguientes:

- Ampliación del diseño y capacidades del sistema.
- Creación de un módulo para administradores del sistema.
- Definir métricas que aumenten la seguridad en el sistema.
- Creación de pantallas para profesores, en donde se puedan asignar tareas a sus alumnos.
- Optimización del tiempo de consulta y respuestas.
- Creación de perfiles de dispositivos, en donde la aplicación construya sus interfaces de acuerdo al tipo de dispositivo que lo consulta.

Apéndice A

Sistemas Operativos

El sistema operativo es un programa (o conjunto de programas) de control que tiene por objetivo gestionar y administrar eficientemente los recursos hardware, permitiendo que se ejecuten concurrentemente varios programas, sin que haya conflictos en el acceso de cada uno de ellos a cada uno de los recursos que necesite, y sin que ningún programa monopolice alguno de ellos. [7]

A continuación, se listan los más populares actualmente:

- **Windows Mobile:** Es un sistema operativo compacto, con una suite de aplicaciones básicas para dispositivos móviles basados en la API Win32 de Microsoft. Los dispositivos que llevan Windows Mobile son Pocket PC's, Smartphone's, Black Berry´s y Media Center portátil. Ha sido diseñado para ser similar a las versiones de escritorio de Windows.

  Tanto Windows Mobile para Pocket PC, como Windows Mobile para Smartphone's, poseen bastantes aspectos parecidos, pero en cuanto a aplicaciones desarrolladas, una aplicación no servirá para ambas versiones Windows Mobile PPC (Pocket PC) y Windows Mobile Sp (Smartphone), por lo que generalmente se debe programar específicamente para el dispositivo en que se ejecutará la aplicación.

  Windows Mobile 6, antes con el nombre en clave *Crossbow* es la última versión de la plataforma Windows Mobile y fue lanzado el 12 de febrero del 2007 en el 3GSM World Congress 2007. Ofrece tres versiones: Windows Mobile 6 Standard para Smartphone's (teléfonos sin pantalla táctil), Windows Mobile 6 Professional para PDAs con la funcionalidad del teléfono (Pocket PC Phone Edition), y Windows Mobile 6 Classic para PDAs sin radios celulares. [6]

- **Windows CE:** Es el sistema operativo de Microsoft incrustado modular de tiempo real para dispositivos móviles de 32-bits inteligentes y conectados. Windows CE combina la compatibilidad y los ping a servicios de aplicación avanzados de Windows con soporte para múltiples arquitecturas de CPU y opciones incluidas de comunicación y redes para proporcionar una fundación abierta para crear una variedad de productos.

  Esta plataforma modular permite a los desarrolladores crear software para que la nueva generación de dispositivos móviles de 32-bits se integre con Windows e Internet.

  Windows CE no es un subconjunto de Windows XP, o de Windows NT, sino que fue desarrollado a base de nuevas arquitecturas y una nueva plataforma de desarrollo. Aun así mantiene cierta conexión con sus hermanos. Windows CE tiene sus propias APIs para desarrollo, y necesita sus propios drivers para el hardware con el cual va a interactuar.

  Windows CE también ha permitido la creación de un sistema denominado AutoPC, que consiste en un PC empotrado en un automóvil y que va ubicado donde normalmente va una radio. De esta manera permite controlar la radio, el reproductor de CD y revisar el correo electrónico. Windows CE también permite la creación de aplicaciones en tiempo real.

  La última versión del Windows CE actualmente es Windows Mobile 6.1, Upgrade de Windows Mobile 6.0, sucesor de Windows Mobile 5.0, y sirve tanto para Pocket PC (PDA) como para Smartphone.

  Cabe destacar que este sistema operativo es el único producto de Microsoft que se distribuye junto con el código fuente (tal como Linux) y usa una licencia llamada Shared Source, así pues permite al usuario final modificar el código fuente sin notificar al propietario. [6]

- **Palm OS:** Es el más difundido en el mundo ya que entró primero en el mercado. Permite correr varios programas a la vez y cuenta con buena velocidad de operación. Como ventajas están, que es libre y que es de fácil uso.

