INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

“DESARROLLO DE UN DIRECTORIO USANDO UN SERVICIO WEB”

T E S I S

Q U E P A R A O B T E N E R E L T Í T U L O D E:
INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

P R E S E N T A N

GARCIA ZAVALA JESICA XCHEL

MARTINEZ VAZQUEZ DANIEL

RIVERA CORONA DANTE JOSE MARIA

ING. CATALINA PATIÑO GALLEGOS

M. en C. GREGORIO GARCIA PEREZ

MÉXICO, D.F. 2009

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TITULO DE **INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**

POR LA OPCIÓN DE TITULACIÓN **TESIS Y EXAMEN ORAL INDIVIDUAL**

DEBERA(N) DESARROLLAR **JESICA XCHEL GARCIA ZAVALA**
DANIEL MARTINEZ VAZQUEZ
DANTE JOSE MARIA RIVERA CORONA

"DESARROLLO DE UN DIRECTORIO USANDO UN SERVICIO WEB

DISEÑAR UNA APLICACIÓN QUE UTILICE UN SERVICIO WEB PARA CONSULTAR, AGREGAR Y MODIFICAR DATOS DE PROFESORES DE LA BASE DE DATOS DEL DIRECTORIO DEL DEPARTAMENTO DE HORARIOS DE ESIME ZACATENCO.

- **INTRODUCCIÓN**
- **METODOLOGÍAS DE DESARROLLO**
- **SERVICIOS WEB**
- **BASE DE DATOS**
- **ANALISIS Y DIAGNÓSTICO**
- **DISEÑO E IMPLEMENTACIÓN**
- **CONCLUSIONES**

MÉXICO D. F., A 18 DE FEBRERO DE 2009.

A S E S O R E S

ING. CATALINA PATIÑO GALLEGOS

M. EN C. GREGORIO GARCÍA PÉREZ

M. EN C. RICARDO MENESES GONZÁLEZ
JEFE DEL DEPTO. ACADÉMICO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA

Agradecimientos

A nuestros Padres

A quienes nos han brindado el tesoro más valioso que puede dársele a un hijo. Amor. A quienes sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida, para formarnos y educarnos. A quienes la ilusión de su vida ha sido en convertirnos en personas de provecho. Por esto y más gracias.

A nuestros Hermanos

Por que además de ser nuestros mejores amigos, son la mejor compañía para compartir el mismo techo.

A nuestros Asesores y Profesores

Por toda su comprensión, disposición, paciencia, consejos y el apoyo incondicional que nos brindaron. Por participar en nuestro desarrollo profesional durante nuestra carrera, sin su ayuda y conocimientos no estaríamos en donde nos encontramos ahora.

A nuestros Amigos

Por su apoyo, por que estuvieron con nosotros compartiendo tantas aventuras, experiencias, desveladas y triunfos. Gracias a cada uno de ustedes por hacer que nuestra amistad sea duradera.

INDICE

Glosario Técnico

ADO.NET: ActiveX Data Objects .NET, Es una evolución del modelo de acceso a datos de ADO que controla directamente los requisitos del usuario para programar aplicaciones escalables. Se diseño específicamente para el web, teniendo en cuenta la escalabilidad, la independencia y el estándar XML.

API: Application Programming Interface. Conjunto de convenciones internacionales que definen como debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos API comunes a los que deben ajustarse todos los desarrolladores.

Aplicación: Software que realiza una función particular para el usuario, correo electrónico, consulta de datos en línea.

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 facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad.

Archivo WSDL: Web Services Description Language. Archivo utilizado para descubrir Servicios Web y que permite a las aplicaciones el describirle a otras aplicaciones, las reglas para interactuar entre sí.

Argumento: Parte de una función que identifica los datos sobre los cuales se puede operar.

Autentificación: Capacidad de probar una entidad, por ejemplo un usuario o una computadora es quien dice ser.

Biblioteca Digital: Extensión electrónica de funciones que los usuarios típicamente realizan y los recursos que ellos acceden en una biblioteca tradicional.

Base de Datos: Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior.

Browser: Visor o examinador. Programa cliente y herramientas básicas de navegación para buscar los diferentes recursos de internet. Los más usados son Netscape, Navigator, Microsoft Internet Explorer, y Mosaic de la NCSA.

Clase: En programación orientada a objetos, un tipo de datos definido por el usuario que especifica un conjunto de objetos que comparten las mismas características. Un miembro de la

clase (objeto) es un “ejemplo” o caso de la clase. Las clases concretas están diseñadas para citar como ejemplos las clases abstractas, para pasar las características por herencia.

Cliente/Servidor: modelo lógico de una forma de proceso cooperativo, independiente de plataformas hardware y sistemas operativos. El concepto se refiere más a una filosofía que a un conjunto determinado de productos. Generalmente, el modelo se refiere a un puesto de trabajos o cliente que accede mediante una combinación de hardware y software a los recursos situados en una computadora denominado servidor.

Código de Acceso: Combinación de letras, números y signos que debe introducirse para tener acceso a un programa o partes de un programa determinado, una terminal o computadora personal, un punto en la red, etc.

Código Fuente: Programa en su forma original, tal y como fue escrito por el programador, el código fuente no es ejecutable directamente por el computador, debe convertirse en lenguaje de maquina mediante compiladores, ensambladores o interpretes.

Comando: Instrucción dirigida a una computadora que invoca la ejecución de una secuencia de instrucciones programada previamente.

C++: Es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes como ROOT (enlace externo). Las principales características del C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica (templates).

C#: Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (más notablemente de Delphi y Java).

DNS:

Firewall: Cortafuegos (o firewall en inglés), es un elemento de hardware o software utilizado en una red de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red que haya definido la organización responsable de la red.

Framework: En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

HOST: Término host (equipo anfitrión) en informática o computación puede referirse a:

Aquel ordenador de la red que ofrece servicios a otros ordenadores conectados a dicha red.

A una máquina conectada a una red de ordenadores y que tiene un nombre de equipo (en inglés, hostname). Es un nombre único que se le da a un dispositivo conectado a una red informática. Puede ser un ordenador, un servidor de archivos, un dispositivo de almacenamiento por red, una máquina de fax, impresora, etc. Este nombre ayuda al administrador de la red a identificar las máquinas sin tener que memorizar una dirección IP para cada una de ellas.

Por extensión, a veces también se llama así al dominio del equipo (Un dominio es la parte de una URL por la que se identifica al servidor en el que se aloja)

También es el nombre de un fichero (fichero Hosts) que se encuentra en los ordenadores y resuelve algunos DNS.

IDE: Entorno de desarrollo integrado .Un entorno de desarrollo integrado o en inglés **Integrated Development Environment** ('IDE') es un programa compuesto por un conjunto de herramientas para un programador.

Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic por ejemplo puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic, etc.

Java: Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Lenguaje de programación: Es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Log o log in: Un registro oficial de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática un log es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (who, what, when, where y why, las cinco “w”) un evento ocurre para un dispositivo en particular o aplicación.

Mainframe: Nombre que se da a las grandes computadoras, capaces de atender a miles de usuarios y miles de programas "al mismo tiempo" asignándole un periodo muy pequeño a la atención de cada programa. Su capacidad de trabajo es muy alta, por lo que normalmente se encuentran en empresas de gran tamaño. Sus programas están compuestos por cientos de miles o millones de líneas de código.

Microsoft: Es una empresa multinacional estadounidense, fundada en 1975 por Bill Gates y Paul Allen. Dedicada al sector de la informática, con sede en Redmond, Washington, Estados Unidos. Microsoft desarrolla, fabrica, licencia y produce software para equipos electrónicos. Siendo sus productos más usados el Sistema operativo Microsoft Windows y la suite Microsoft Office.

Internet Explorer es un navegador web producido por Microsoft para el sistema operativo Windows y más tarde para Apple Macintosh y Solaris Unix, estas dos últimas descontinuadas en el 2006 y 2002 respectivamente.

Microsoft Internet Explorer: Es un navegador web producido por Microsoft para el sistema operativo Windows y más tarde para Apple Macintosh y Solaris Unix, estas dos últimas descontinuadas en el 2006 y 2002 respectivamente. Fue creado en 1995 tras la adquisición por parte de Microsoft del código fuente de Mosaic, un navegador desarrollado por Spyglass, siendo rebautizado entonces como Internet Explorer. Actualmente es el navegador de Internet más popular y más utilizado en el mundo, rebasando en gran medida a las competencias existentes, aún cuando algunas de éstas han incrementado su popularidad en los últimos años. Su popularidad es debido a que Internet Explorer es el navegador oficial de Windows, y viene incluido de fábrica en dicho sistema operativo.

Microsoft Windows: Windows es una familia de sistemas operativos desarrollados y comercializados por Microsoft. Existen versiones para hogares, empresas, servidores y dispositivos móviles, como computadores de bolsillo y teléfonos inteligentes. Hay variantes para procesadores de 16, 32 y 64 bits.

Incorpora diversas aplicaciones como Internet Explorer, el Reproductor de Windows Media, Windows Movie Maker, Windows Mail, Windows Messenger, Windows Defender, entre otros.

Desde hace muchos años es el sistema operativo más difundido y usado del mundo, de hecho la mayoría de los programas (tanto comerciales como gratuitos y libres) se desarrolla originalmente para este sistema. Todos los fabricantes del planeta dedicados a equipos basados en procesadores

Intel o compatibles con éstos (excepto Apple Inc.) preinstalan Windows en su versión más reciente y todas sus variantes.

Windows Vista es la versión más reciente para computadoras personales, Windows Server 2008 para servidores y Windows Mobile 6.0 en los dispositivos móviles.

.NET (Dot-net o en español punto-net): Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado. .NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones o como la misma plataforma las denomina, soluciones permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

Página Web: Una página web es una fuente de información adaptada para la World Wide Web (WWW) y accesible mediante un navegador de Internet. Esta información se presenta generalmente en formato HTML y puede contener hiperenlaces a otras páginas web, constituyendo la red enlazada de la World Wide Web.

Las páginas web pueden ser cargadas de un ordenador o computador local o remoto, llamado Servidor Web, el cual servirá de HOST. El servidor web puede restringir las páginas a una red privada, por ejemplo, una intranet, o puede publicar las páginas en el World Wide Web.

Password, contraseña o clave (en inglés password): Es una forma de autenticación que utiliza información secreta para controlar el acceso hacia algún recurso. La contraseña normalmente debe mantenerse en secreto ante aquellos a quien no se les permite el acceso. Aquellos que desean acceder a la información se les solicita una clave; si conocen o no conocen la contraseña, se concede o se niega el acceso a la información según sea el caso.

PHP: PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+. PHP es un acrónimo recursivo que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdof en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Fundation considera esta licencia como software libre.

RPC: Remote Procedure Call (Llamada de Procedimiento Remoto) Es un protocolo que permite a un programa de computadora ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro del paradigma cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente.

Sistema Operativo: Un sistema operativo es un software de sistema, es decir, un conjunto de programas de computadora destinado a permitir una administración eficaz de sus recursos. Comienza a trabajar cuando se enciende el computador, y gestiona el hardware de la máquina desde los niveles más básicos, permitiendo también la interacción con el usuario.

Un sistema operativo se puede encontrar normalmente en la mayoría de los aparatos electrónicos que utilicen microprocesadores para funcionar, ya que gracias a éstos podemos entender la máquina y que ésta cumpla con sus funciones (teléfonos móviles, reproductores de DVD, autoradios, computadoras, etc.).

Sistema heredado (o sistema legacy): Es un sistema informático (ordenador o aplicación) que continúa siendo utilizado por el usuario (típicamente una organización) y no quiere o puede ser reemplazado o actualizado. Habitualmente se utiliza este término para referirse a sistemas anticuados.

Los sistemas heredados son considerados potencialmente problemáticos por numerosos ingenieros de software por diversos motivos. Dichos sistemas a menudo operan en ordenadores obsoletos y lentos, cuyo mantenimiento tiene elevados costes y difíciles de actualizar por falta de componentes adecuados.

TI: Tecnologías de la Información y la Comunicación (TIC), se encargan del estudio, desarrollo, implementación, almacenamiento y distribución de la información mediante la utilización de hardware y software como medio de sistema informático.

Las tecnologías de la información y la comunicación son una parte de las tecnologías emergentes que habitualmente suelen identificarse con las siglas TIC y que hacen referencia a la utilización de medios informáticos para almacenar, procesar y difundir todo tipo de información con diferentes finalidades (formación educativa, organización y gestión empresarial, toma de decisiones en general, etc.).

World Wide Web: WWW ("Web") o Red Global Mundial es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces.

XML: siglas en inglés de Extensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

Objetivos

La realización de este proyecto tiene los siguientes objetivos:

Objetivo general

- Diseñar una aplicación que utilice servicios web para consultar, agregar y modificar datos de profesores de la base de datos del directorio del departamento de horarios de ESIME Zacatenco.

Objetivos específicos

- Diseño y desarrollo de un directorio usando un servicio web que permita a un usuario acceder a la información (base de datos de los horarios de profesores) del directorio del departamento de horarios de ESIME Zacatenco.
- Diseñar y crear una interfaz grafica de usuario de la aplicación web.
- Diseñar y crear la base de datos para la aplicación web.
- Diseñar los servicios web que permitan consultar y modificar la base de datos.

Metas del proyecto

- Proponer un método sencillo por el cual la comunidad de ESIME Zacatenco obtenga información, del directorio de profesores del departamento de horarios por vía internet, y permitir de esta forma una mejor comunicación y desempeño entre alumnos-profesores y administrativos-profesores.
- Proponer un medio por el cual poder consultar información de los profesores como un servicio a la comunidad de ESIME Zacatenco.

CAPITULO 1
INTRODUCCION

1.1 Conceptos generales de protocolos y servicios.

Los Servicios Web ó en inglés Web Services, surgieron ante una necesidad de estandarizar la comunicación entre distintas plataformas (PC, Mainframe, Mac, etc.) y lenguajes de programación (**PHP**, **C#**, **Java**, etc.).

Anteriormente se habían realizado intentos de crear estándares pero fracasaron o no tuvieron el suficiente éxito, algunos de ellos son DCOM y CORBA, por ser dependientes de la implementación del vendedor DCOM - **Microsoft**, y CORBA - ORB (a pesar que CORBA de múltiples vendedores pueden operar entre sí, hay ciertas limitaciones para aplicaciones de niveles más altos en los cuales se necesite seguridad o administración de transacciones).

Otro gran problema es que se hacía uso de **RPC** (Remote Procedure Call) para realizar la comunicación entre diferentes nodos. Esto, además de presentar ciertos problemas de seguridad, tiene la desventaja de que su implementación en un ambiente como es Internet, es casi imposible (muchos **firewalls** bloquean este tipo de mensajes, lo que hace prácticamente imposible a dos computadoras conectadas por Internet comunicarse).

Los **Web Services** surgieron para finalmente poder lograr la tan esperada comunicación entre diferentes plataformas. En la actualidad muchos ***sistemas legacy** están pasando a ser web Services.

Es por esto que en 1999 se comenzó a plantear un nuevo estándar, el cual terminaría utilizando XML, SOAP, WSDL, y UDDI. **[1]**

Los servicios web trabajan bajo los conceptos generales de recopilación, organización y almacenamiento de información de manera digital, con el propósito de que los usuarios tengan opciones de búsqueda, recuperación y procesamiento de información, utilizando las actuales tecnologías como son: las redes de computadoras e Internet.

El presente proyecto tiene como propósito el diseño y creación de un directorio de profesores de la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional en la Unidad Profesional Adolfo López Mateos (Zacatenco, México D.F.), basado en la actual tecnología de Servicios Web y representa una alternativa para el

almacenamiento, procesamiento y recuperación de información y del manejo de diferentes fuentes de información.

¿De qué tipo de información se habla?, en el tiempo en que se desarrolló este proyecto y previamente a este, se ha presentado una situación particular, situación en la cual alumnos, profesores, personal administrativo y personas externas a la comunidad de ESIME, han experimentado en algún momento la falta de información que no les ha permitido desempeñar sus actividades y roles de manera adecuada.

¿Cómo se podría lograr esto?, con ayuda de un tipo de tecnología de la información (**TI**), en nuestro caso una aplicación web. Una aplicación web es un conjunto de elementos de software que integran un sistema, el cual es posible acceder desde la red propia de una institución o utilizando internet (tomar en cuenta que es necesario ocupar una computadora con acceso a la red interna del IPN ó con acceso a internet) , los usuarios en este caso: alumnos, profesores y personal administrativo pueden acceder a este servicio y consultar, manipular y modificar información de manera que permita la fácil y correcta utilización de los recursos, así como hacer más eficientes las actividades propias de los miembros de la comunidad de ESIME Zacatenco.

La aplicación web le permitiría a los usuarios consultar o manipular información de su interés mediante una página web, accediendo a ella mediante un dominio (un nombre fácil de recordar y de buscar en internet), dicha página permitiría a cada usuario tener acceso a información tal como: horario de profesor, nombre, área de cubículo que le corresponde, materias que imparte, grupos que imparte, etc. Esta página permitiría a los alumnos y profesores ver rápidamente y en forma clara los datos anteriores; al personal administrativo le permitirá consultar y modificar de manera fácil y rápida la información que necesite ser consultada o modificada.

¿Qué características tendría la aplicación web? Al utilizar web Services (es un grupo de reglas que permiten intercambiar datos entre aplicaciones), la aplicación podría ser accesible desde diferentes plataformas (Sistemas Operativos en las computadoras de los usuarios), podría ser publicada y ser visible (puesta en la red) sin tener que lidiar y modificar las reglas de seguridad de la red, ya que sería invisible a estas. Los servicios web son muy prácticos porque pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada; es decir pueden crearse varios servicios web que acceden a la información y ser utilizados por diferentes aplicaciones y para diversos motivos según lo requieran los usuarios.

1.2 Antecedentes

Debido a la gran masificación de Internet a niveles insospechables, el gran impacto causado por las tecnologías de la información en las últimas dos décadas del siglo pasado, la manera de hacer negocios, la comunicación entre las personas y las empresas, el acelerado crecimiento en los volúmenes de información, la limitación de recursos humanos para su procesamiento, el surgimiento de nuevas tecnologías informáticas tales como Internet, las bases de datos y las aplicaciones para desarrollo Web, se hicieron cada vez mayor las necesidades de integrar y compartir información entre distintas plataformas de software y hardware.

Las empresas se percataron que era imposible crear una plataforma integrado de forma individual, así que decidieron atacar el problema de raíz. Para esto decidieron que en vez de crear la mejor plataforma integradora, era mejor buscar un lenguaje común de intercambio de información aprovechando los estándares existentes en el mercado.