  Actualmente son muchos los fabricantes como Oracle, Nokia, Handspring, Symbol y Sony que utilizan diversas variantes y versiones de este sistema operativo que en conjunto representan el 66% de todos los sistemas instalados en dispositivos de cómputo móviles.

Según la filosofía de Palm, ellos intentan tratar a la computación móvil no como versiones en miniatura de los sistemas de escritorio, sino como dispositivos y aplicaciones dedicados a tareas y usos que tienen su propia identidad y reclaman sus propios recursos y soluciones.

En los últimos años, la versión más extendida ha sido la 4.1 que entre sus principales características, presenta el soporte "teórico" de 65 mil de colores así como la gestión de tarjetas de memoria externa. Una mejora significativa ha sido la posibilidad de manejar resoluciones de 160 x 160 a 320 x 320 píxeles.

Cualquiera que sea la conexión inalámbrica utilizada, WiFi Lan, Bluetooth, GSM/GPRS, o CDMA, el sistema Palm OS 5 integra las API necesarias. O sea, que los dispositivos equipados con Palm OS 5 pueden comunicarse fácilmente con todos los dispositivos existentes que estén basados en esas normas tales como teléfonos móviles, impresoras, módems, etc.

Las normas de seguridad incorporadas en el sistema, permiten que las transacciones sean hechas de forma segura, asegurando, también, el uso de firmas digitales homologadas. También ofrece servicios de encriptación Secure Sockets Layer de 128 bit. (SSL 3.0/TSL 1.0) para las conexiones.

El sistema incluye asimismo un browser para Internet, el NetFont que suporta entre otras normas, HTML 4.01, XHTML los GIF animados, el modo seguro de acceso a la red VPN (Virtual Private Network) y la interpretación de código JavaScript. Estas normas ya utilizadas en los sistemas de las computadoras de escritorio se introducen por primera vez en los equipos de mano.

- **Linux:** Es un sistema operativo compatible con UNIX. Dos características muy peculiares lo diferencian del resto de los sistemas más extendidos en el mercado, la primera, es que es libre, la segunda, es que el sistema viene acompañado del código fuente.

  LINUX se distribuye bajo la licencia pública del proyecto GNU que fue lanzado en 1984 para desarrollar el Linux de libre distribución.

  El sistema ha sido diseñado y programado por multitud de programadores alrededor del mundo. El núcleo del sistema sigue en continuo desarrollo. En los últimos tiempos, ciertas compañías de software comercial han empezado a distribuir sus productos para Linux y la presencia del mismo en empresas aumenta rápidamente por la excelente relación calidad-precio que se consigue.

Linux es multitarea lo que se refiere a la capacidad de ejecutar varios programas al mismo tiempo. LINUX utiliza la llamada multitarea preventiva, la cual asegura que todos los programas que se están utilizando en un momento dado serán ejecutados, siendo el sistema operativo el encargado de ceder tiempo de microprocesador a cada programa.

En los últimos años, algunos fabricantes de dispositivos móviles han incorporado Linux a sus productos. Se están desarrollando versiones de Embedded Linux que constituyen otra alternativa a Palm OS y Windows CE para los dispositivos de cómputo Móviles. Así, LinuxDevices.com, ha creado una guía de referencia para computadores de mano basados en Linux, con la que pretende mantener actualizados de manera permanente los productos Linux para móviles.

- **Symbian:** Este sistema operativo está diseñado para teléfonos móviles capaces de gestionar datos y algunas aplicaciones. La mayoría de los actuales teléfonos avanzados, o SmartPhones utilizan las versiones 6.1 o 7.0 del sistema operativo Symbian. Pero Symbian 8.0, incluye mejoras significativas para Java y actividades multimedia.

  Es un sistema operativo abierto, está diseñado para los requerimientos específicos de los teléfonos móviles de 2G, 2.5 y 3G. En las versiones 6.X de este sistema se tiene soporte para desarrollo en C++, Java, WAP, HTML.

  Entre las principales características de Symbian OS 9 se destaca el envío de música a auriculares a través de tecnología inalámbrica, gráficos 3D con mejor calidad, funciones de seguridad mejoradas y compatibilidad con sistemas de cámaras digitales de hasta 2 Millones de píxeles.

  Symbian está incorporado en unos 20 millones de teléfonos móviles, esto proporciona una enorme base, y en expansión, para los desarrolladores de software para móviles, que además deben tener en cuenta el sistema operativo Symbian.

  Symbian OS se ejecuta aproximadamente en 30 modelos de teléfonos de fabricantes como Nokia, Sony Ericsson, Motorola y Samsung.

- **iPhone OS:** El iPhone OS es el sistema operativo que utiliza el iPhone y el iPod touch. Está basado en una variante del Mach kernel que se encuentra en Mac OS X. El iPhone OS incluye el componente de software “Animación Core” de Mac OS X v10.5 que, junto con el PowerVR MBX el hardware de 3D, es responsable de las animaciones usadas en el interfaz de usuario.

IPhone OS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de Servicios Principales, la capa de Medios de comunicación y la capa de Cocoa Touch. El sistema operativo ocupa bastante menos de medio gigabyte del total del dispositivo, de 8 GB o de 16 GB.[24] Esto se realizó para poder soportar futuras aplicaciones de Apple, así como aplicaciones de terceros publicadas en la iTunes Store o la AppStore.

Este sistema operativo no tenía un nombre oficial hasta que salió la primera versión beta del iPhone SDK, el 6 de marzo de 2008. Antes de esto, Apple declaró, que "el iPhone controla un sistema operativo X", una referencia al padre de los sistemas operativos de los iPhone, el Mac OS X.

Como un iPod, el iPhone se maneja con la versión 7.3 o superior de iTunes, el cual es compatible con Mac OS X versión 10.4.10, y con Windows XP o Windows Vista de 32 bits.[25] A partir de la versión 7.6 de iTunes se incluyeron las versiones de 64 bits del Windows XP y del Vista,[26] y se han facilitado parches para versiones anteriores de Windows.[27]

Las aplicaciones del iPhone no se pueden copiar directamente de Mac OS X ya que tienen que ser escritas y compiladas específicamente para el iPhone. Además, el navegador Safari soporta aplicaciones web escritas con AJAX.

Apéndice B

UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

Diagramas

En UML 2.0 hay 13 tipos diferentes de diagramas, los cuales se clasifican en:

Diagramas de Estructura: Enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes

- Diagrama de objetos
- Diagrama de estructura compuesta
- Diagrama de despliegue
- Diagrama de paquetes

Diagramas de Comportamiento: Enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de estados

Diagramas de Interacción: Son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de tiempos
- Diagrama de vista de interacción

A continuación, se describirán brevemente cada uno de los diferentes diagramas que define UML.

Diagrama de Clases: Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Diagrama de Componentes: Un diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo archivos, cabeceras, módulos, paquetes, etc.) y muestra las dependencias entre estos componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. En él situaremos librerías, tablas archivos, ejecutables y documentos que formen parte del sistema. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

Diagrama de Objetos: Los diagramas de objetos son utilizados durante el proceso de Análisis y Diseño de los sistemas informáticos en la metodología UML. Se puede

considerar un caso especial de un diagrama de clases en el que se muestran instancias específicas de clases (objetos) en un momento particular del sistema. Los diagramas de objetos utilizan un subconjunto de los elementos de un diagrama de clase. Los diagramas de objetos no muestran la multiplicidad ni los roles, aunque su notación es similar a los diagramas de clase.

Diagrama de Estructura Compuesta: El diagrama de estructura compuesta toma el modelo para describir las relaciones entre los elementos para trabajar junto a una clasificación. Este es similar al diagrama de clase, pero muestra partes y conectores. Las partes no son necesariamente clases en el modelo y ellos no representan las instancias particulares, pero ellos pueden tener roles donde las clasificaciones pueden jugar.