Bajo este contexto nacen los Servicios Web basados en XML, que son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

La tendencia hacia el desarrollo basadas en la Web, han hecho que la mayoría de las bases de datos, busquen publicar su información a través de este medio, para así consultarla y poder trabajar con ella.

Un usuario (que juega el papel de cliente dentro de los Servicios Web), a través de una aplicación, solicita información haciendo una petición a una agencia que ofrece sus servicios a través de Internet. La agencia ofrecerá a su cliente (usuario) la información requerida. Para proporcionar al cliente la información que necesita, esta agencia solicita a su vez información a otros recursos (otros Servicios Web). La agencia obtendrá información de estos recursos, lo que la convierte a su vez en cliente de esos otros Servicios Web que le van a proporcionar la información solicitada. Por último, el usuario

realizará el pago a través de la agencia que servirá de intermediario entre el usuario y el servicio Web que gestionará el pago.

En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP (Protocolo Simple de Acceso a Objetos). Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc. SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto por un envelope (sobre), cuya estructura está formada por los siguientes elementos: header (cabecera) y body (cuerpo).

Para optimizar el rendimiento de las aplicaciones basadas en Servicios Web, se han desarrollado tecnologías complementarias a SOAP, que agilizan el envío de los mensajes (MTOM) y los recursos que se transmiten en esos mensajes (SOAP-RRSHB).

Por otro lado, WSDL (Lenguaje de Descripción de Servicios Web), permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes.

Durante la evolución de las necesidades de las aplicaciones basadas en Servicios Web de las grandes organizaciones, se han desarrollado mecanismos que permiten enriquecer las descripciones de las operaciones que realizan sus servicios mediante anotaciones semánticas y con directivas que definen el comportamiento. Esto permitiría encontrar los Servicios Web que mejor se adapten a los objetivos deseados. Además, ante la complejidad de los procesos de las grandes aplicaciones empresariales, existe una tecnología que permite una definición de estos procesos mediante la composición de varios Servicios Web individuales, lo que se conoce como coreografía.

1.3 Planteamiento del problema

El departamento de horarios de ESIME Zacatenco, es el departamento encargado de la asignación de la carga de horas de los profesores (distribución de horario de clases y grupos), su principal función es la de publicar y organizar los horarios de profesores por academias y carreras así como la distribución de los grupos, este departamento administra la información de la situación de los profesores, promociones entre otras cosas. La información administrada por el departamento de horarios es de carácter administrativo, es decir involucra a personal administrativo y personal docente

(profesores), los horarios son distribuidos con un semestre de anticipación, pero son modificados incluso cuando ya han sido publicados, estas modificaciones se deben, a que en ocasiones los profesores y academias necesitan ajustes en los horarios, ya sea para cubrir completamente el horario de las asignaturas y distribuir correctamente al personal docente, o ya sea por que el personal docente necesite adecuar su horario de trabajo, otra razón es cuando un grupo nuevo en periodo extraordinario es abierto y necesita que personal docente cubra las asignaturas, entonces es cuando se necesita hacer un reajuste y reacomodar los horarios, es en estas situaciones cuando se necesita un medio que permita hacer de forma fácil y rápida la modificación y la publicación. Para poder llevar a cabo una distribución de horarios es necesario que academia, profesores y departamento de horarios coincidan y aprueben los ajustes.

Desde el punto de vista de los alumnos, estos últimos pueden consultar su horario de clases de acuerdo a su grupo, esto puede realizar por medio de la página de control escolar, en donde los horarios son publicados por grupo. Por este medio no es posible y actualizar los horarios de acuerdo a los cambios del departamento de control escolar, y por tanto el alumno no tiene acceso en el momento en el que el horario de sus profesores es modificado.

1.4 Justificación

El presente proyecto pretende cubrir la necesidad de la falta de un medio que permita a los alumnos de ESIME Zacatenco tener información de sus profesores para poder localizarlos y poder tratar información relacionada con las actividades académicas. En nuestra experiencia académica como alumnos de la ESIME, experimentamos la necesidad de tener un medio ó procedimiento para saber el horario de atención de los profesores de las de las diferentes asignaturas, en muchas ocasiones con la necesidad de reunirse con el motivo de apoyo extra clase, entrega de tareas, etc. Dichas reuniones pretendían hacerse en el cubículo del profesor, un área de residencia en el horario de trabajo del profesor donde se le puede localizar, pero en muchas ocasiones los alumnos nos encontramos con varios problemas: el primero, en ocasiones al solicitar información acerca del cubículo de un determinado profesor los alumnos no sabían el nombre completo del profesor de la asignatura, esta falta de información limitaba la localización de un profesor e incluso cuando se le preguntaba a una persona que posiblemente conociera el cubículo del profesor buscado, esta era incapaz de proporcionar una información concreta que ayudara al alumno a localizar al profesor y su cubículo. El segundo problema radicaba en que en otras ocasiones no solo no se sabía el nombre completo del profesor sino que no

se sabía en qué área de cubículos se encontraba el profesor, en este caso solo se podía localizar el cubículo extendiendo la búsqueda con personas que conocieran al profesor o que supieran donde localizarlo.

Cabe mencionar que aunque es un protocolo común entre profesores informar cual es su área de cubículos y su nombre completo al momento de presentarse por primera vez ante un grupo, no siempre esta información es válida durante todo el curso, ya que por motivos administrativos (ajustes en el horario de las asignaturas y profesores que la imparten) el horario de trabajo y por lo tanto el tiempo de estancia en los cubículos de los profesores variaría a lo largo del semestre.

El propósito general de este trabajo es diseñar y desarrollar un Servicios Web, que sirva de modelo y proporcione funcionalidad a un directorio de profesores, al mismo tiempo que se documenten los pasos necesarios para llegar a este fin, esto dirigido a gente que esté interesada en el desarrollo de un servicio web, esperando sea de completa comprensión.

Una de las razones importantes para optar por esta tecnología es que los Servicios Web pueden aportar gran independencia entre la aplicación que usa el Servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante dado que la tendencia de una administración escolar suele ser masiva y en ocasiones cambiante.

Los servicios web están considerados como una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, estos a su vez pueden ser empleados por distintas aplicaciones de software para intercambiar información a través de la red.

Una de las ventajas más importantes del uso de esta tecnología es la aportación de interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar abiertos como XML, SOAP, WDSL y UDDI.

1.5 Análisis y Alcances

En el presente trabajo de tesis, se describe el diseño y desarrollo de un Servicio Web, que dan funcionalidad a una base de datos, con los siguientes análisis y alcances:

- Diseñar los Servicios Web para apoyar la operación de la base de datos para los profesores de ESIME Zacatenco.
- Diseñar una aplicación web para la comunidad de ESIME Zacatenco, que le permita a esta comunidad el fácil acceso y utilización de la información y los recursos.

CAPITULO 2
METODOLOGIAS DE DESARROLLO

Marco Teórico

2.1 Ciclo de Vida Del Software SDLC (Systems Development Life Cycle)

El desarrollo de software va unido a un ciclo de vida compuesto por una serie de etapas que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo producto software, hasta aquel en que el producto deja definitivamente de ser utilizado por el último de sus usuarios.

2.1.1 Etapas en el ciclo

Veamos, a grandes rasgos, una pequeña descripción de etapas con que podemos contar a lo largo del ciclo de vida del software; una vez delimitadas en cierta manera las etapas, habrá que ver la forma en que estas se afrontan (existen diversos modelos de ciclo de vida, y la elección de un cierto modelo para un determinado tipo de proyecto puede ser de vital importancia; el orden de las etapas es un factor importante, por ejemplo, tener una etapa de validación al final del proyecto, tal como sugiere el modelo en cascada o lineal, puede implicar serios problemas sobre la gestión de determinados proyectos; hay que tener en cuenta que retomar etapas previas es costoso, y cuanto más tarde se haga más costoso resultará, por tanto el hecho de contar con una etapa de validación tardía tiene su riesgo y, por su situación en el ciclo, un posible tiempo de reacción mínimo en caso de tener que retornar a fases previas).

Expresión de necesidades

Esta etapa tiene como objetivo la consecución de un primer documento en que queden reflejados los requerimientos y funcionalidades que ofrecerá al usuario del sistema a desarrollar (qué, y no cómo, se va a desarrollar).

Dado que normalmente se trata de necesidades del cliente para el que se creará la aplicación, el documento resultante suele tener como origen una serie de entrevistas cliente-proveedor situadas en el contexto de una relación comercial, siendo que debe ser comprendido por ambas partes (puede incluso tomarse como base para el propio acuerdo comercial).

Especificaciones

Ahora se trata de formalizar los requerimientos; el documento obtenido en la etapa anterior se tomará como punto de partida para esta fase. Su contenido es aún insuficiente y lleno de imprecisiones que será necesario completar y depurar.

Por medio de esta etapa se obtendrá un nuevo documento que definirá con más precisión el sistema requerido por el cliente (el empleo de los casos de uso, use cases, de Jacobson es una muy buena elección para llevar a cabo la especificación del sistema).

Lo más normal será que no resulte posible obtener una buena especificación del sistema a la primera; serán necesarias sucesivas versiones del documento en que irán quedando reflejada la evolución de las necesidades del cliente (por una parte no siempre sabe en los primeros contactos todo lo que quiere realmente, y por otra parte pueden surgir cambios externos que supongan requerimientos nuevos o modificaciones de los ya contemplados).

Análisis

Es necesario determinar qué elementos intervienen en el sistema a desarrollar, así como su estructura, relaciones, evolución en el tiempo, detalle de sus funcionalidades, ... que van a dar una descripción clara de qué sistema vamos a construir, qué funcionalidades va a aportar y qué comportamiento va a tener. Para ello se enfocará el sistema desde tres puntos de vista relacionados pero diferentes:

- ✓ Funcional.
- ✓ Estático.
- ✓ Dinámico.

Diseño

Tras la etapa anterior ya se tiene claro que debe hacer el sistema, ahora tenemos que determinar cómo va a hacerlo (¿cómo debe ser construido el sistema?; aquí se definirán en detalle entidades y relaciones de las bases de datos, se pasará de casos de uso

esenciales a su definición como casos expandidos reales, se seleccionará el lenguaje más adecuado, el Sistema Gestor de Bases de Datos a utilizar en su caso, librerías, configuraciones hardware, redes, etc.).

Observación:

Aunque todo debe ser tratado a su tiempo, y sería muy deseable que las decisiones correspondientes en esta etapa fueran tomadas precisamente en esta etapa, muchas veces nos vamos a encontrar con unas decisiones previamente impuestas sobre lenguaje, plataforma, etc. Unas veces se dirán justificadas en simple política de empresa y por mantener "compatibilidad" en lo que respecta a los demás proyectos de la propia empresa, y en otras ocasiones por rumores de que tal o cual herramienta mejoraría la velocidad de desarrollo u otro aspecto de interés (en parte de los casos no serán rumores con fundamento o estudios previos realizados al efecto, sino más bien debidos a la propia publicidad como consejera).

Implementación

Llegado este punto se empieza a codificar algoritmos y estructuras de datos, definidos en las etapas anteriores, en el correspondiente lenguaje de programación y/o para un determinado sistema gestor de bases de datos.

Análisis:

Lamentablemente en la actualidad, año 2000, quedan bastantes empresas en las que, tras una reunión comercial en que tan solo se ha conseguido recabar una breve lista de requerimientos, a pesar de tener que enfrentarse a proyectos grandes-medios, se pasa directamente a la etapa de implementación; son proyectos guiados por el riesgo que supone adoptar un modelo de ciclo de vida de codificar-corregir (code and fix) donde se eliminan las fases de especificaciones, análisis y diseño con la consiguiente pérdida de control sobre la gestión del proyecto.

Pruebas

El objetivo de estas pruebas es garantizar que el sistema ha sido desarrollado correctamente, sin errores de diseño y/o programación. Es conveniente que sean planteadas al menos tanto a nivel de cada módulo (aislado del resto), como de integración

del sistema (según sea la naturaleza del proyecto en cuestión se podrán tener en cuenta pruebas adicionales, por ejemplo de rendimiento).

Validación

Esta etapa tiene como objetivo la verificación de que el sistema desarrollado cumple con los requisitos expresados inicialmente por el cliente y que han dado lugar al presente proyecto (para esta fase también es interesante contar con los use cases, generados a través de las correspondientes fases previas, que servirán de guía para la verificación de que el sistema cumple con lo descrito por estos).

Mantenimiento y evolución

Finalmente la aplicación resultante se encuentra ya en fase de producción (en funcionamiento para el cliente, cumpliendo ya los objetivos para los que ha sido creada). A partir de este momento se entra en la etapa de mantenimiento, que supondrá ya pequeñas operaciones tanto de corrección como de mejora de la aplicación (p.ej. mejora del rendimiento), así como otras de mayor importancia, fruto de la propia evolución (p.ej. nuevas opciones para el usuario debidas a nuevas operaciones contempladas para el producto).

La mayoría de las veces en que se desarrolla una nueva aplicación, se piensa solamente en un ciclo de vida para su creación, olvidando la posibilidad de que esta deba sufrir modificaciones futuras (que tendrán que producirse con casi completa seguridad para la mayor parte de los casos).

2.2 Programación Extrema

Los test antes que el programa

La XP llevada al extremo implica que se escriban los test (debug tests) antes que la propia aplicación. Esto tampoco es estrictamente necesario, pero sin embargo hay "Tests Units Frameworks" que son una serie de tests pre-fabricados para aplicar directamente a tu aplicación. ¿Por qué? ¿Por qué son tan importantes? Cuando se programa de la forma old-school, uno deja para el final el testear su aplicación y encontrar errores. Esto significa que se le da una menor importante porque "en teoría" el programa no debería de fallar, aunque la realidad sea otra. Además suele ocurrir que al encontrar un fallo se aplica un parche que sólo ensucia el código y que ni siquiera se integra bien con el código.

Testeando nuestra aplicación desde el primer momento, nos aseguramos de que el código escrito hasta entonces es correcto y no estropea lo anterior. La programación orientada a objetos facilita esta tarea, ya que los objetos tienen una "interfaz" y deben responder ante ciertos "estímulos". Así por ejemplo, en un caso práctico, podríamos escribir los métodos vacíos de una clase, dentro de cada método escribir solamente el código necesario para que imprima por pantalla un mensaje que diga que se llegó a tal método o a tal otro. Y después, poco a poco, ir escribiendo el resto del código.

2.2.1 El proceso de desarrollo extremo

La programación extrema parte del caso habitual de una compañía que desarrolla software, generalmente software a medida, en la que hay diferentes roles: un equipo de gestión, un equipo de desarrolladores y los clientes. La relación con el cliente es totalmente diferente a lo que se ha venido haciendo en las metodologías tradicionales que se basan fundamentalmente en una fase de captura de requisitos previa al desarrollo y una fase de validación posterior al mismo.

Interacción con el cliente

En la programación extrema al cliente no sólo se le pide que apoye al equipo de desarrollo, en realidad podríamos decir que es parte de él. Su importancia es capital a la hora de abordar las historias de los usuarios y las reuniones de planificación. Además, será tarea suya realimentar al equipo de desarrolladores después de cada iteración con los

problemas con los que se ha encontrado, mostrando sus prioridades, expresando sus sensaciones... Existirán métodos como pruebas de aceptación que ayudarán a que la labor del cliente sea lo más fructífera posible.

En resumen, el cliente se encuentra mucho más cercano al proceso de desarrollo. Se elimina la fase inicial de captura de requisitos y se permite que éstos se vayan definiendo de una forma ordenada durante el tiempo que dura el proyecto. El cliente puede cambiar de opinión sobre la marcha y a cambio debe encontrarse siempre disponible para resolver dudas del equipo de desarrollo y para detallar los requisitos especificados cuando sea necesario.

El proceso de captura de requisitos de XP gira en torno a una lista de características que el cliente desea que existan en el sistema final. Cada una de estas características recibe el nombre de historias de usuarios y su definición consta de dos fases:

En la primera fase el cliente describe con sus propias palabras las características y el responsable del equipo de desarrollo le informa de la dificultad técnica de cada una de ellas y por lo tanto de su coste. A través del diálogo resultante el cliente deja por escrito un conjunto de historias y las ordena en función de la prioridad que tienen para él. En este momento ya es posible definir unos hitos y unas fechas aproximadas para ellos.

La segunda fase consiste en coger las primeras historias que serán implementadas (primera iteración) y dividirlas en las tareas necesarias para llevarlas a cabo. El cliente también participa, pero hay más peso del equipo de desarrollo, que dará como resultado una planificación más exacta. En cada iteración se repetirá esta segunda fase para las historias planificadas para ella.

2.2.2 Planificación del proyecto

Es probablemente en este punto donde nos debamos enfrentar a la planificación de entregas (release planning) donde planificaremos las distintas iteraciones. Para ello existen una serie de reglas que hay que seguir para que las tres partes implicadas en este proceso (equipo de gestión, equipo de desarrollo y cliente) tengan voz y se sientan parte de la decisión tomada, que al fin y al cabo debe contentar a todos.

La planificación debe de seguir unas ciertas premisas. La primordial es que las entregas se hagan cuanto antes y que con cada iteración el cliente reciba una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo habrá para trabajar

en ella posteriormente. Se aconsejan muchas entregas y muy frecuentes. De esta forma, un error en una parte esencial del sistema se encontrará pronto y, por tanto, se podrá arreglar antes.

Sin embargo, los requisitos anteriores en cuanto a la planificación no deben suponer horas extra para el equipo de desarrollo. El argumento que se esboza es que lo que se trabaja de más un día, se deja de trabajar al siguiente. Diversas prácticas como las pruebas unitarias, la integración continua o el juego de la planificación permiten eliminar los principales motivos por los que suele ser necesario trabajar muchas horas extra.

Pero lo mejor de todo es que a la hora de planificar uno se puede equivocar. Es más, todos sabemos que lo común es equivocarse y por ello la metodología ya tiene previsto mecanismos de revisión. Por tanto, es normal que cada 3 a 5 iteraciones se tengan que revisar las historias de los usuarios y renegociar nuevamente la planificación.

La planificación en iteraciones y el diseño iterativo dan pie a una práctica poco común en el desarrollo tradicional que son las discusiones diarias de pie. De esta forma, se fomenta la comunicación, ya que los desarrolladores cuentan con tiempo para hablar de los problemas a los que se enfrentan y cómo van con su(s) tarea(s), a la vez que su carácter informal las hace agradables y, sobre todo, no se alargan.

2.2.3 Diseño, Desarrollo y Pruebas

El desarrollo es la pieza clave de todo el proceso de programación extrema. Todas las tareas tienen como objetivo que se desarrollo a la máxima velocidad, sin interrupciones y siempre en la dirección correcta.