Las partes son mostradas de una manera similar a los objetos. El diagrama de estructura compuesta es usado para mostrar el "runtime" de la arquitectura de cualquier tipo de clasificación.

Diagrama de Despliegue: El Diagrama de Despliegue es un tipo de diagrama que se utiliza para modelar el hardware utilizado en la implementación de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones. En el UML 2.0 los componentes ya no están dentro de nodos. En cambio, puede haber artefactos u otros nodos dentro de un nodo. Un artefacto puede ser algo como un archivo, un programa, una biblioteca, o una base de datos construida o modificada en un proyecto. Estos artefactos implementan colecciones de componentes. Los nodos internos indican ambientes, un concepto más amplio que el hardware propiamente dicho, ya que un ambiente puede incluir al lenguaje de programación, a un sistema operativo, un ordenador o un cluster de terminales.

Diagrama de Paquetes: Un diagrama de paquetes muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Con estas líneas maestras sobre la mesa, los paquetes son buenos elementos de gestión. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

Diagrama de Actividades: Un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un diagrama de actividades muestra el flujo de control general.

Diagrama de Casos de Uso: Un diagrama de casos de uso es una especie de diagrama de comportamiento. El lenguaje de modelado unificado define una notación gráfica para representar casos de uso llamada modelo de casos de uso. El valor verdadero de un caso de uso reposa en dos áreas:

- La descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio. Esta descripción se enfoca en el valor suministrado por el sistema a entidades externas tales como usuarios humanos u otros sistemas.

- La posición o contexto del caso de uso entre otros casos de uso. Dado que es un mecanismo de organización, un conjunto de casos de uso coherente y consistente promueve una imagen fácil del comportamiento del sistema, un entendimiento común entre el cliente/propietario/usuario y el equipo de desarrollo.

Diagrama de Estados: Los diagramas de estados se usan para representar gráficamente máquinas de estados finitos. Las tablas de transiciones son otra posible representación. Hay muchas formas de diagramas de estados que difieren levemente y tienen semánticas diferentes.

Diagrama de Secuencia: El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

Diagrama de Colaboración: Es un diagrama que muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran explícitamente las relaciones de los roles. Por otra parte, un diagrama de colaboración no muestra el tiempo como una dimensión aparte, por lo que resulta necesario etiquetar con números de secuencia tanto la secuencia de mensajes como los hilos concurrentes.

Diagrama de Tiempos: Un diagrama de tiempos o cronograma es una gráfica de formas de onda digitales que muestra la relación temporal entre varias señales, y cómo varía cada señal en relación a las demás. Un cronograma puede contener cualquier número de señales relacionadas entre sí. El propósito primario del diagrama de tiempos es mostrar los cambios en el estado o la condición de una línea de vida (representando una Instancia de un Clasificador o un Rol de un clasificador) a lo largo del tiempo lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o estímulos aceptados. Los eventos que se reciben se anotan, a medida que muestran cuándo se desea mostrar el evento que causa el cambio en la condición o en el estado.

Vista de Interacción: Representa cómo interactúan cooperativamente los objetos para implementar el comportamiento definido por los casos de uso.

Apéndice C

Paso de Información entre Páginas Web

Para la programación de páginas ASP.NET, en ocasiones, dependiendo de la funcionalidad de la página, es necesario el envío de datos, información o parámetros entre las diversas páginas que conformen un sistema web, por lo que ASP.NET dispone de cuatro diferentes formas de pasar información con ASP.NET, los cuales son actualmente utilizados por desarrolladores, teniendo ventajas y desventajas cada uno de estos métodos, no se puede definir que alguno de ellos sea el mejor o peor, eso ya depende del propio sistema, el uso que tendrá y las preferencias del desarrollador hacia estos métodos.

a) Cookies

Las *cookies* son pequeños archivos de texto que envía el servidor web y que se almacenan en el disco duro del usuario. Por este motivo hay gente que las considera peligrosas y las tiene desactivadas en su navegador, en este caso, no se puede usar las cookies, pues el navegador las ignorará.

Normalmente las cookies se utilizan para guardar datos que pueden interesar para la próxima vez que el usuario se conecte al web, como por ejemplo el idioma seleccionado, de esta forma la próxima vez que se visite la página el servidor buscará la cookie que guarda el idioma y automáticamente se verán las páginas en el último idioma que se ha visitado y no se tendrá que volver a seleccionar el idioma.

Las cookies son una colección del objeto **Response** de ASP.NET.

```
<%
 Response.Cookies("nombre") = "Marcos"
 Response.Cookies("micookie")("valor1") = "el valor que quiera"
 Response.Cookies("micookie")("valor2") = "otro valor"
%>
```

Las cookies también son una colección del objeto **Request** de ASP.NET, en este caso para leer su valor.

```
<%
 Dim nombre
 Dim valor1
 nombre = Request.Cookies("nombre")
 valor1 = Request.Cookies("micookie")("valor1")
%>
```

A partir de los ejemplos anteriores se puede ver que es muy sencillo almacenar valores en las cookies y recuperarlos, de esta forma desde una página se pueden almacenar valores y recuperarlos desde cualquier otra página o modificarlos.

Además la colección **Cookies** tiene otras propiedades como la fecha de caducidad. A continuación se muestra un pequeño resumen de sus propiedades:

```
  <%
  Response.Cookies("miCookie") = "Hola caracol!" 'Asignamos un valor

  Response.Cookies("miCookie").Expires = "01/01/2010" 'Indicamos la
fecha de caducidad.

  Response.Cookies("miCookie").ExpiresAbsolute = "Febrero 14, 2001"
  'Si no se indica una fecha de caducidad posterior a la actual la
Cookie caducará al finalizar la sesión.

  Response.Cookies("miCookie").HasKeys 'Indica si la cookie tiene o no
claves. Esta propiedad es de sólo lectura.

  Response.Cookies("miCookie").Path = "/"
  'Si se especifica, la cookie sólo se envía a las peticiones de esa
ruta.
  'Si no se especifica, se utiliza la ruta de la aplicación.

  Response.Cookies("miCookie").Secure = True 'Indica si la cookie es
segura.
  %>
```

b) Variables Sesión

Además de las cookies podemos utilizar variables del servidor para almacenar valores y leerlos desde nuestras páginas ASP.NET, de forma que podremos comunicar datos de unas páginas a otras.

```
  <%
  Session("miVariable") = "Hola caracol!" 'Almacenamos un valor en una
variable
  Response.write Session("miVariable") 'Mostramos el valor de una
variable
  %>
```

Hay que tener en cuenta que puesto que se trata de variables del servidor se almacenarán en la memoria del servidor, por lo que no conviene abusar de ellas para no ocupar recursos innecesarios del servidor.

Los valores de estas variables se mantendrán mientras dure la sesión, cuando ésta caduque, los valores se eliminarán y ya no serán accesibles. Por default la duración de la sesión es de 20 minutos, esto se puede variar. Cada vez que se carga una página del servidor se está renovando la sesión.

Además también podemos eliminar las variables del objeto session utilizando el método *Remove* o el método *RemoveAll* para eliminarlas todas a la vez.

```
<%
 Session.Contents.Remove("miVariable")
 'o bien
 Session.Contents.RemoveAll
%>
```

Si no queremos que el servidor realice un seguimiento de sesiones podemos deshabilitarlas con el siguiente código:

```
<%@ EnableSessionState=False %>
```

Un método importante del objeto Session es **Abandon**, este método destruye el objeto Session y libera sus recursos.

```
<% Session.Abandon %>
```

Nota: el uso del objeto Session requiere que el navegador del usuario tenga habilitadas las cookies, pues el servidor envía una cookie al principio de la sesión que será la que identifique al usuario durante el resto de la sesión.

c) Formularios: Metodos Get, Post y Query String

La otra opción para el paso de parámetros entre páginas ASP.NET es mediante formularios, y ésta es la única opción que no requiere el uso de cookies, por lo que será la más compatible para todos los usuarios, tengan o no navegadores que soporten cookies o aunque las tengan desactivadas.