También se otorga una gran importancia al diseño y establece que éste debe ser revisado y mejorado de forma continua según se van añadiendo funcionalidades al sistema. Esto se contrapone a la práctica conocida como "Gran diseño previo" habitual en otras metodologías. Los autores de XP opinan que este enfoque es incorrecto dado que a priori no se tiene toda la información suficiente para diseñar todo el sistema y se limita la posibilidad del cliente de cambiar de opinión respecto a las funcionalidades deseadas. Como veremos a continuación a cambio se establecen los mecanismos para ir remodelando el diseño de forma flexible durante todo el desarrollo.

La clave del proceso de desarrollo de XP es la comunicación. La gran mayoría de los problemas en los proyectos de desarrollo son provocados por falta de comunicación en el

equipo, así que se pone un gran énfasis en facilitar que la información fluya lo más eficientemente posible.

Es en este punto donde entra uno de los términos estrella de la programación extrema: la metáfora. El principal objetivo de la metáfora es mejorar la comunicación entre los todos integrantes del equipo al crear una visión global y común del sistema que se pretende desarrollar. La metáfora debe estar expresada en términos conocidos para los integrantes del grupo, por ejemplo comparando lo que se va a desarrollar con algo que se puede encontrar en la vida real. Aunque también se incluye información sobre las principales clases y patrones que se usarán en el sistema.

Aunque en general el diseño es realizado por los propios desarrolladores en ocasiones se reúnen aquellos con más experiencia o incluso se involucra al cliente para diseñar las partes más complejas. En estas reuniones se emplean un tipo de tarjetas denominadas CRC (Class, Responsabilities and Collaboration - Clases, Responsabilidades y Colaboración) cuyo objetivo es facilitar la comunicación y documentar los resultados. Para cada clase identificada se rellenará una tarjeta de este tipo y se especificará su finalidad así como otras clases con las que interaccione. Las tarjetas CRC son una buena forma de cambiar de la programación estructurada a una filosofía orientada a objetos. Aunque los grandes gurús de la programación extrema sostienen que bien hechas suelen hacer el diseño obvio, recomiendan hacer sesiones CRC en caso de que el sistema que se pretenda crear tenga un grado de complejidad grande. Este tipo de sesiones es una simulación, tarjetas CRC en mano, de las interacciones entre los diferentes objetos que puede realizar el equipo de desarrollo.

2.2.4 Principio de la programación extrema

Los principios fundamentales se apoyan en los valores y también son cuatro. Se busca:

- Realimentación veloz
- Modificaciones incrementales
- Trabajo de calidad
- Elevación de simplicidad.

Los principios suponen un puente entre los valores (algo intrínseco al equipo de desarrollo) y las prácticas, que se verán a continuación, y que están más ligadas a las técnicas que se han de seguir.

2.2.5 Prácticas de la programación extrema

Por su parte, las prácticas son las siguientes:

- El juego de la planificación (the planning game)
- Pequeñas entregas (small releases)
- Metáfora (metaphor)
- Diseño simple (simple design)
- Pruebas (testing)
- Refactorización (refactoring)
- Programación por parejas (pair programming)
- Propiedad colectiva (collective ownership)
- Integración continua (continous integration)
- 40 horas semanales (40-hour week)
- Cliente en casa (on-site costumer)
- Estándares de codificación (coding standards)

Como ya hemos visto con anterioridad, uno de los principios de la programación extrema es la simplicidad. El diseño debe ser lo más simple posible, pero no más simple. El paradigma KISS ("Keep It Small and Simple" para unos o "Keep it Simple, Stupid" para otros) se lleva hasta las últimas consecuencias. Por ejemplo, se hace énfasis en no añadir funcionalidad nunca antes de lo necesario, por las sencillas razones de que probablemente ahora mismo no sea lo más prioritario o porque quizás nunca llegue a ser necesaria.

Supongamos que ya hemos planificado y dividido en tareas, como se ha comentado en los párrafos anteriores. Lo lógico sería empezar ya a codificar. Pues no. Nos encontramos con otro de los puntos clave de la programación extrema (y que sí es innovador en ella): las pruebas unitarias se implementan a la vez hay que el código de producción. De hecho cada vez que se va a implementar una pequeña parte se escribe una prueba sencilla y luego el código suficiente para que la pase. Cuando la haya pasado se repite el proceso con la siguiente parte. Aunque intuitivamente esto parezca contraproducente, a la larga hará que la generación de código se acelere. Los creadores de la programación extrema argumentan que encontrar un error puede llegar a ser cien veces más caro que realizar las pruebas unitarias. La idea, en definitiva, se resumen en la siguiente frase: "Todo código que pueda fallar debe tener una prueba". Además, hay que tener en cuenta que se hacen una vez y luego se pueden reutilizar multitud de veces, incluso por otros desarrolladores que desconocen los entresijos de esa parte o de todo el sistema, por lo que permiten compartir código (otra de las prácticas que permiten acelerar el desarrollo tal y como se verá más adelante).

Esta forma de usar las pruebas unitarias ayuda a priorizar y comprobar la evolución del desarrollo y que ofrecen realimentación inmediata. Ya no hay imprescindibles dos equipos diferenciados que desarrollan y prueban cada uno por su cuenta. Ahora el ciclo se basa en implementar una prueba unitaria, codificar la solución y pasar la prueba, con lo que se consigue un código simple y funcional de manera bastante rápida. Por eso es importante que las pruebas se pasen siempre al 100% [Jeffries].

Hay mucha literatura sobre las pruebas unitarias [Beck2][Gamma][Gamma2]. La mayoría de los autores están de acuerdo en que cuanto más difícil sea implementar una prueba, más necesarias son. Algunos incluso dicen que entonces quizás sea porque lo que se intenta probar no es lo suficientemente sencillo y ha de rediseñarse. En cuanto a herramientas para realizar tests unitarios, existen varias para los diferentes lenguajes, lo que hace que su ejecución sea simple y, sobre todo, automáticas [Impl].

Las pruebas unitarias no se han de confundir con las pruebas de aceptación que han sido mencionadas con anterioridad. Éstas últimas son pruebas realizadas por el cliente o por el usuario final para constatar que el sistema hace realmente lo que él quiere. En caso de que existan fallos, debe especificar la prioridad en que deben ser solucionados los diferentes problemas encontrados. Este tipo de pruebas son pruebas de caja negra y se hacen contra las historias de los usuarios. Se suele tender a que sean parcialmente automáticos y que los resultados sean públicos.

Es hora entonces de ampliar el ciclo de creación de pruebas unitarias, codificación, paso de las pruebas y añadirle un paso más: la integración. La programación extrema viene a perseguir lo que se ha venido a llamar integración continua. De esta forma, haciéndolo cada vez con pequeños fragmentos de código, se evita la gran integración final. Las ventajas de este enfoque es que permite la realización de pruebas completas y la pronta detección de problemas de incompatibilidad. Además, ya no será necesario un equipo independiente de integración que haga uso del mágico pegamento al enfrentarse a problemas de divergencias y fragmentación de código.

En todo desarrollo de programación extrema debería existir, por tanto, una versión siempre integrada (incluso se puede asegurar su existencia mediante cerrojos - locks). La sincronización por parte de los desarrolladores con el repositorio central debe darse como mínimo una vez al día, de manera que los cambios siempre se realicen sobre la última versión. De esta forma nos podemos asegurar de que las modificaciones que hacemos no se estén haciendo sobre una versión obsoleta.

La última afirmación, de la probabilidad de encontrarse una versión de código obsoleta (más antigua que el código actual) es muy baja. En cierto modo, esto es cierto en las metodologías tradicionales, pero en la programación extrema no: nos topamos con la importancia de refactorizar [Fowler3]. Refactorizar consiste básicamente en quitar redundancia, eliminar funcionalidad que no se usa o "rejuvenecer" diseños viejos. Tiene su justificación principal en que el código no sólo tiene porque funcionar, también debe ser simple. Esto hace que a la larga refactorizar ahorre mucho tiempo y suponga un incremento de calidad. Por cierto, tal es el énfasis que se pone en la refactorización que de la misma no se libran ni las pruebas unitarias.

Como uno de los objetivos de la programación extrema es que cualquier miembro del equipo de desarrollo puede mejorar cualquier parte del sistema, llegamos fácilmente a la conclusión de que se busca que el código sea de todos. Cualquier desarrollador puede realizar cambios, corregir erratas o refactorizar en cualquier momento. Para eso, entre otras cosas, tenemos el colchón de las pruebas unitarias por si nos equivocamos. Además, es una forma coherente de plasmar que todo el equipo es responsable del sistema en su conjunto y de que no haya feudos personales. En consecuencia, un desarrollador que deje el proyecto (algo habitual, por otra parte) no tiene por qué convertirse en un hecho catastrófico. El mejor método para conseguir que el código sea de todos es seguir unos estándares de codificación consistentes, de manera que la lectura (y refactorización) por parte del resto del equipo de desarrollo se facilite al máximo.

Para terminar esta, ya extensa, descripción de un proceso de desarrollo de programación extrema, he dejado una de sus joyas para el final. El proceso de desarrollo no lo va a hacer un desarrollador en solitario, sino siempre con otra persona, algo que se ha venido a llamar programación por parejas. Una pareja de desarrolladores debe compartir ordenador, teclado y ratón. El principal objetivo es realizar de forma continua y sin parar el desarrollo una revisión de diseño y de código. Las parejas deben ir rotando de forma periódica para hacer que el conocimiento del sistema se vaya difundiendo por el equipo (facilitándose que el código sea de todos), a la vez que se fomentan el entrenamiento cruzado [Jeffries2]. Existen estudios que concluyen que esta práctica es eficaz en la práctica justificándola con aspectos psicológicos y sociológicos [Cockburn]. Con este apoyo los gurús de la programación extrema no dudan en afirmar que dos personas trabajando conjuntamente en pareja generan en cantidad el mismo código (o mejor dicho, la misma funcionalidad) que dos personas por separado, pero de mayor calidad. Sin embargo esta es la práctica que más reticencias provoca por parte de jefes y de los propios programadores.

El proceso que recomiendan los autores de XP es el siguiente: identifica el principal problema del proceso de desarrollo actual. Escoge la práctica que ayuda a resolver ese problema y aplícala. Cuando ese haya dejado de ser un problema, escoge el siguiente. En realidad se recomienda que se apliquen las prácticas de dos en dos. El objetivo es que las prácticas de XP se apoyan unas a otras y por tanto dos prácticas aportan más que la suma de ambas y por tanto es más fácil comprobar los resultados.

El objetivo final debe ser aplicar todas las prácticas, ya que representan un conjunto completo, "si no las aplicas todas no estás haciendo eXtreme Programming".

2.3 Desarrollo en Cascada

En Ingeniería de software el desarrollo en cascada, también llamado modelo en cascada, es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma tal que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

Un ejemplo de una metodología de desarrollo en cascada es:

- **Análisis de requisitos**
- **Diseño del Sistema**
- **Diseño del Programa**
- **Codificación**
- **Pruebas**
- **Implantación**
- **Mantenimiento**

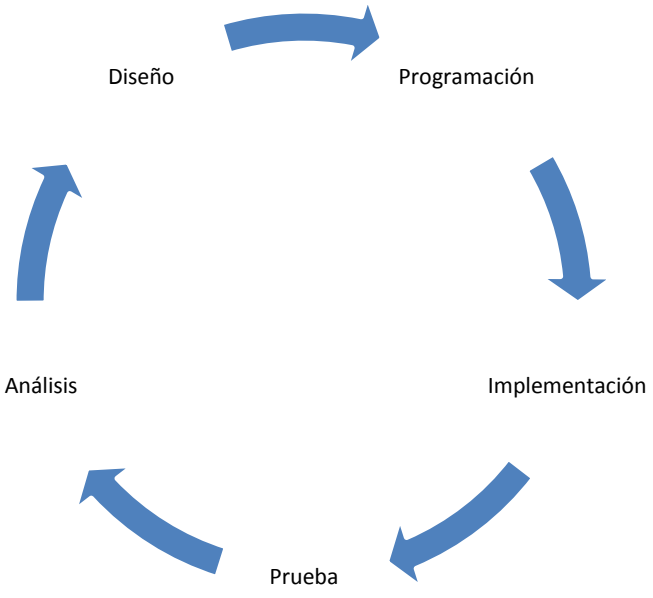


Figura 2.3.1 ejemplo de una metodología de desarrollo en cascada

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costes del desarrollo. La palabra cascada sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el paradigma más seguido al día de hoy.

2.3.1 Fases del modelo.

Análisis de requisitos

Se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se deben consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

Diseño del Sistema

Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Diseño del Programa

Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.

Codificación

Es la fase de programación o implementación propiamente dicha. Aquí se implementa el código fuente, haciendo uso de prototipos así como pruebas y ensayos para corregir errores.

Dependiendo del lenguaje de programación y su versión se crean las librerías y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

Pruebas

Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente antes de ser puesto en explotación.

Implantación

El software obtenido se pone en producción. Se implantan los niveles software y hardware que componen el proyecto. La implantación es la fase con más duración y con más cambios en el ciclo de elaboración de un proyecto. Es una de las fases finales del proyecto

Durante la explotación del sistema software pueden surgir cambios, bien para corregir errores o bien para introducir mejoras. Todo ello se recoge en los Documentos de Cambios.

Variantes

Existen variantes de este modelo; especialmente destacamos la que hace uso de prototipos y en la que se establece un ciclo antes de llegar a la fase de mantenimiento, verificando que el sistema final esté libre de fallos.

2.3.2 Desventajas

En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.

Difícilmente un cliente va a establecer al principio todos los requerimientos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases.

Los resultados y/o mejoras no son visibles, el producto se ve recién cuando este esté finalizado, lo cual provoca una gran inseguridad por parte del cliente que anda ansioso de ver avances en el producto. Esto también implica toparse con requerimientos que no se habían tomado en cuenta, y que surgieron al momento de la implementación, lo cual provocara que se regrese nuevamente a la fase de requerimientos.

2.3.3 Ventajas

Se tiene todo bien organizado y no se mezclan las fases. Es perfecto para proyectos que son rígidos, y además donde se especifiquen muy bien los requerimientos y se conozca muy bien la herramienta a utilizar.

CAPITULO 3
SERVICIOS WEB

3.1 Conceptos generales

¿Qué son los servicios Web?

Esta es una pregunta fundamental: Que problema solucionan los servicios Web, Cual es la necesidad del mercado para esta tecnología, la empresa Microsoft está alentando a los distribuidores de software y comercios electrónicos a que desplieguen servicios web. Como cada vez hay más organizaciones a nivel mundial que se conectan a internet, el concepto de aplicaciones que llaman métodos a través de la red se ha vuelto más práctico. Los servicios web representan el siguiente paso en la programación orientada a objetos, en vez de desarrollar software a partir de un pequeño número de bibliotecas de clases que se proporcionan en una ubicación, los programadores pueden acceder a las bibliotecas de clases de un servicio web que estén distribuidas en todo el mundo.

Un servicio web es una clase que permite que sus métodos sean llamados por otros métodos en otros equipos a través de formatos de datos y protocolos comunes, como XML y HTTP.

En la plataforma **.Net** un servicio web es un componente de software que se almacena en un equipo, al que una aplicación (o cualquier otro componente de software) puede acceder desde otro equipo a través de una red. El equipo en el que reside el servicio web se conoce como equipo remoto. La aplicación (el cliente) que accede al servicio web envía la llamada a un método a través de una red hasta el equipo remoto, el cual procesa la llamada y devuelve una respuesta a la aplicación, a través de la red.

La implementan comúnmente a través del protocolo simple de acceso a objetos (SOAP), un protocolo basado en XML que describe cómo marcar solicitudes y respuestas, de manera que puedan transferirse a través de protocolos como HTTP

3.2 SOAP

El Protocolo de acceso a objetos simple (SOAP, Simple Object Access Protocol), es una especificación que define cómo los mensajes se pueden enviar entre dos sistemas de software con el uso del lenguaje de marcas extendido (XML, eXtensible Markup Languaje). Estos mensajes suelen seguir un modelo de Solicitud / Respuesta; un equipo hace llamada de método, y el otro equipo realiza cierto cómputo o servicio y, después, devuelve un resultado a la aplicación que

llama. Hay otras maneras de utilizar SOAP, pero ésta es probablemente la más habitual en un principio.
Las palabras acceso a objetos en SOAP pueden ser un poco confusas por que también se tiene acceso a los procedimientos, no necesariamente tienen que ser objetos. Cuando se piensa en los objetos, normalmente se piensa en la definición de objeto de la Programación orientada a objetos (OOP, Object Oriented Programming), en la que un objeto es una representación en código de algo del mundo verdadero o abstracto. Perros, lápices, acceso a datos, etc., entran en esta definición. Por tanto, un objeto contiene algo más que unos cuantos procedimientos sin relación agrupados en un paquete. Los objetos tienen métodos y propiedades que defines sus acciones y atributos, respectivamente.

Aunque se puede usar SOAP para tener acceso a los métodos y a las propiedades de un objeto, SOAP no tiene que ver con OOP. SOAP es simplemente una especificación, una definición del formato que deben tener las solicitudes y las respuestas para que dos aplicaciones que admiten SOAP puedan comunicarse entre sí. SOAP no hace nada por sí mismo. Cada proveedor de plataforma o de software debe decidir cómo implementar lo mejor posible la especificación de SOAP para conseguir maximizar su valor. SOAP sólo tiene significado dentro del contexto de la comunicación entre dos sistemas de software que admitan la especificación.

Una característica de SOAP que le hace sobresalir frente a otros mecanismos, es que la especificación de SOAP fue desarrollada por muchas importantes organizaciones distintas, como Microsoft, IBM y Lotus. Otras organizaciones importantes, como el Grupo de modelado de objetos (OMG, Object Modeling Group), Sun y Oracle también están admitiendo este estándar. La primera especificación fue sometida a consideración por el Consorcio de la World Wide Web (o W3C, un órgano que gobierna los estándares de Internet) en septiembre de 1999.

Sin embargo, dado que SOAP fue creado desde una actitud cercana al protocolo HTTP, su fuerza continúa siendo el formato de solicitud/respuesta que predomina en Internet. Esto no quiere decir que los mensajes no se podrían enviar mediante correo electrónico y ponerse en cola hasta que otro proceso en el servidor compruebe la bandeja de entrada, responda la solicitud, y envíe un correo electrónico a los solicitantes originales.