El uso de formularios es obligatorio en el caso de que tengamos que pasar datos que un usuario ha introducido. Puesto que Netscape Navigator obliga a incluir la etiqueta <FORM> o no dibuja los elementos del formulario, incluso para un sencillo Combo Box es interesante utilizar este método.

Para pasar datos mediante formularios, hay que definir simplemente dos cosas:

- El destino del formulario (ACTION)
- El método de envío de los datos (METHOD)

El destino es la página ASP.NET que va a recoger los datos. El método, en cambio, especifica de qué forma se pasan los datos al script. El método POST pasa los datos de forma que el usuario no puede verlos, mientras que el método GET los muestra en la barra de direcciones, en forma de una QueryString, como:

http://www.aspfacil.com/formulario.aspx?nombre=Carlos+de+la+Ordeb&telefono=22334

Para obtener el valor de las variables se utiliza el método GET, se tendrá que usar la propiedad Request.Form (*parámetro*). Por ejemplo, supongamos la siguiente página:

uno.aspx

```
<HTML>
<BODY>
<FORM ACTION=dos.aspx METHOD=get>
Introduce tu nombre:
<INPUT TYPE="text" NAME="Nombre">
<BR>
<INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Que envía el nombre a "dos.aspx". Veamos esta página:

dos.aspx

```
<%
For each var in Request.QueryString
```

```
    Response.Write (var & ": " & Request.QueryString(var))
    Response.Write("<br>")
Next
%>
```

La segunda página recorre la QueryString completamente e imprime el nombre de cada parámetro y su valor. Si introdujésemos Carlos en el formulario, nos devolvería:

```
        Nombre: Carlos
```

Si utilizamos POST, en cambio, tendremos que usar en la página receptora la colección `Request.Form`. Al igual que la anterior, podemos recorrerla o bien seleccionar uno de los campos del formulario. Así, la página emisora sería:

```
<HTML>
<BODY>
<FORM ACTION=dos.aspx METHOD=POST>
Introduce tu nombre:
<INPUT TYPE="text" NAME="Nombre">
<BR>
<INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Y la página receptora:

```
<%
For each var in Request.Form
        Response.Write (var & ": " & Request.Form(var))
Next
%>
```

Que devolvería exactamente lo mismo.

Bibliografía

Sistemas Operativos

[1] http://www.microsoft.com/windowsmobile/developers/default.mspx
[2] Sistemas Operativos y Lenguajes de Programación, ENRIQUE QUERO CATALINAS, Edit. Paraninfo

.NET

[3] http://msdn.microsoft.com/es-es/library/
[4] http://es.wikipedia.org/wiki/Algoritmo_MD5
[5] Pro ASP.NET 2.0 in C# 2005; Autores: Matthew MacDonald and Mario Szpuszta; Edit. Apress. 2005.
[6] Microsoft Visual Studio 2008 – Unleashed; Autores: Lars Powers, Mike Snell; Edit: SAMS.

XML y Serialización

[7] http://msdn2.microsoft.com/es-es/library/182eeyhh(VS.80).aspx
[8] Pro ASP.NET 3.5 in C# 2008, Second Edition; Autores: Matthew MacDonald and Mario Szpuszta; Editorial: Apress.
[9] Understanding Web Services XML, WSDL, SOAP and UDDI. Autor: Eric Newcomer, Editorial: David Chappell series editor.
[10] Mobile Web Services Architecture and Implementation; Autores: Frederick Hirsch, John Kemp y Jani Ilkka.; Editorial: Wiley.

Modelo de 3 capas

[11] http://es.wikipedia.org/wiki/Arquitectura_software
[12] Patterns of Enterprise Application Architecture. Martin Fowler. Addison Wesley.
[13] Net Patterns, Architecture, Design and Process. Christian Thilmany. Addison Wesley

Aplicaciones Distribuidas

[14] http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml#DISEÑAN

[15] http://www.wikilearning.com/ejemplo_wap_hoteldirect_de-wkccp-3917-5.htm