Relaciones entre SOAP y los servicios web

SOAP es una especificación que define la estructura de un mensaje XML para que se pueda enviar a través de la red para enviar y recibir resultados desde una aplicación. Servicio Web es el término que se utiliza para describir una aplicación que permite disponer de los métodos de SOAP (o de otros tipos). Es importante decir que los servicios Web no son exclusivos de Microsoft, cualquier proveedor de software o de sistema operativo pueden crear servicios Web y no tiene que utilizar necesariamente SOAP para enviar los mensajes. Sin embargo, hasta ahora parece que SOAP y los servicios Web se están utilizados casi como sinónimos, y Microsoft ha acuñado el término con bastante entusiasmo nombrando su puesta en práctica de los servicios Web “servicios Web de ASP.NET”.

SOAP y XML

Varias especificaciones existentes son similares a SOAP, y todas tienen un objetivo: facilitar las llamadas de procedimiento remoto (RPC). RPC es una tecnología que permite a una aplicación llamar un procedimiento (función, método, o lo que se quiera llamar).

SOAP utiliza XML para expresar las llamadas RPC y, por tanto, puede aprovechar las ventajas de XML, así como de sus tecnologías relacionadas.

Un problema de XML es que muchas de las tecnologías relacionadas (como esquemas de XML) todavía están bajo consideración por parte del W3C como "borradores". Sin embargo estas especificaciones están cerca de la fase de "recomendación", así que cualquier cambio actual tendrá poco impacto en la especificación de SOAP.

Además, como la esencia de SOAP es XML, SOAP es, por tanto, texto sin formato (una enorme diferencia frente a los protocolos que tienen un formato binario [IIOP/ ORB / CORBA y DCOM]).

Dado que XML es la opción de SOAP selecciona para expresar las llamadas RPC, esas llamadas con fáciles de leer tanto para un ser humano como para una computadora. Incluso u programador principiante puede leer los documentos de SOAP y determinar exactamente lo que se está llamando, que parámetros se están enviando, etcétera.

Aunque es evidente, para usar SOAP debe tener en cuenta lo siguiente: tanto el que llama como el que recibe deben comprender SOAP. Sin embargo, no debe suponer la dificultad insalvable. Como se ha dicho, los mensajes SOAP son XML simple en teto ASCII y, como tal, se pueden usar prácticamente en cualquier plataforma con cualquier herramienta de software.

La belleza de SOAP es que al ser XML de una manera específica (según la especificación de SOAP), absolutamente cualquier lenguaje de programación o de secuencias de comandos que pueda concatenar, validar y devolver las cadenas de los procedimientos, puede proporcionar y consumir los servicios Web de SOAP.

Los servicios Web se pueden crear usando cualquier lenguaje compatible con .NET Framework, incluyendo Visual Basic y C# así como JScript y otros lenguajes de terceros. Los servicios Web se pueden crear usando un editor de textos sencillo. .NET SDK ofrece las demás herramientas necesarias para compilar, probar y poner a punto el código, pero se podría gozar potencialmente de una productividad mayor mediante Visual Studio .NET para la parte de la programación.

Funcionamiento del comportamiento del servicio web

El comportamiento de los servicios web se implementa como un archivo de control de HTML (HTC). La complejidad de las siguientes tareas y características esta encapsulada y oculta para los programadores que implementan los comportamientos de los servicios Web en sus páginas:

- Comprensión del WDSL del servicio Web.

- Saber cómo dar formato a una petición SOAP basada en el WSDL.
- Saber cómo comunicarse con el origen del servicio Web.
- Traducción de una respuesta SOAP.
- Traducción de un error SOAP.

Puesto que el archivo HTC es simplemente un archivo de texto, puede abrirlo para ver la implementación real de estas características, si lo desea. Una vez haya visto el archivo, lo único que queda por hacer lo siguiente:

- Conectar al comportamiento del servicio Web.
- Identificar el servicio Web que se va a utilizar.
- Llamar a los métodos en el servicio Web.
- Controlar los resultados de las llamadas de los servicios Web.

3.3 ¿Qué es WSDL?

Los servicios Web son muy parecidos a los objetos creados en sus aplicaciones durante años. Estos objetos tienen métodos y propiedades accesibles para que los que se crean instancias a partir de una definición denominada *clase*, parecida a una clase es el Lenguaje de descripción de de servicios Web (WSDL, Web Services Description Languaje), que es un documento XML que define las interfaces de programación de un servicio Web.

Funcionamiento de WSDL

Cuando los clientes desean usar el servicio Web, se les dirige hacia el URI del documento WSDL manualmente o a través de cierto mecanismo automatizado (como UDDI o DISCO). Normalmente, el programador de software no usa un explorador web para tener acceso al archivo de WSDL y para crear manualmente el código para tener acceso al servicio Web, aunque esto sea ciertamente posible. En su lugar, los programadores dirigen su IDE a un archivo de WSDL o usan una utilidad que tenga acceso al archivo WSDL, y después crean el código para tener acceso al servicio Web, según la especificación de WSDL.

A partir de ahí, en teoría no es necesario el archivo WSDL, porque la descripción del servicio Web esta codificada en el código del proxy del usuario. Sin embargo, quizás tenga que volver a utilizar alguna vez el archivo de WSDL.

A su vez, el archivo de WSDL debe ser siempre una descripción completa y exacta del servicio Web exacto que pretende describir. Si no es así, los consumidores no podrán tener acceso correctamente al servicio Web.

3.4 Servicios Web ASP.NET

La tercera puesta en práctica de SOAP por parte de Microsoft son los servicios Web de ASP.NET, que fue introducido junto con .NET Remoting, a finales del verano de 2000 con la versión inicial de .NET SDK.

Los servicios Web de ASP.NET permiten que los programadores desarrollen fácilmente las aplicaciones basadas en SOAP que se crean en el ambiente de las aplicaciones de Páginas Active Server (ASP, Active Server Pages).

El procesador de ASP se implementó como una extensión de la Interfaz de programación de aplicaciones de Internet Server (ISAPI, Internet Server Application Programming Interface) (asp.dll). ISAPI es una interfaz de programación que los programadores pueden usar para ampliar las funciones de los Servicios de Internet Information Server.

La asp.dll es simplemente un objeto COM que admite interfaces y tipos específicos para convertirse en una extensión de ISAPI. Para ASP, el tipo de archivo .asp esta asociado con la asp.dll.

Problemas de las Páginas Active Server

El modelo clásico de ASP hizo su servicio bien, pero tenía algunas limitaciones inherentes que se podían optimizar, cosa que se haría tarde o temprano. Algunas de estas limitaciones y problemas, cuestiones de rendimiento, de mantenimiento, de administración de estado y de uso de componentes COM.

Rendimiento

El analizador de ASP es un software increíble que puede controlar todo el código ASP que se le entregue, esté bien o mal escrito. Sin embargo, últimamente, incluso después de que el código se haya analizado y almacenado en caché, sigue siendo secuencias de comandos que confían en una maquina Active Scripting para ejecutarse.

Mantenimiento

Las páginas ASP son una mezcla de HTML y de secuencia de comandos de servidor, como VBScript o JScript. Además, las páginas podrían contener otra secuencia de comandos de cliente, como VBScript o Java Script. Con estas distintas secciones de código mezcladas a lo largo del documento, van intentando implementar reglas de negocio más complejas en sus páginas web, se hace más evidente la necesidad de separación en el código.

Administración de estado

La administración de estado era una de las características más importantes de ASP en sus inicios, debido a las soluciones de cosecha propia se generaron los programadores. Sin embargo, la

administración clásica de estado de ASP ya contenía defectos inherentes, algunos de los cuales se combinaron con técnicas defectuosas de codificación y uso correcto por parte de los programadores. Hay cuatro aspectos relacionados con la administración de estado usando los objetos Application y Session ASP clásico:

- Estos objetos requieren que las *cookies* del explorador de web del usuario estén activadas. Un identificador único se guarda en una *cookie* que permite que la información de estado se corresponda con la solicitud del usuario y la información almacenada en caché de la *cookie.*

- Si el usuario no actúa durante un periodo de tiempo definido, las *cookies* caducan y los datos de usuario se pierde n para siempre.

- Los programadores guardaban las referencias de objeto **Application y Session**, lo que causaba importantes problemas de memoria.

- El estado de sesión no funciona usando las *cookies* a través de una granja Web porque cada sesión se crea desde un único servidor web; no esta diseñado para la transferencia a múltiples servidores.

Usos de componentes COM

Cuando fue necesario tener acceso a datos, implementar seguridad, enviar correo, o cualquier otra cosa, era necesario hacer referencia a un componente COM. Aunque es una opción viable que ha demostrado un cierto éxito, tiene problemas de configuración, de implementación y de rendimiento. En un mundo ideal, todos estos servicios estarían integrados dentro de ASP para que al basarse en ellos existiera una mayor integración con la aplicación.

Microsoft .NET Framework

El equipo de desarrollo de ASP buscó una manera más adecuada de crear las aplicaciones Web que mejorarían la velocidad, para proporcionar servicios más integrados en el sistema, mejorar la administración de estado, y separar el código HTML (la interfaz) de la secuencia de comandos (reglas de negocios). En resumen, el objetivo era facilitar la creación de aplicaciones basadas en Web. Otro objetivo era ofrecer a los programadores nuevas formas de tener acceso a los puntos de ampliación dentro de este nuevo marco de trabajo ASP, que permitiría más cosas a los programadores (es decir, todos los programadores que no fueran programadores de C++). La culminación de los esfuerzos del equipo de ASP, y de otros equipos de Microsoft, culminó en el mayor avance realizado hasta ahora conocido como .NET Framework.

CAPITULO 4
BASES DE DATOS

4.1 ¿Qué son las bases de datos?

Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente. A continuación te presentamos una guía que te explicará el concepto y características de las bases de datos.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una **base de datos** se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

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

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más **columnas** y **filas**. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Definición de base de datos

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

Características

Entre las principales características de los sistemas de base de datos podemos mencionar:

- ✓ Independencia lógica y física de los datos.
- ✓ Redundancia mínima.
- ✓ Acceso concurrente por parte de múltiples usuarios.
- ✓ Integridad de los datos.
- ✓ Consultas complejas optimizadas.
- ✓ Seguridad de acceso y auditoria.
- ✓ Respaldo y recuperación.
- ✓ Acceso a través de lenguajes de programación estándar.

Sistema de Gestión de Base de Datos (SGBD)

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las

aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Ventajas de las bases de datos

Control sobre la redundancia de datos:

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.

En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

Consistencia de datos:

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

Compartición de datos:

En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

Mantenimiento de estándares:

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

Mejora en la integridad de datos:

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Mejora en la seguridad:

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

Mejora en la accesibilidad a los datos:

Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

Mejora en la productividad:

El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación.

El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

Mejora en el mantenimiento:

En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.

Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados.

Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

Aumento de la concurrencia:

En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

Mejora en los servicios de copias de seguridad:

Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que

hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.

En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

Desventajas de las bases de datos

Complejidad:

Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.

Coste del equipamiento adicional:

Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

Vulnerable a los fallos:

El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse. Es por ello que deben tenerse copias de seguridad (Backup).

Tipos de Campos

Cada Sistema de Base de Datos posee tipos de campos que pueden ser similares o diferentes. Entre los más comunes podemos nombrar:

- **Numérico:** entre los diferentes tipos de campos numéricos podemos encontrar enteros “sin decimales” y reales “decimales”.
- **Booleanos:** poseen dos estados: Verdadero “Si” y Falso “No”.
- **Memos:** son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados.
- **Fechas:** almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.
- **Alfanuméricos:** contienen cifras y letras. Presentan una longitud limitada (255 caracteres).
- **Autoincrementables:** son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta: Servir de identificador ya que resultan exclusivos de un registro.

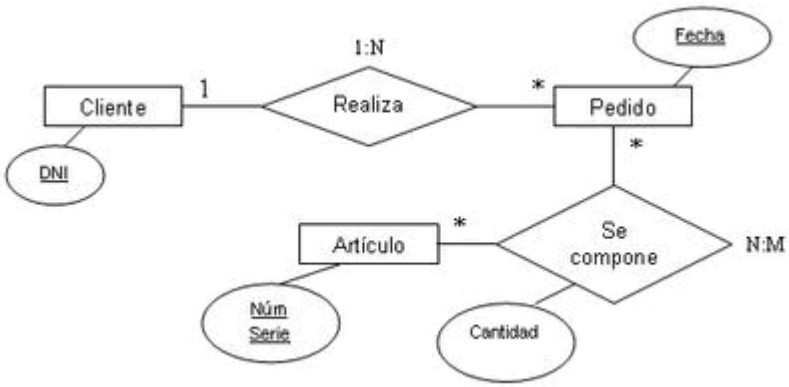
Tipos de Base de Datos

Entre los diferentes tipos de base de datos, podemos encontrar los siguientes:

- **MySql:** es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.
- **PostgreSql y Oracle:** Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.
- **Access:** Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa Access, el cual crea un archivo .mdb con la estructura ya explicada.
- **Microsoft SQL Server:** es una base de datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.

Modelo entidad-relación

Los diagramas o modelos entidad-relación (denominado por su siglas, ERD "Diagram Entity relationship") son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.



Estructura de una Base de Datos

Una base de datos, a fin de ordenar la información de manera lógica, posee un orden que debe ser cumplido para acceder a la información de manera coherente. Cada base de datos contiene una o más tablas, que cumplen la función de contener los campos.

	Field	Type
☐	id	int(11)
☐	titulo	varchar(100)
☐	texto	blob
☐	fecha	varchar(10)

Los datos quedarían organizados como mostramos en siguiente ejemplo:

←T→			id	titulo	texto	fecha
☐	✎	✕	1	saludos	[BLOB - 0 B]	22-10-2007
☐	✎	✕	2	como estas ???	[BLOB - 0 B]	23-10-2007

Por consiguiente una base de datos posee el siguiente orden jerárquico:

- Tablas
- Campos
- Registros
- Lenguaje SQL

El lenguaje SQL es el más universal en los sistemas de base de datos. Este lenguaje nos permite realizar consultas a nuestras bases de datos para mostrar, insertar, actualizar y borrar datos.

4.2 Base De Datos Relacionales

Se conoce como base de datos relacional a una representación lógica de datos, que permite acceder a los mismos de manera independiente de su estructura física, dichos datos se encuentran organizados en tablas. Una tabla se encuentra compuesta por filas y columnas, para entender mejor esta idea de representación, piense que la tabla es una cuadricula, donde las filas son los registros de la tabla (los datos adquiridos o introducidos y que hacen referencia a un elemento básico de información a almacenar en la base de datos) y las columnas son los campos de la tabla (los tipos de datos de los que está compuesto un registro). Para visualizar mejor obsérvese el siguiente ejemplo de tabla:

	Número	*Nombre*	*Departamento*	*Salario*	*Ubicación*
	22056	Juan	412	1200	México
Fila →	23123	Miguel	413	1800	Guadalajara
	23765	Omar	412	2500	México
	25059	Carlos	423	2000	Pachuca
	25275	Iván	625	5000	Monterey
	86586	Daniel	412	25000	México
	↑ **Clave Primaria**		↑ **Columna**		

Fig. 4.2.1 Tabla empleados

En la tabla la columna "Número", es la clave primaria; una clave primaria es una columna o grupo de estas que, que requiere un valor único que no puede duplicarse en otras filas.

4.3 SQL

¿Qué es?

SQL (Structured Query Language), en español Lenguaje de Consulta Estructurado, es un lenguaje de acceso a bases de datos relacionales que permite gran variedad de operaciones sobre las mismas. Es el lenguaje más universal existente para trabajar con bases de datos.

Estándar ISO y ANSI. Se puede insertar dentro del código de la mayoría de lenguajes de programación para así acceder a datos de BD. Se puede emplear dentro de cualquier base de datos relacional actual (Oracle, Access...)

El SQL se compone principalmente de:

- Lenguaje de Descripción de Datos (DDL), permite las tareas de definición de las estructuras que almacenarán los datos, como de los procedimientos o funciones que permitan consultarlos.

- Lenguaje de Manipulación de Datos (DML), permite llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

Entre las funciones principales del SQL se pueden destacar:

- Definición de datos
- Recuperación de datos
- Manipulación de datos
- Control de acceso
- Compartición de información
- Integridad de datos

Historia

La historia de SQL (que se pronuncia deletreando en inglés las letras que lo componen, es decir "ese-cu-ele" y no "siquel" como se oye a menudo) empieza en 1974, cuando un grupo de IBM desarrolló un lenguaje para la especificación de las características de las bases de datos que adoptaban el modelo relacional. Este lenguaje se llamaba SEQUEL (Structured English Query Language). Sin embargo, fue ORACLE quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el antecesor del SQL. En 1986 es estandarizado por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86. Al año siguiente, en 1987, este estándar es también adoptado por la ISO.

En los años siguientes, éste ha sufrido diversas revisiones que han conducido a la versión SQL-89 y, posteriormente, se lanza un nuevo estándar ampliado y revisado llamado SQL-92.

El hecho de tener un estándar definido por un lenguaje para bases de datos relacionales abre potencialmente el camino a la intercomunicación entre todos los productos que se basan en él. Desde el punto de vista práctico, por desgracia las cosas fueron de otro modo. Efectivamente, en general cada productor adopta e implementa en la propia base de datos sólo el corazón del lenguaje SQL (el así llamado Entry level o al máximo el Intermediate level), extendiéndolo de manera individual según la propia visión que cada cual tenga del mundo de las bases de datos.

Revisiones y agregados que sufrió a lo largo del tiempo

SQL	Comentarios
SQL-86	Primera publicación hecha por ANSI. Confirmada por ISO en 1987
SQL-89	Revisión menor
SQL-92	Revisión mayor

SQL:1999	Introduce expresiones regulares y consultas recursivas
SQL:2003	Introduce algunas características de XML

Estructura

El lenguaje de recuperación SQL está compuesto por:

- COMANDOS. Existen dos tipos de comandos SQL:
  DLL. Permite crear y definir nuevas bases de datos, tablas, campos...

  DML. Permite generar consultas para extraer datos de interés.
- CLÁUSULAS. Condiciones de modificación para seleccionar los datos.
- OPERADORES. Permiten enfocar la búsqueda vinculando términos de búsqueda y definiendo la relación entre ellos.
- FUNCIONES DE AGREGADO. Devolver un único valor que se aplica a un grupo de registros.

Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos

COMANDOS DLL

CREATE. Crear una tabla nueva

ALTER. Añadir, modificar, eliminar campos y claves de una tabla

DROP. Eliminar una tabla

COMANDOS DML

INSERT. Añadir registros o modificar datos

UPDATE. Actualizar registros, cambia el valor de una o varias celdas por un nuevo valor

DELETE. Eliminar los registros que cumplan alguna condición

SELECT. Selección registros

CLÁUSULAS

FROM. Especificar la tabla de la cual se van a seleccionar los registros

WHERE. Especificar las condiciones que debe cumplir los registros seleccionados

GROUP BY. Condición para agrupar por campo/s determinados datos

HAVING. Expresar la condición para agrupar datos determinados

ORDER BY. Ordenar los registros o campos seleccionados con orden

ASC. Ordenar ascendente

DESC. Ordenar descendente

OPERADORES

AND. Operador booleano Y, devuelve valores que cumplan ambas condiciones

OR. Operador booleano O, devuelve valores que cumplan al menos una condición

LIKE. Utilizado para la comparación de un modelo

BETWEEN. Utilizado para especificar un intervalo de valores

IS NOT NULL / IS NOT NULL. Si permite campos vacíos o no

NOT. Negación, devuelve el valor contrario de la expresión

=. Igual que

<. Mayor que

>. Menor que

<>. Distinto de

FUNCIONES DE AGREGADO

AVG. Calcular el promedio (media) de los valores de un campo determinado

COUNT. Contar el número de registros de la selección

SUM. Sumar todos los valores de un campo determinado

MAX. Recuperar el valor máximo de un campo especificado

MIN. Recuperar el valor mínimo de un campo especificado

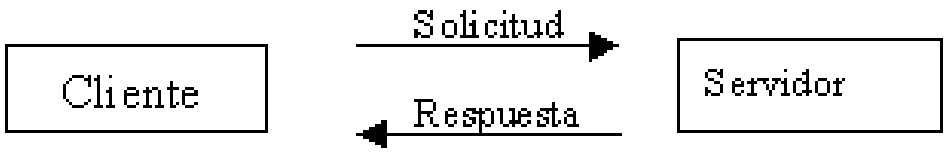
4.4 Modelo Cliente-Servidor.

¿Qué es Cliente/servidor?

Definición del modelo cliente/servidor

La tecnología cliente/servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. Desde el punto de vista funcional, se puede definir la computación cliente/servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aun en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.



Modelo cliente servidor

La idea es tratar a una computadora como un instrumento, que por sí sola pueda realizar muchas tareas, pero con la consideración de que realice aquellas que son más adecuadas a sus características. Si esto se aplica tanto a clientes como servidores se entiende que la forma más estándar de aplicación y uso de sistemas clientes/servidores es mediante la explotación de las PC a través de interfaces gráficas de usuario; mientras que la administración de datos y su seguridad e integridad se deja a cargo de computadoras centrales tipo mainframe.

Como se desprende de las definiciones anteriores, tanto clientes como servidores son entidades independientes que operan conjuntamente a través de una red para realizar una tarea. Pero para hacer la distinción respecto de otras formas de arquitecturas o software distribuidos, se presenta una lista de características que debieran cumplir los sistemas cliente/servidor:

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos lo que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- Las plataformas de software y hardware entre clientes y servidores son independientes. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema cliente/servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores

Componentes del modelo Cliente/Servidor

- ✓ Nivel de Presentación: Agrupa a todos los elementos asociados al componente Cliente.
- ✓ Nivel de Aplicación: Agrupa a todos los elementos asociados al componente Servidor.
- ✓ Nivel de comunicación: Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor.
- ✓ Nivel de base de datos: Agrupa a todas las actividades asociadas al acceso de los datos.

Cliente

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- ➢ Administrar la interfaz de usuario.
- ➢ Interactuar con el usuario.
- ➢ Procesar la lógica de la aplicación y hacer validaciones locales.
- ➢ Generar requerimientos de bases de datos.

- Recibir resultados del servidor.
- Formatear resultados.

Servidor

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

CAPITULO 5
ANÁLISIS Y DIAGNOSTICO

5.1 Infraestructura actual

Dentro de la infraestructura actual de los sitios web de servicios (sitios para trámites y de carácter informativo) de la ESIME Zacatenco no se cuenta con un directorio general para consultar profesores, cubículos, horarios, adscripción académica, asignatura, profesor tutor, etc. Como se puede apreciar en los siguientes sitios web:

- http://www.esimez.ipn.mx/index1.htm
- http://148.204.219.23/cescolar/Index.asp
- http://www.sigue.esimez.ipn.mx/
- http://148.204.19.11/academias/
- http://148.204.19.11/profesores/

En el sitio http://148.204.219.23/cescolar/Index.asp dedicado a los alumnos, para trámites de inscripción, altas, bajas, consulta de horario, inscripción de exámenes y horarios de exámenes y grupos, posee solo una sección donde el alumno puede consultar el horario de todos los grupos, así como el profesor de cada asignatura de cada grupo (Fig. 5.1 y Fig. 5.2).

Fig. 5.1 **Página de control escolar opción de consulta de horarios.**

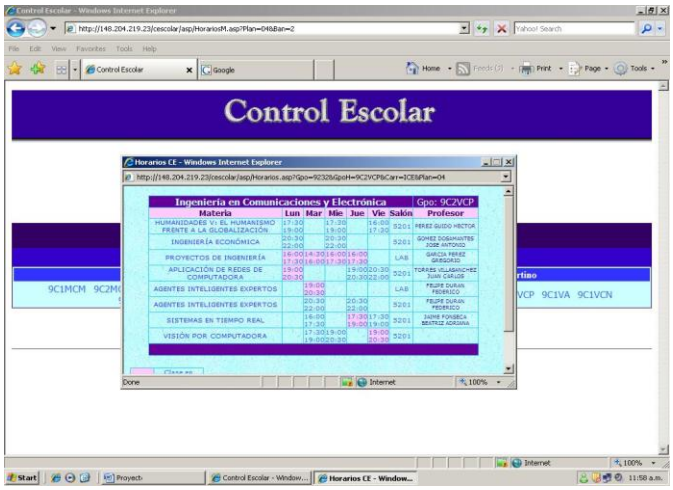


Fig. 5.2 **Página de control escolar consultando horario, grupo, asignatura y Profesores.**

En este sitio web del control escolar se puede conocer, al profesor de asignatura, horario de cada grupo, **pero no es posible conocer todo el horario de un determinado profesor**, así como su horario de atención en cubículo, extensión telefónica y ubicación de cubículo.

El sitio web http://148.204.219.23/cescolar/Index.asp, se encuentra bajo la administración del departamento de control escolar, dicho departamento tiene como función principal, llevar a cabo tramites de inscripción para los alumnos, aunque este sitio posee una sección en la cual el alumno puede consultar el horario de los grupos, no es posible localizar el horario de cubículo de un profesor, ni tampoco le permite a aquellos alumnos en el programa de tutorías consultar el horario de su profesor tutor.

Un sitio web que posee un directorio es http://www.esimez.ipn.mx/extensiones/ICE.HTM este sitio cuenta con un directorio de extensiones telefónicas, de academias y departamentos, de igual forma que el sitio web del departamento de control escolar no es posible obtener el horario de cada profesor, cubículo, academia etc., (ver Fig. 5.3).

Fig. 5.3 **Página de ESIME Zacatenco que contiene extensiones telefónicas de los profesores del departamento de ICE.**

El departamento de horarios ha puesto a disposición de los profesores y academias, un par de sitios web, que permiten a profesores y presidentes de academia hacer modificaciones en el horario de clase de los profesores, estas operaciones de modificaciones son comúnmente llevadas a cabo y en un gran número antes de que se publique el horario del semestre siguiente.

Como se mencionó los sitios http://148.204.19.11/profesores/ y http://148.204.19.11/academias/ están enfocados a profesores, no cuentan con una página donde un usuario que no esté registrado (un usuario que no sea profesor) pueda consultar los horarios de profesores, debido a que es un sitio de carácter administrativo (ver figuras 5.4 y 5.5).

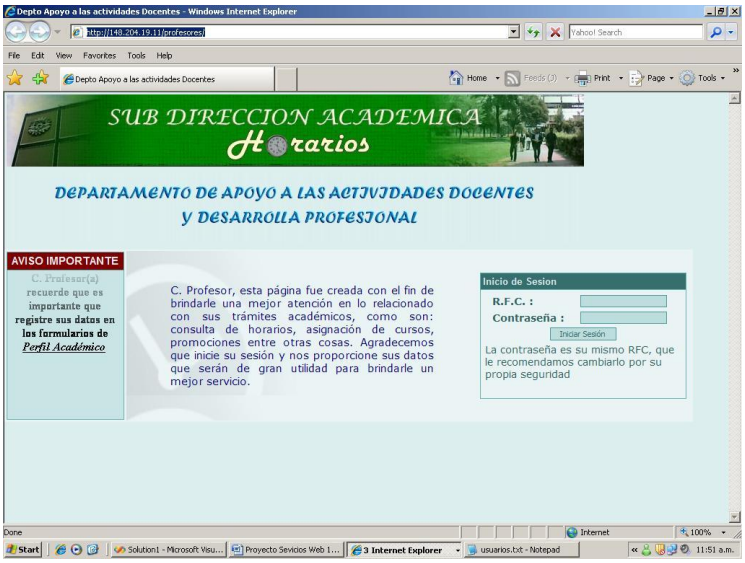


Figura 5.4. **Página de ESIME Zacatenco de carácter administrativo dirigida a profesores.**

Figura 5.5 **Página de ESIME Zacatenco de carácter administrativo dirigida a los presidentes de academias.**

De acuerdo con la infraestructura actual, el grupo de páginas dedicado a la información de horarios, no permiten a cualquier tipo de usuario acceder a la información determinada, ya sea porque no es parte del personal administrativo ó por que el servicio dedicado a su perfil no tiene esa capacidad (ver figura 5.6).

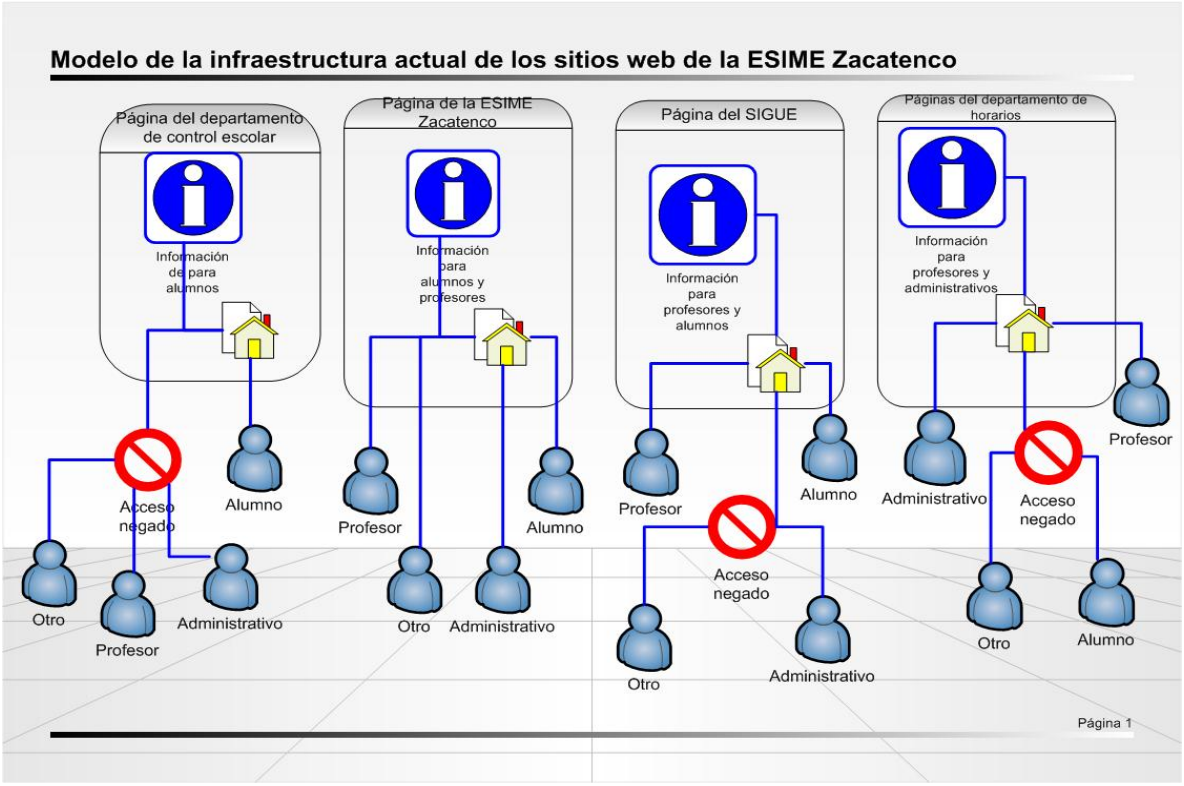


Fig. 5.6 **Estado actual de los sitios web de la ESIME Zacatenco.**

5.2 Arquitectura de los servicios actuales

Para poder conocer la arquitectura y operación de los servicios actuales, se tuvo una entrevista con el administrador del servidor en el departamento de horarios, la profesora Catalina Patiño Gallegos. El administrador expuso la arquitectura actual de las páginas: http://148.204.19.11/profesores/, http://148.204.19.11/academias/ y http://www.ce.esimez.ipn.mx/ (esta última en su sección de horarios). En la arquitectura actual está basada en un servidor local de tipo “personal web server”, este tipo de servidores web permiten publicar sitios, y permitir responder a cualquier cliente que haga una petición por el puerto configurado, el cual generalmente es el 80 para el protocolo HTTP, como dato adicional se sabe que el web server no es IIS de Microsoft.

El tipo de consultas en este servidor son llevadas a cabo utilizando páginas web de tecnología ASP.net, el código de las páginas ASP se encuentra en lenguaje Visual Basic, el cual permite correr scripts (pequeños secciones de código que son ejecutadas sin necesidad ser compiladas). Las consultas de a la base de datos son enviadas a un manejador Microsoft Access que reside en la misma máquina física que el web server.

Dicha base de datos posee alrededor de diecisiete tablas que son modificadas continuamente mediante la interface web.

Aunque el manejador Microsoft Access (ver figura 5.7), posee herramientas para operar como servidor de base de datos cuenta con un nivel de seguridad y capacidad para brindar servicios mucho menor que otros manejadores como Microsoft SQL server o MySql.

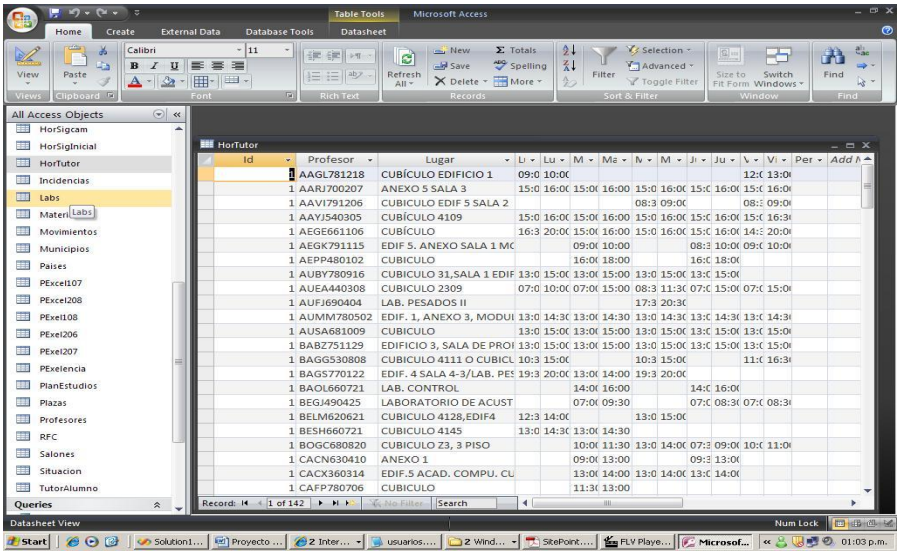


Fig. 5.7 IDE del manejador Microsoft Access.

5.3 Modelo de funcionamiento actual

Como se mencionó anteriormente el servidor web y el servidor de la base de datos, residen en un mismo equipo, este equipo responde y lleva a cabo todas las operaciones solicitadas por los usuarios en los sitios del departamento de horarios, este equipo del tipo Workstation, este tipo de computadoras están diseñadas para tareas especificas que demanden un desempeño mayor que una PC estándar, e incluso sus capacidades de almacenamiento y procesamiento suelen ser mayores. Aunque el equipo no ha presentado problemas cuando hay una demanda en las consultas del servidor web, este equipo es usado por el personal del departamento de horarios, para realizar tareas propias del departamento como son; elaboración de documentos, cambio de registros, impresiones, navegación, uso del correo electrónico, etc.

La implementación de este equipo como servidores y estación de trabajo (ver figura 5.8), tiene vicisitudes propias, en cuanto a las ventajas se tiene que se está ahorrando físicamente equipos de servidores, ya que están siendo implementados en uno solo, esto representa ahorro de recursos, económicos y de hardware. Otra ventaja que se tiene es que es posible manipular las bases de datos directamente en la misma computadora sin necesidad de interfaz, esto es benéfico en caso de que el administrador esté familiarizado con el ambiente del manejador, en caso contrario no es una ventaja.

En cuanto a las desventajas se tiene que: Primeramente tener dos servidores ejecutándose en un mismo equipo es potencialmente peligroso, ya que en caso de fallo por hardware o software del mismo, ambos servidores quedarían fuera de servicio incrementando los problemas en el sistema de información de la ESIME Zacatenco, el segundo aspecto negativo, es que los procesos que el equipo éste ejecutando, entre estos los propios del sistema operativo, del servidor web, del servidor de base de datos, y de las tareas de usuario tienen prioridades que al final de cuentas se verán afectados por el grupo que mayor demanda tenga. Otro aspecto es que al ser un equipo del tipo Workstation es un equipo que no tiene un UPS (sistema de energía ininterrumpido por sus siglas en inglés), que le permita estar en activo y auto apagarse y reiniciarse en caso de fallas eléctricas. El último aspecto y quizá el más trivial es que en el mismo edificio donde se encuentra el equipo que actualmente funciona como servidores del departamento de horarios, existe otro equipo del tipo servidor dedicado, que posee todas las características necesarias como son: UPS, espacio en disco, software de desarrollo y manejo de base de datos, conexión a la red, dicho equipo no ha sido puesto en uso y se encuentra disponible para este propósito.

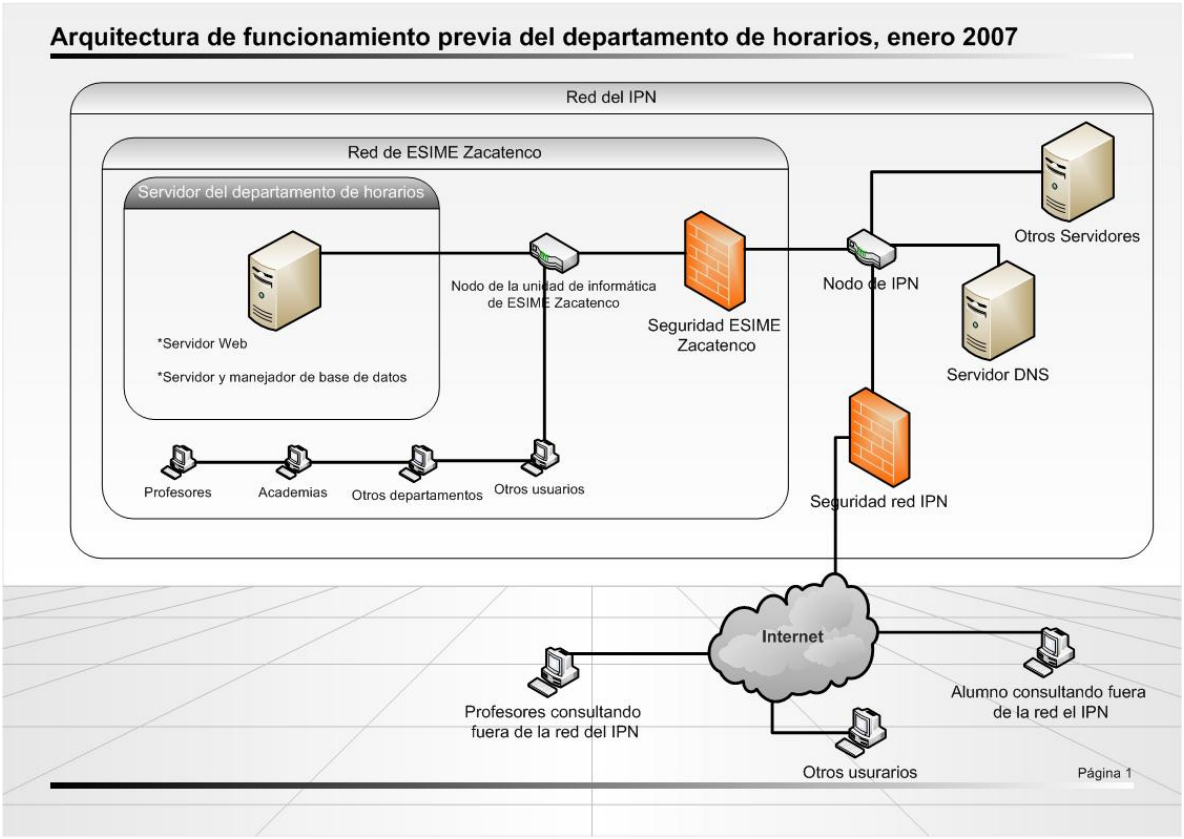


Fig. 5.8 Estado de los servidores que proporcionan el servicio de consultas dentro de ESIME y el Politécnico en general.

5.4 Estudio del Mercado

A continuación se describen los pasos para realizar una investigación de mercados:

Identificación del problema

¿Cuál es el problema que quiero resolver?

El problema consiste en cubrir la necesidad de la falta de un medio que permita a los alumnos de ESIME Zacatenco tener información de sus profesores para poder localizarlos y poder tratar información relacionada con las actividades académicas

¿Qué pregunta quiero contestar con esta investigación?
La necesidad de tener un medio ó procedimiento para saber el horario de atención de los profesores de las diferentes asignaturas, lugares de estancia después de sus clases en muchas ocasiones con la necesidad de reunirse con el motivo de apoyo extra clase, entrega de tareas, etc.

Cabe mencionar que aunque es un protocolo común entre profesores informar cual es su área de cubículos y su nombre completo al momento de presentarse por primera vez ante un grupo no siempre esta información es válida durante todo el curso, ya que por motivos administrativos (ajustes en el horario de las asignaturas y profesores que la imparten) el horario de trabajo y por lo tanto el tiempo de estancia en los cubículos de los profesores variaría a lo largo del semestre.

Objetivos.

¿Qué objetivos plantearía para ingresar su producto al mercado?
Entonces respondiendo a la pregunta, quiero brindar un fácil, práctico y amigable acceso a esta información dirigido a la comunidad estudiantil y a los profesores de ESIME Zacatenco.
Como objetivos proponemos resolver esta necesidad con el simple hecho de acceder al sitio
Y consultando en la base de datos.

Buscar fuentes de información:

Nuestras fuentes de información fueron sacadas directamente de la comunidad estudiantil, de las páginas de horarios de la ESIMEZ Zacatenco así como de la de control escolar.

Instrumentos para recolección de datos

En este caso nuestros instrumentos fueron estas preguntas a modo de encuesta:

1.- ¿Has tenido problemas para saber tu horario de clases?

Si ( ) No ( ) Algunas Veces (x)

2.- ¿Has tenido problemas en encontrar a tus profesores de clase?

Si (x) No ( ) Algunas Veces ( )

3.- ¿Sabes la ubicación del cubículo de tu profesor, así como el teléfono y extensión?

Si ( ) No (x) Algunas Veces ( )

4.- ¿Sabes el horario de tu profesor?

Si ( ) No (x) Algunas Veces ( )

5.- ¿Sabes el nombre completo de tu profesor?

Si ( ) No (x) Algunas Veces ( )

6.- ¿Sabes el horario de atención a alumnos?

Si ( ) No (x) Algunas Veces ( )

7.- ¿Te gustaría poder consultar todos estos datos sin problema alguno?

Si (x) No ( ) Algunas Veces ( )

8.- ¿Te gustaría que existiera una página para saber el horario de atención de tus profesores?

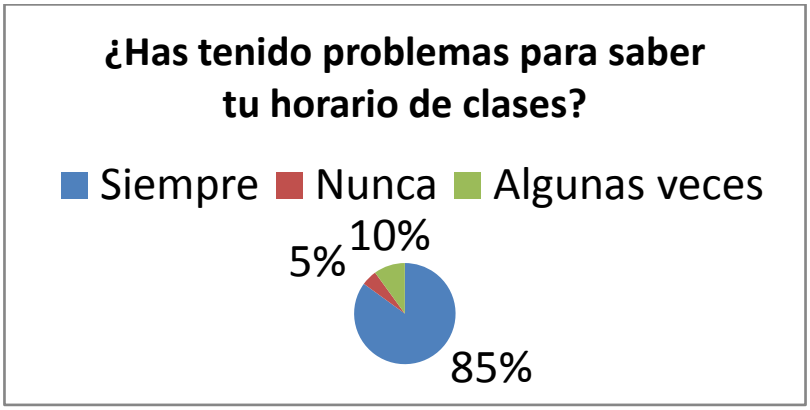
Si (x) No ( )

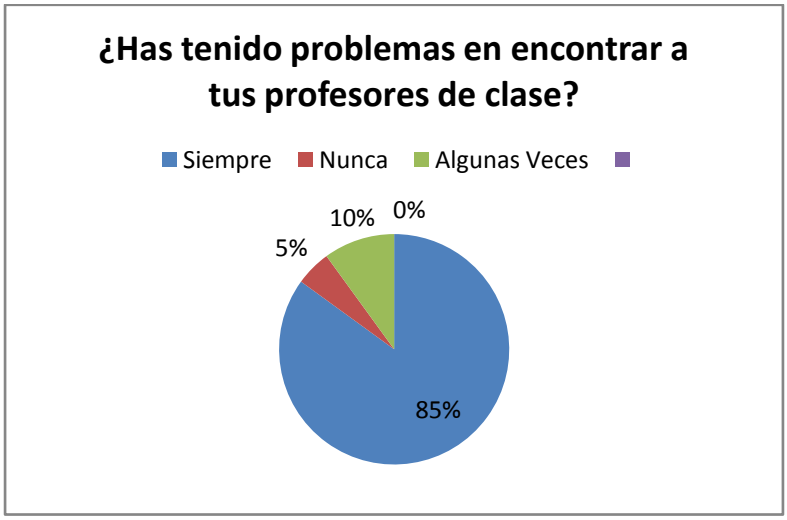
Análisis de datos

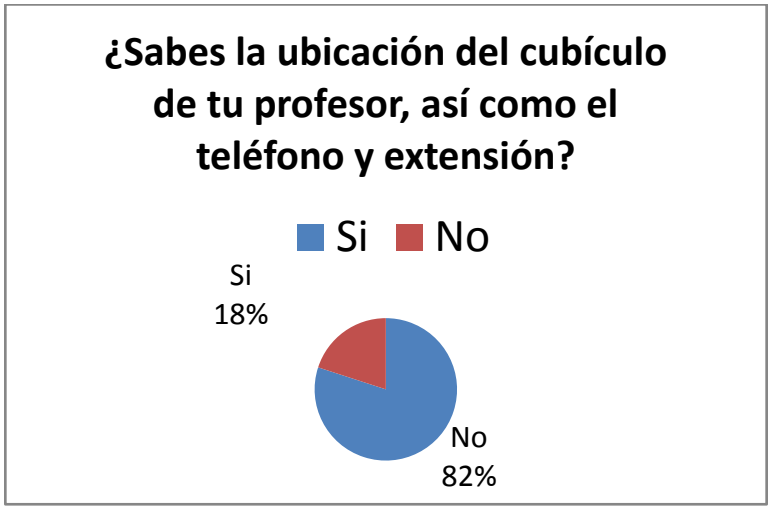
De acuerdo a la muestra obtenida, las preguntas 1, a la 6 reflejan la existencia de una necesidad de parte de los alumnos, de un medio que permita a los alumnos tener un medio que provea información de donde pueden localizar y estar en contacto con sus profesores.
Así de acuerdo al estudio realizado, que en este caso seis preguntas realizadas entre la comunidad politécnica de la ESIME en lo cual nosotros nos apoyamos como nuestro estudio de mercado las preguntas están resueltas ya que ofrecemos una búsqueda dirigida al administrador, profesor y alumno para saber en primer lugar nombre completo del profesor , la materia que este imparte la academia a la que pertenece, el turno en el que esta ,su horario de clases algo muy importante que es el lugar de ubicación fuera de clases que corresponde a su cubículo muy útil este ultimo debido a que alumnos pueden encontrarlo para asesorías y distintos tipos de necesidades .

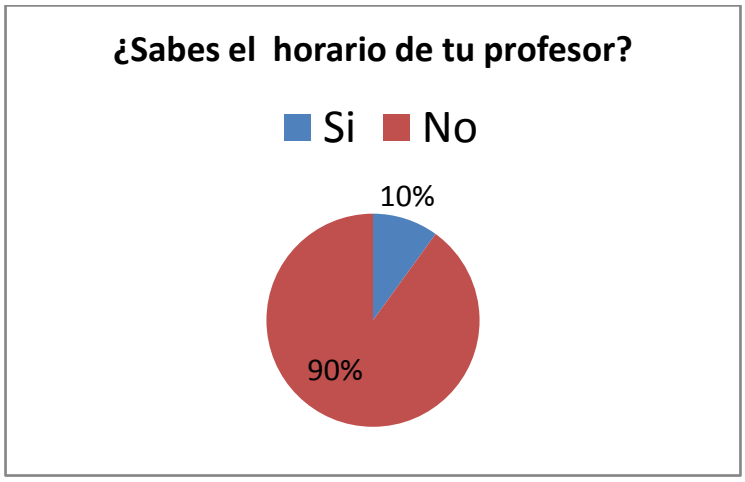
Con estas funciones de búsqueda ofrecidas abarcamos la falta de información que uno como alumno debería tener al mismo tiempo se incluye a la comunidad tanto de de profesores como jefes de academia , que vendrían siendo algunos administradores teniendo la característica de poder insertar , modificar y eliminar en la base de datos.

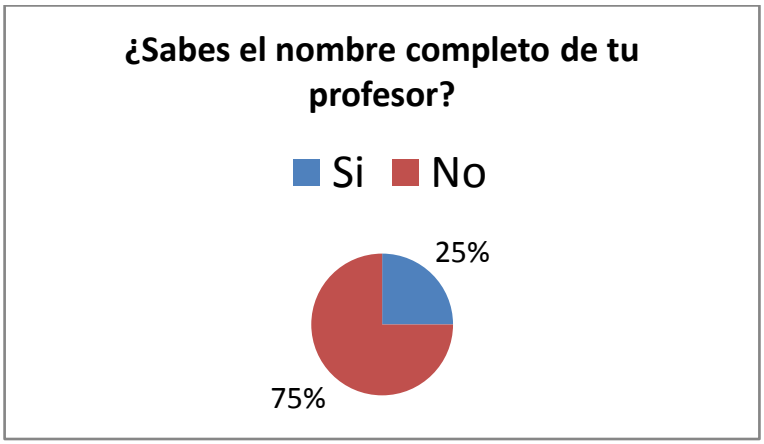
Gráficas de las Encuestas

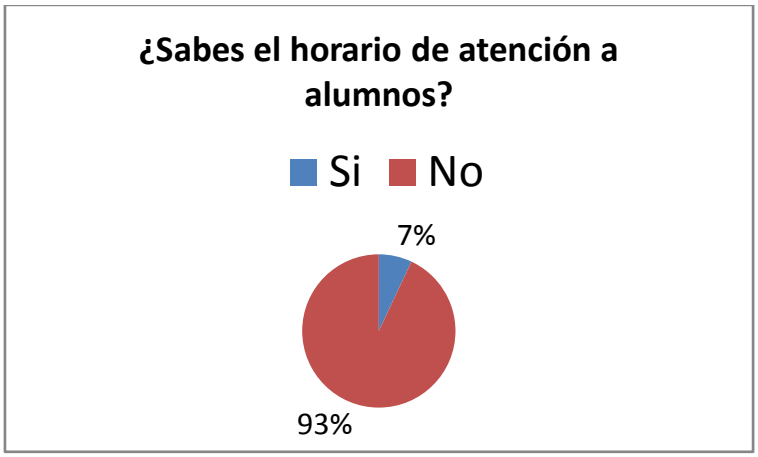




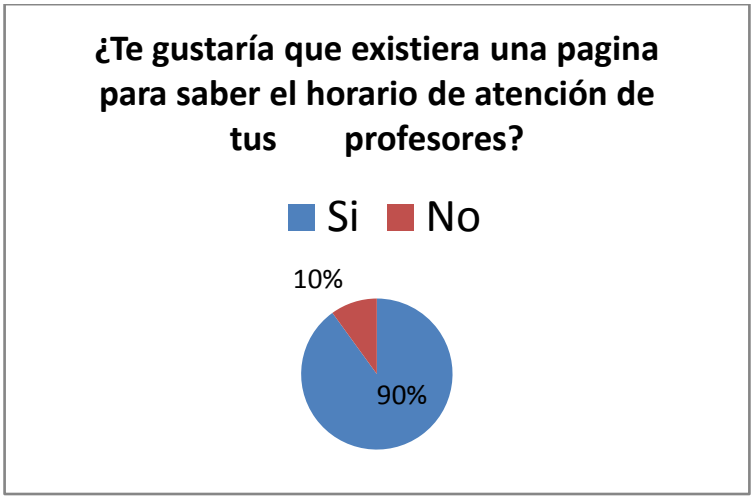



¿Sabes el horario de tu profesor?
Si
No
10%
90%


¿Sabes el nombre completo de tu profesor?
Si
No
25%
75%


¿Sabes el horario de atención a alumnos?
Si
No
7%
93%


¿Te gustaría poder consultar todos estos datos isn problema alguno?
Si
No
3%
97%


¿Te gustaría que existiera una pagina para saber el horario de atención de tus profesores?
Si
No
10%
90%

5.5 Análisis De Costos

El desarrollo de nuevas aplicaciones lleva unas etapas que conviene no omitir:

- Detección de la necesidad y resultados requeridos
- Análisis del sistema
- Desarrollo
- Pruebas
- Puesta en marcha y capacitación

Esta enumeración de tareas no debe asustar a nadie, pues muchas veces el primer punto se completa en un par de días y el segundo en otro par de días. La importancia de no saltear estos temas radica en que de esta forma se obtendrá el producto final exacto que el cliente requiere.

Sin duda, integran el estudio del proyecto tanto el presupuesto disponible como el hardware instalado y los recursos humanos que se asignarán. Todo debe estar equilibrado dentro de las capacidades de la empresa, de modo que la incorporación de un nuevo soft produzca claros beneficios y no complicaciones.

Costos del Software

Los costos del software a menudo dominan al costo del sistema. El costo del software en un PC es a menudo más caro que la PC.

- Productos de Software
- Productos genéricos.
- Productos que son producidos por una organización para ser vendidos al mercado.
- Productos hechos a medida.
- Sistemas que son desarrollados bajo pedido a un desarrollador específico.

La mayor parte del gasto del software es en productos genéricos, pero hay más esfuerzo en el desarrollo de los sistemas hechos a medida.

El software desarrollado para servicios escolares es un producto hecho a la medida, ya que tenemos un programa que consulta horarios, ubicación, y nombres de los profesores, los usuarios son de tres tipos, alumnos, maestros, y administradores.

Los administradores pueden consultar, insertar, modificar y eliminar.

Existen aplicaciones parecidas pero esta se adecuo a la ESIME Zacatenco pero es válida para cualquier tipo de administración escolar.

El desarrollo de un servicio web en este caso dedicado al departamento de servicios escolares de la institución está compuesto por las siguientes etapas: el análisis y diagnostico que constas de definir la situación del sistema y el contemplar la forma de resolver el problema, el diseño de la estructura del sistema es decir planificar y crear el medio de interacción entre la aplicación y el usuario, al mismo tiempo se contempla y se busca la manera práctica de llevar esto al lenguaje de programación lo que da pauta a buscar las herramientas necesarias enfocadas la mayor parte en el software, una vez teniendo el diseño y las herramientas pasamos al desarrollo , la parte de programación en donde radica la mayor parte del tiempo invertido y el costo, podemos decir que la parte de pruebas , que es la siguiente etapa es la ultima donde tenemos la parte de corrección de fallos y mejoras que encontremos ya que este servicio web es parte de un dominio omitiremos la parte de la capacitación teniendo únicamente la puesta en marcha.

Partiendo de nuestra siguiente tabla estimada tenemos:

Análisis y diagnostico	$D4 = $42.12 x hora
Diseño:	$D5 = $52.65 x hora
Desarrollo:	$D6=$63.18 x hora
Implementación:	$D4 = $42.12 x hora

COSTOS Y GASTOS VARIABLES POR UNIDAD PRODUCIDA: $22 270.95 (suma de todos los gastos).

Análisis y diagnostico:

Llevado a cabo durante dos semanas trabajando 2 horas por día teniendo un total de 20 horas multiplicado por tres personas lo que nos da un total de 60 horas totales.

De acuerdo con nuestra tabla tenemos:

US$4 = $42.12 x hora

$42.12 x 60 horas =$2 527.20

Diseño:

Llevado a cabo durante otras dos semanas 2.5 horas diarias total de 25 horas por tres persona nos da un total de 75 horas.

US$5 = $52.65 x hora

$52.65 x 75 horas =$ 3 948.75

Desarrollo:

Desarrollada durante un mes 3.5 horas diarias nos da un total de 70 horas diarias por tres personas 210 horas.

US$6=$63.18 x hora

$63.18x 210 horas =13 267.80

Implementación:

Llevada durante dos semanas 2 horas diarias nos da un total de 20 horas por tres personas tenemos 60 horas.

US$4 = $42.12 x hora

$42.12 x 60 horas =$2 527.20

COSTOS Y GASTOS FIJOS: $ 18 675.79

- Visual Studio
- Microsoft Visual Studio 2005 Professional Edition License

  US$774.58 = $ 8156.32
- SQL Server Versión gratuita de http://www.microsoft.com/spain/sql/default.mspx
- Windows server 2003
- Web Server Edition versión económica, con un precio de venta al usuario final de 399 dólares.

Usamos Windows Server 2003 Standard Edition con un precio de 999 dólares, que incluye acceso para cinco usuarios o dispositivos (CAL).

USD$999= $10 519.47

De acuerdo con lo del principio (costos del software) , como este es un software hecho a la medida bastaría con sumar las cantidades de los costos y gastos variables por unidad producida con la de los costos y gasto fijos pero si quisiéramos hacer un producto genérico tendríamos que:

El punto de equilibrio se puede calcular tanto para unidades como para valores en dinero. Algebraicamente el punto de equilibrio para unidades se calcula así:

$$PEunidades = \frac{CF}{PVq - CVq}$$

Donde: CF = costos fijos; PVq = precio de venta unitario; CVq = costo variable unitario

O también se puede calcular para ventas de la siguiente manera.....

$$PEventas = \frac{CF}{1 - \frac{CVT}{VT}}$$

¿Cuántas unidades se deberán producir y vender para no arrojar pérdidas operacionales?¿Cuál es el punto de equilibrio?

Proponemos un precio de venta de $25 000.00

PEunidades = $\frac{18\,675.79}{(25\,000 - 22\,270.95)}$ =6.8433 unidades debemos vender al menos para recuperar lo invertido de ahí en adelante se consideran ganancias.

CAPITULO 6
DISEÑO E IMPLEMENTACIÓN

6.1 Diseño

Hasta este momento es claro que lo que se requiere para cubrir la necesidad de diferentes usuarios para tener acceso a la información del departamento de horarios es una aplicación web, así mismo el diseño e implementación de dicha aplicación debe resolver las deficiencias del actual modelo de implementación.

Debido a que a que la esencia fundamental y materia de este proyecto es la información de manipular, y mostrar información, se debe encontrar y comenzar por la unidad mínima manipulable, en este caso corresponde al tipo de datos de un registro. Puede considerarse al tipo de dato como la unidad mínima, aunque por sí solo es incapaz de aportar mayor información concreta, ya que este valor mínimo es parte de un grupo de datos (campos) que en conjunto representan un registro y el cual es capaz de aportar información concreta. Por este motivo, comenzaremos por el diseño de los registros.

Un registro representa una entidad del universo que se pretende representar, en otras palabras un registro corresponderá a la información de un profesor, academia, materia, carrera, ó alumno, y a su vez este registro posee campos que representas los atributos de la entidad a representar, en nuestro caso la entidad profesor posee; nombre, edad, domicilio, número telefónico, etc.

Como hemos identificado, la entidad principal es la de profesor, el paso siguiente es identificar todas aquellas entidades que existen en el dominio del departamento de horarios.

El segundo paso es identificar el resto de las entidades que existen en el dominio de información del departamento de horarios.

El resto de las entidades se obtiene realizando un análisis al tipo de actividad que ese realiza, específicamente aquellos trámites que se llevan a cabo y hacia quien están dirigidos.

El resultado del análisis junto con una revisión a la actual base de datos del departamento, permite ver claramente que el resto de las entidades son:

- ✓ Materia
- ✓ Academia
- ✓ Carrera
- ✓ Actividad
- ✓ Horario de clases
- ✓ Alumno

De la base de datos puede determinarse cómo y cuales relaciones existen entre las distintas entidades, dichas relaciones permiten construir el modelo relacional para la base de datos. El modelo relacional es un diagrama que permite apreciar cómo y en qué forma se relacionan las entidades de una base de datos. El modelo relacional, también permite ubicar los atributos de cada entidad, dichos atributos, como ya se mencionó son las características que identifican a cada registro dentro de una tabla, hablando en términos de la estructura lógica de la base de datos.

El resultado del modelo relacional obtenido de la base de datos y original y de las actividades del departamento, (ver figura 6.1).

Las relaciones obtenidas generan nuevas tablas, que son el producto de relacionar a las que comparten información de las entidades, debido a esta característica son llamadas bases de datos relacionales.

Una vez obtenido el modelo relacional, por consecuencia definido por las tablas, el siguiente paso es asignar un tipo de dato especifico que cumpla con el dominio para cada campo en la tabla, esto se lleva a cabo mediante la elaboración de los diccionarios de datos, que son convenios que permiten definir qué tipo de dato (entero, carácter, fecha, etc.) y que dominio (0-9, a-z, A-Z, 1900-2050) puede tener un campo.

El diccionario de datos básico obtenido para los campos es el siguiente:

Diccionario de datos del departamento de horarios ESIME Zacatenco												
Número	Nombre del campo	Descripción	Tipo de dato	Tipo de variable	Longitud	Dominio	Llave	Llave alterna	Valor nulo permitido	Observaciones	Tipo tentativo de variable para codificación	Tentativa de codificación
1	Clave	Clave del profesor	Texto	Alfanumérico	5	A-Z, a-z, 0-9	Si	Si	No		String	sProfessorKey
2	Nombre	Nombre del profesor	Texto	Caracteres	64	A-Z, a-z	No	Si	No		String	sProfessorName
3	RFC	RFC del profesor	Texto	Alfanumérico	64	A-Z, a-z, 0-9	No	No	Si		String	sRFC
4	Fecha de nacimiento	Fecha de nacimiento del profesor	Fecha	Formato de fecha/caracteres		1920-2020	No	No	Si		DateTime/string	sBirth
5	Academia	Academia a la que el profesor está adscrito	Texto	Caracteres	20	A-Z, a-z	No	Si	No	Un profesor puede estar adscrito a varias academias	String	sAcademy
6	Tipo	Tipo de profesor	Texto	Caracteres	64	A-Z, a-z	No	Si	No		String	sType
7	Sit		Text	Caracteres	64	A-Z, a-z	No	Si	No		String	sProfessorSt

			o									atus
8	Turno	Horario de trabajo del profesor	Texto	Caracteres	20	A-Z, a-z	No	Si	No		String	sScheldule
9	Cubículo	Cubículo donde se localiza al profesor	Texto	Alfanumérico	12	A-Z, a-z, 0-9	No	No	Si		String	sCubicle
10	Teléfono	Teléfono del profesor	Texto	Numérico	15	0-9	No	No	Si		String, Int64, Long	sPhoneOfProfessor
11	Horario de clases	Horario de clases lunes a viernes	Texto	Caracteres	15	A-Z, a-z, 0-9	No	No	Si		String	sHour
12	Materia	Nombre de la materia	Texto	Alfanumérico	30	A-Z, a-z, 0-9	No	No	No		String	sSubjectName
13	Clave materia	Clave de la materia	Texto	Alfanumérico	15	A-Z, a-z, 0-9	Si	No	No		String	sSubjectKey

En este diccionario está constituido por las siguientes columnas, en primer lugar el nombre del campo, la descripción (que significa), tipo de dato (representación computacional), longitud (número de dígitos permitidos), dominio, llave (especifica si es un campo único para cada registro), llave alterna (un segundo campo único para cada registro), valor nulo permitido (especifica si puede quedar vacio).

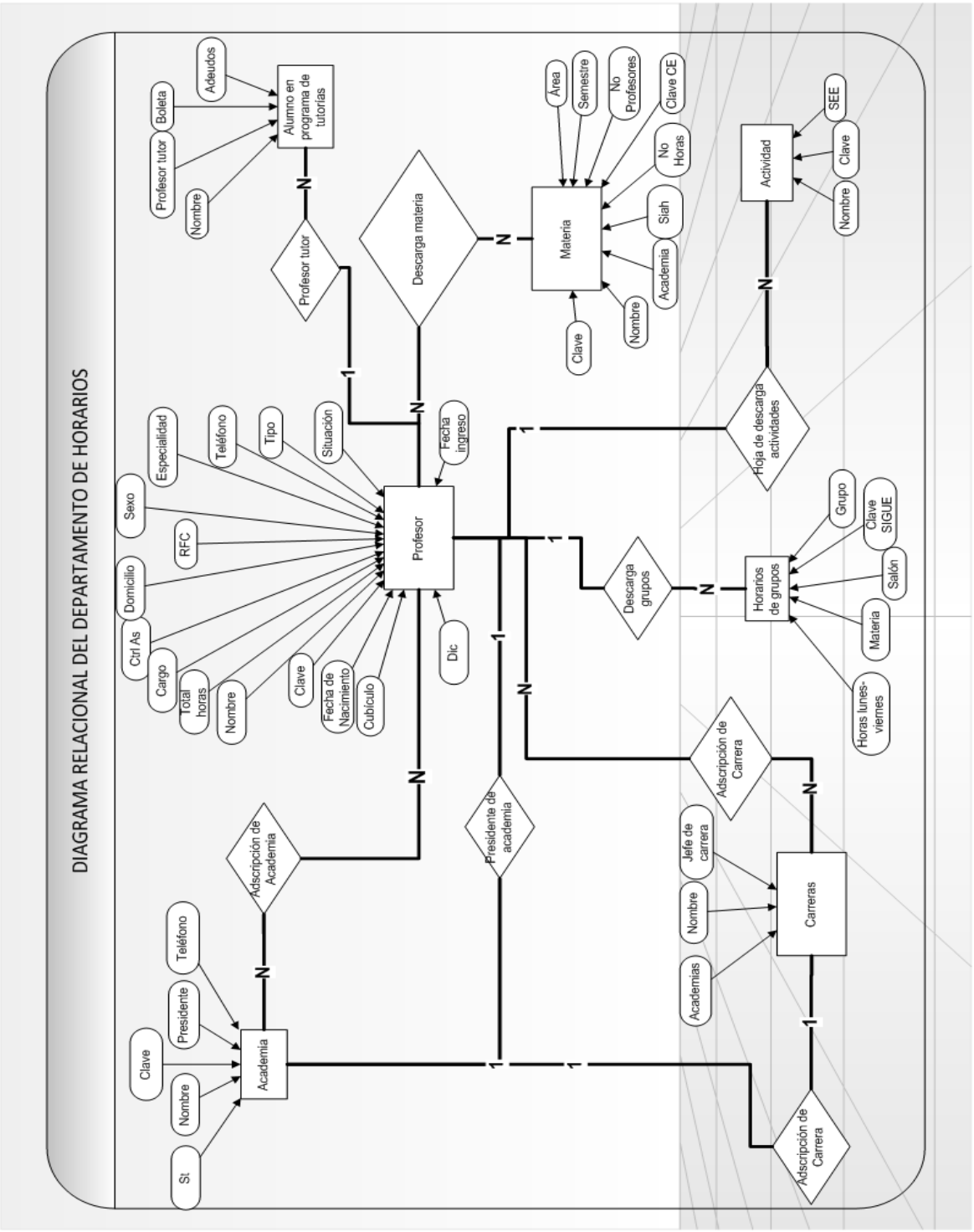


Fig. 6.1 Modelo relacional de las actividades del departamento de horarios.

6.2 Diseño de la aplicación web

En esta sección se comienza por diseñar la interfaz del usuario, es decir cómo serán las pantallas de usuario y la secuencias de navegación de las páginas que conforman el sitio web. El siguiente diagrama muestra las secuencias de navegación, las flechas representa los posibles movimientos de navegación. El diagrama muestra a la página del departamento de horarios, separada en dos secciones. La primera sección está dedicada a usuarios en general, la segunda sección está dedicada a usuarios administrativos, en esta sección se pueden realizar los mismos procedimientos que en la página dedicada a usuarios en general pero esta permite hacer modificaciones, para poder acceder a esta página es necesario iniciar una sesión de usuario, está sesión permite controlar el acceso y tener un nivel de seguridad con el fin de evitar que cualquier usuario no permitido realice modificaciones a la base de datos del departamento de horarios.

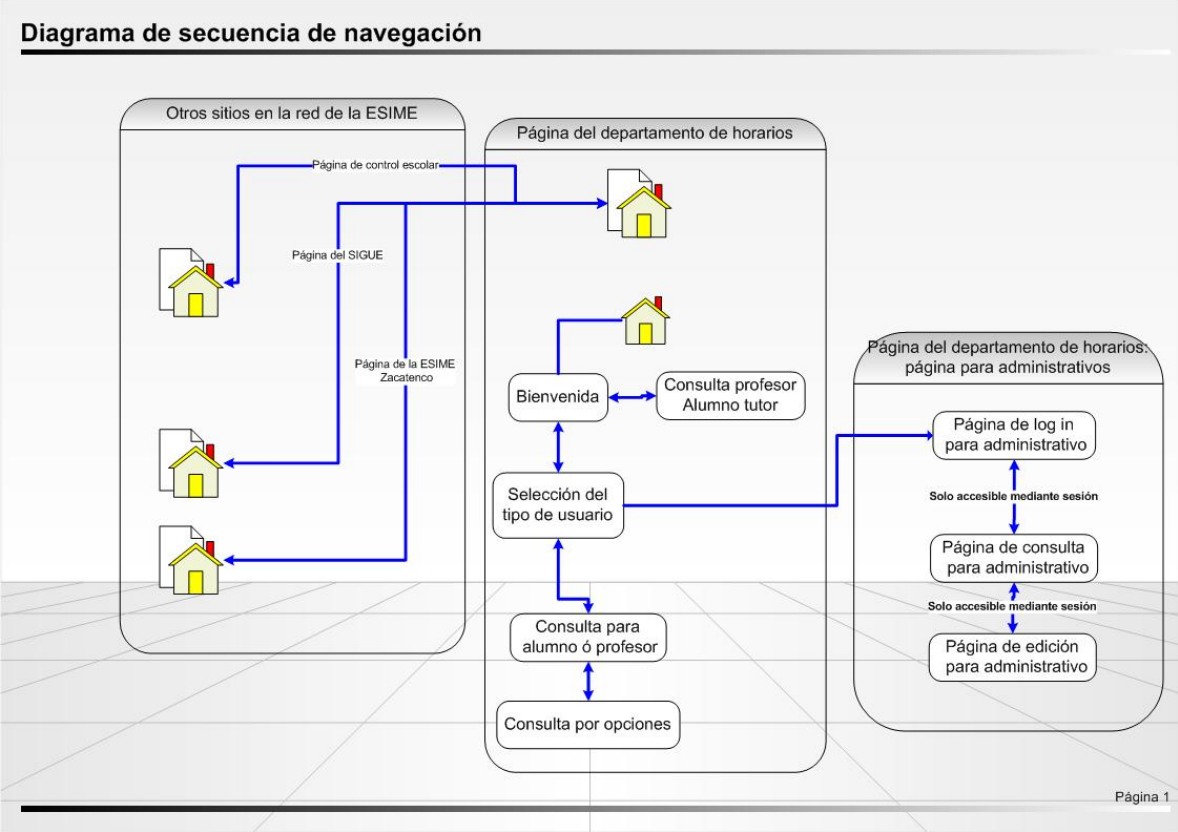


Fig. 6.2 El siguiente diagrama muestra las secuencias de navegación, las flechas representa los posibles movimientos de navegación. El diagrama muestra a la página del departamento de horarios, separada en dos secciones.

En cuanto a la interfaz gráfica, se utilizará una página maestra que permita navegar conservando la posición y formato en todas las páginas, de este modo solo cambiara las partes que sean necesarias, permitiendo al usuario la fácil localización de los botones (ver figura 6.3).

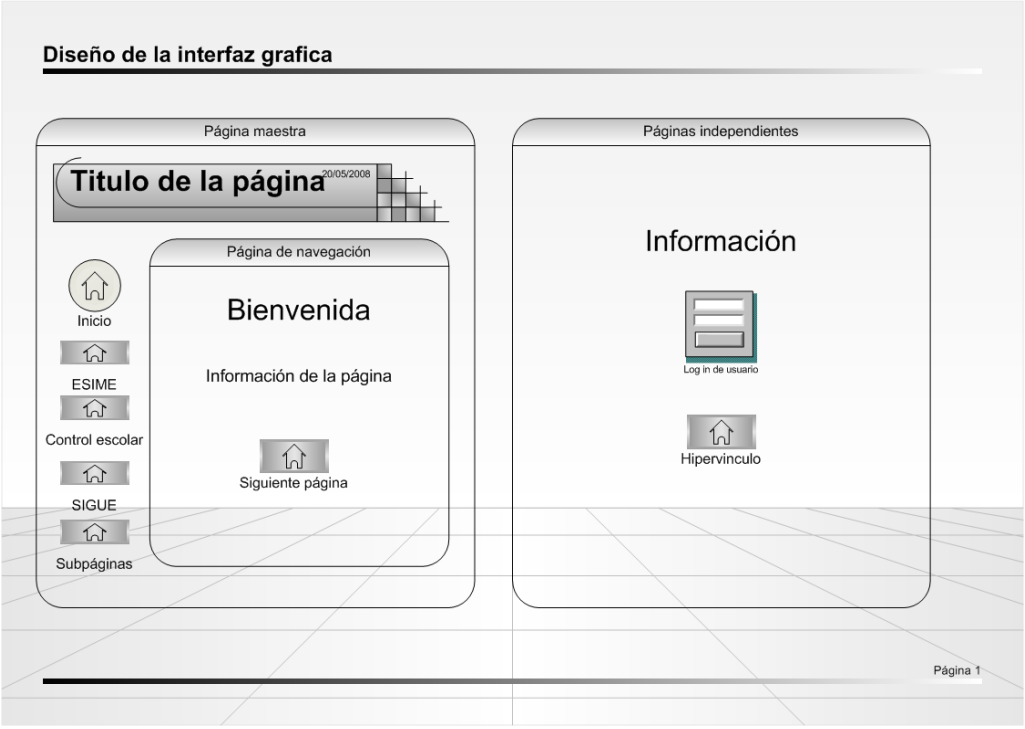


Fig. 6.3 Página maestra que permita navegar conservando la posición y formato en todas las páginas.

Las páginas etiquetadas como páginas independientes son páginas, a las que se puede acceder mediante la página maestra, estas son independientes (que no presentan el mismo contenido que una página maestra) debido a que en ellas se solicita información del usuario, y de esta forma se evita que el usuario navegue a un sitio distinto teniendo una sesión abierta, aunque el diseño contempla que las páginas de log in tenga un tiempo de caducidad en la sesión, un usuario podría dejar abierta una sesión y navegar a otra página pudiendo provocar una situación de riesgo (ver figura 6.3 y 6.4).

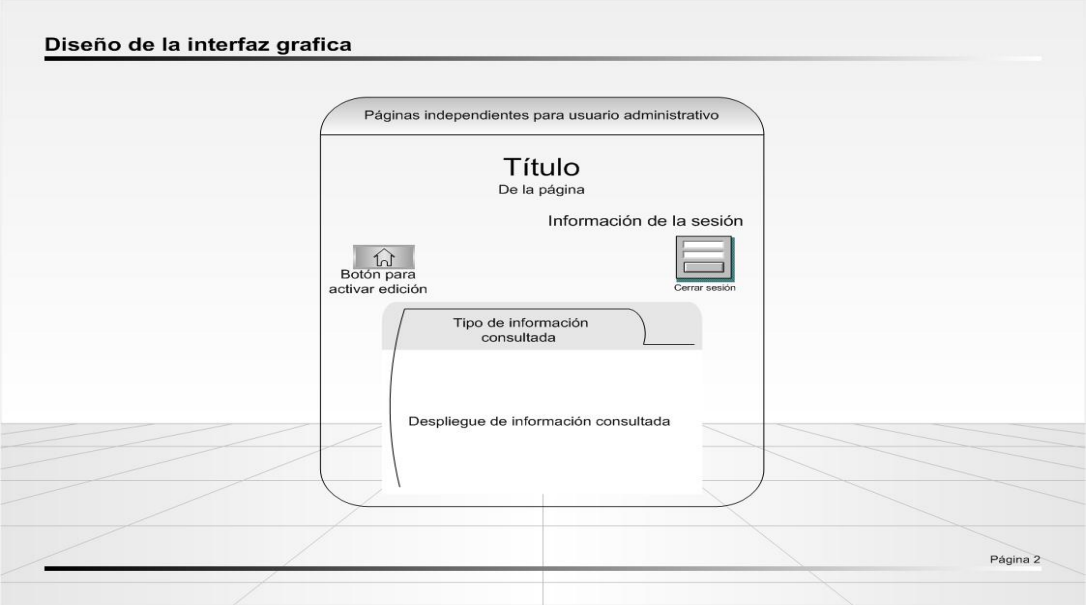


Fig. 6.4 Página etiquetada como página independiente, son páginas, a las que se puede acceder mediante la página maestra, estas son independientes (que no presentan el mismo contenido que una página maestra)

6.3 Diseño del funcionamiento de la aplicación web

En esta parte se tratará con la lógica de la aplicación que es la parte programática es decir el software y la lógica de la aplicación.

El primer aspecto que se debe considerar es que se va a consultar información de un servidor que tiene una base de datos, la información solicitada y enviada por el servidor viajará con la misma forma que en la que está almacenada, a menos que el servidor se le indique cifre o conceda derechos de usuario, restringiendo el acceso, esta vía implicaría que todos los usuarios que quisieran consultar deberían pasar por un log in y deberían estar previamente registrados o en su defecto registrarse. Cómo uno de los objetivos de este documento es el que se acceda a la información de forma rápida y sencilla, la vía de crear usuarios puede desalentar el uso de la aplicación.

Otro punto en la propuesta de este proyecto es la implementación de la aplicación utilizando servicios web, es importante decir que el uso de servicios web agrega a la aplicación inherentemente, flexibilidad y seguridad, ya que cómo se vio en el capítulo dedicado a servicios web, dichos servicios consisten en la serialización de de todos los datos solicitados y enviados, dicha serialización envía los datos a través de un protocolo en ya sea SOAP ó HTTP, no sin antes haber cifrado los datos utilizando funciones hash que

son funciones que permiten cambiar el orden de los datos lo que les hace casi imposibles de interpretar, además de aportarle seguridad al procedimiento.

En cuanto a la flexibilidad, se dice que aporta flexibilidad ya que al ser métodos llamados (invocados) desde un host lejano, permiten la utilización de estos mismos como un recurso para todo aquel desarrollador que tenga acceso a estos mismos, es decir pueden ser utilizados y ejecutados en distintas operaciones, ya sea en otros departamentos o en la transferencia de información entre departamentos.

6.4 Implementación del software

En primer lugar se debe considerar el ¿cómo? y en ¿qué? se implementara la solución, entiéndase solución al conjunto de software que permite resolver el problema.

¿Qué opciones existen en el mercado para poder implementar la solución? y ¿Por qué utilizar una en especial?

En el mercado y específicamente para el propósito de desarrollo web existen principalmente las siguientes opciones de tecnologías:

- PHP (IDE de licencia y uso libre)
- JSP (IDE de licencia bajo costo)
- ASP (IDE de licencia bajo costo y de uso libre)
- ASP.net (IDE de licencia bajo costo y de uso libre)

PHP

PHP es la tecnología Web más extendida en el momento. Nació para trabajar en Linux con servidor Apache, pero hoy en día puede alojarse en casi cualquier tipo de servidor. El código fuente está en abierto por lo que los bugs están muy controlados e inmediatamente solucionados, y además tiene una gran cantidad de módulos prefabricados que ya vienen instalados de fábrica en los servidores, y que no hay más que aprender a utilizar.

Su sintaxis es muy similar a C, quizás algo más simple, y destaca que fue creada para programar páginas Web, aunque también se puede programar en local. Se comunica con bases de datos sin necesidad de usar ODBC. Las últimas versiones ya trabajan con orientación a objetos. Hasta ahora es un lenguaje de script, no compilado

JSP

Los Servlets de Java son muy comunes en aplicaciones Web potentes, como bancos o grandes empresas. Comparte muchas de las ventajas de ASP.NET, sobretodo en cuanto a la programación modular y orientada a objetos, pero sus carencias son también muy destacadas. Sobre todo su bajo enfoque de cara al usuario da mucho trabajo para presentar páginas Web al navegador.

Para muy grandes aplicaciones suele elegirse JSP en lugar de PHP, dado que PHP es un lenguaje de más bajo nivel que JSP y dificulta la modularización y organización por capas de la aplicación.

ASP

ASP es la tecnología pionera en las aplicaciones Web que se ejecutan en el servidor. Desarrollada por Microsoft y optimizada para su ejecución en servidores Windows con tecnología NT bajo IIS, aunque también hay opciones para Windows 98 con el Personal Web Server y para Linux con Chilisoft.

Al ser una tecnología propietaria, no tiene la enorme cantidad de módulos extra que sí tiene PHP, aunque abriendo objetos COM trabaja fácilmente con archivos DLL.

El hecho de que habitualmente los servidores de hospedaje de Internet sean más caros, han hecho que ASP deje de ser la tecnología para página Web dinámicas orientadas a servidor más utilizada, tal y como lo llegó a ser en sus inicios.

Usa Visual Basic Script, lo que para los desarrolladores Visual Basic era una ventaja pues no tenían que aprender otro lenguaje, para otros es una desventaja, pues consideran que no es más que un parche Web de un lenguaje ya existente, y que no fue creado expresamente para los servidores Web, como sí pasa con PHP. También soporta el lenguaje JScript (JavaScript de Microsoft). Se comunica con las bases de datos vía ODBC, y la comunicación con SQL Server es óptima.

ASP.NET

ASP.NET rompe totalmente con el pensamiento de script que se tenía hasta el momento. El cambio en la arquitectura es radical. De hecho, lo único que mantiene de ASP es el nombre, el propietario y la evolución de Visual Basic a Visual Basic .NET (VB.NET).

Dado que la Web no se lee secuencialmente sino que se compila, lo primero que llama la atención es el enorme incremento de velocidad de respuesta del servidor. Además, al compilarse, el incremento en **seguridad y fortaleza** es muy grande.

ASP.NET introduce el concepto del **code-behind**, por el que una misma página se compone de dos ficheros: el de la interfaz de usuario y el de código. Con ello se facilita la programación de aplicaciones en múltiples capas, lo que en definitiva se traduce en la total separación entre lo que el usuario ve y lo que la base de datos tiene almacenado. Por tanto, cualquier cambio drástico de especificaciones minimiza los cambios en la aplicación y maximiza la facilidad de mantenimiento.

Asimismo, ASP.NET nos sirve tanto para Webs sencillas como para grandes aplicaciones. No debemos olvidar que la orientación a objetos y la naturaleza compilada permiten que hagamos uso de herramientas de creación de Webs, las más importantes de la familia del Visual Studio, que nos facilitan mucho la tarea de programación. Estas herramientas permiten hacer Webs sencillas y de bajas prestaciones en un tiempo record, así como llevar el mantenimiento de grandes aplicaciones de forma más sencilla.

Resumiendo, tenemos mayor velocidad, mayor potencia, mayor seguridad, mayor facilidad de mantenimiento y herramientas de trabajo. A continuación enumeramos algunas otras que no tienen ASP, PHP o JSP:

- Caché: se puede almacenar en la caché del servidor tanto páginas enteras, como controles personalizados o simples variables. En páginas críticas con mucha carga de base de datos nos es muy útil almacenar datos de la base de datos en la caché, reduciendo enormemente el consumo de recursos.
- Carpetas especializadas, como por ejemplo app_code que compila automáticamente las clases que se alojan en él, o la carpeta app_theme que alojan ficheros que marcan los temas de estilos de la Web.
- Los archivos de configuración Web.config y Machine.config permiten realizar operación de configuración en ficheros que hasta ahora había que realizar en el servidor.

- La adaptación automática del código devuelto a los dispositivos que le acceden. Una misma página puede servirnos para el Internet Explorer, para el Pocket Internet Explorer desde una PDA o para un navegador de un móvil cualquiera.
- La eliminación total de la necesidad de frames con la introducción de las masterpages.
- La extraordinaria compatibilidad con XML y los servicios Web.

- La multitud de controles Web que permiten mucha funcionalidad con poco código. Desde enlace con las bases de datos o enseñar fácilmente todos los datos, hasta simples etiquetas, hiperenlaces o generadores de imágenes.

- Se puede utilizar hasta cuarenta lenguajes distintos para el desarrollo en ASP.NET, aunque en el 95% de las aplicaciones se usa C#, VB.NET o J#.

Otra ventaja más es que la tecnología ASP.net se puede utilizar sin costo alguno y bajo una licencia libre, ya que viene como parte del framework de desarrollo de Visual Studio 2005 que es una IDE de uso libre en su versión Express y la cual puede ser descargada de internet para su uso sin ningún consto, esto nos llevo a elegir ASP.net como la tecnología que se usará en el desarrollo.

Otro elemento necesario para la implementación es el uso de un manejador de base de datos, en este caso hemos seleccionado SQL Server Express, ya que junto con la suite de Visual Studio Express puede ser descargada y utilizada sin costo alguno, además de ser IDEs complementarias.

Por las razones anteriores se escogió la IDE de Visual Studio Express y el manejador SQL Server Express para la implementación y programación de la aplicación.

¿Por qué la versión ASP.NET 2.0?

Las razones de elegir la versión 2.0 en lugar de la ya madura versión 1.1 es una gran mejora en todos los aspectos en general, pero destacamos la comunicación con base de datos, ADO.NET 2.0, C# 2.0 y el incremento de los controles Web.

No en vano la relación de líneas de código entre ASP.NET 1.1 y ASP.NET 2.0 es de diez a uno, ya que muchos de esos controles realizan de forma óptima multitud de tareas que se repiten en todas las aplicaciones. Por ejemplo, enlazar con la base de datos, asignar una clase para programación multicapa de selección, inserción, borrado y modificación,

enseñar esos datos en pantalla, paginarlos y permitir la modificación o borrado de registros lo podemos realizar en tres líneas:

1: <asp: ObjectDataSource ID="ODS1" runat="server" DeleteMethod="BorrarUsuario" InsertMethod="BorrarUsuario" SelectMethod="BorrarUsuario" TypeName="PFC.Admin.Usuarios" UpdateMethod="BorrarUsuario" />
2: <asp: GridView ID="GV1" runat="server" DataSourceID=" ODS1" AutogenerateColumns = “True” AllowPaging="True" />
3: <Columns><asp: CommandField ShowDeleteButton = "True" ShowEditButton = "True" ShowSelectButton = "True" /> </Columns>

Personalizar las columnas, asignar parámetros a procedimientos almacenados, y demás quehaceres procuran unas pocas líneas de código adicionales. No es nuestra preocupación el hacerlo todo con el menor número de líneas de código posible, sino en centrarnos en el código más útil y no en el que se repite aplicación tras aplicación.

6.5 Construcción de la aplicación

En esta sección se procederá a la construcción de la interfaz de usuario, es decir cómo serán finalmente las páginas que el usuario vera en su navegador web.

Con ayuda de la IDE de Visual Studio, creamos una solución vacía donde agregaremos dos proyectos del tipo ASP.net Web Site y otro del tipo ASP.net Web Service, en los cuales colocaremos las páginas web previamente diseñadas y escribiremos el código del servicio web, (ver figuras 6.5 y 6.6).

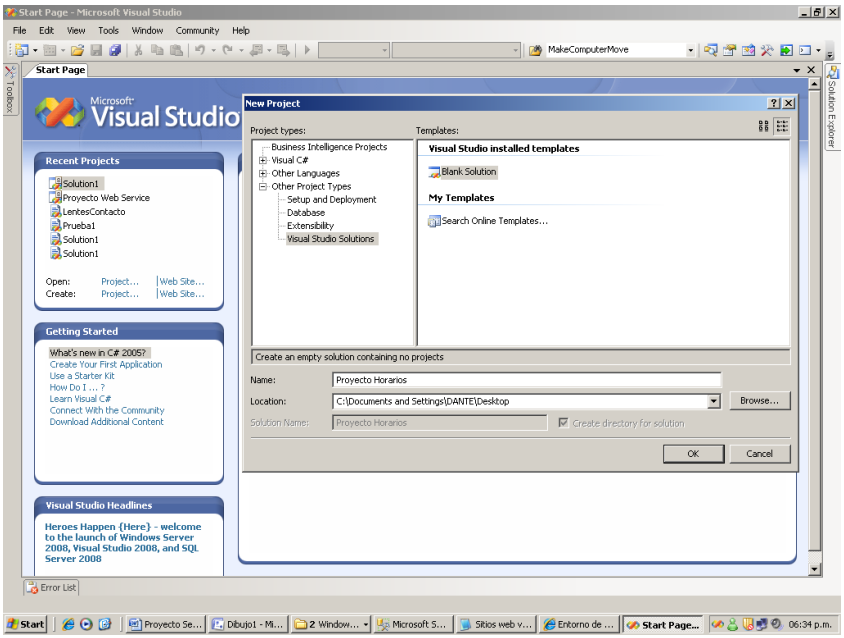


Fig. 6.5 IDE de Visual Studio, creamos una solución vacía donde agregaremos proyecto del tipo ASP.net Web Site.

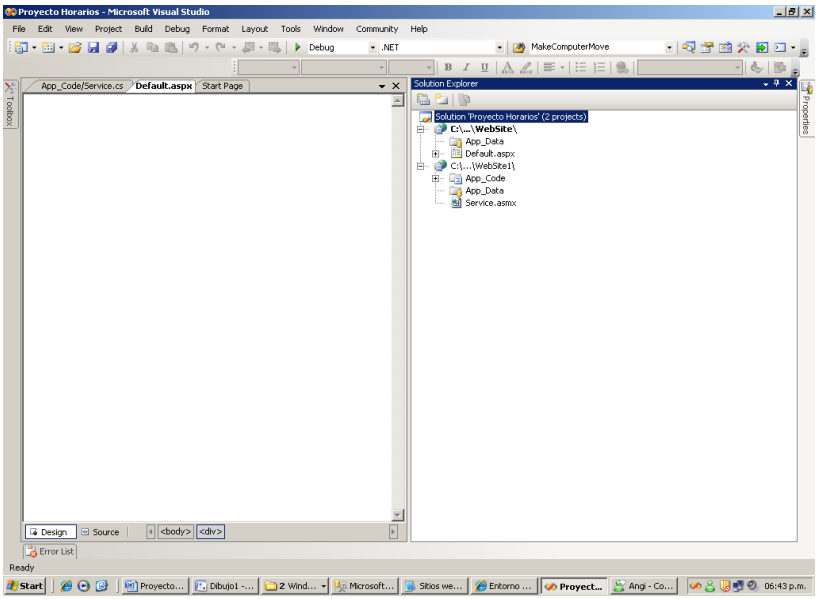


Fig. 6.6 Y otro del tipo ASP.net Web Service.

Utilizando la interfaz grafica del IDE de visual studio creamos el sitio web, utilizando como se mencionó antes una página maestra (master page) y las páginas que se mostraran dentro de página maestra; (ver figura 6.7).

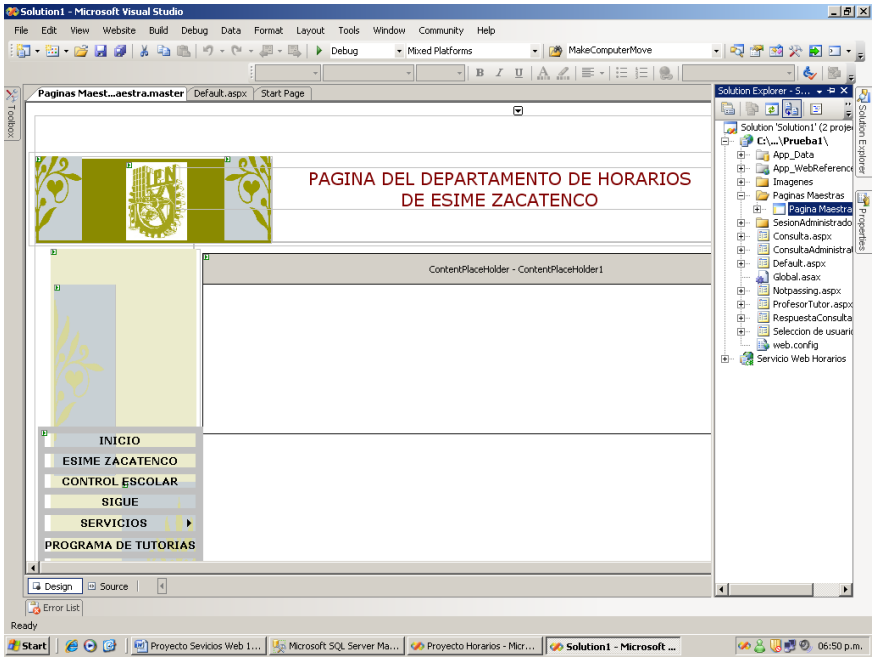


Fig. 6.7 Plantilla de página maestra para la solución.

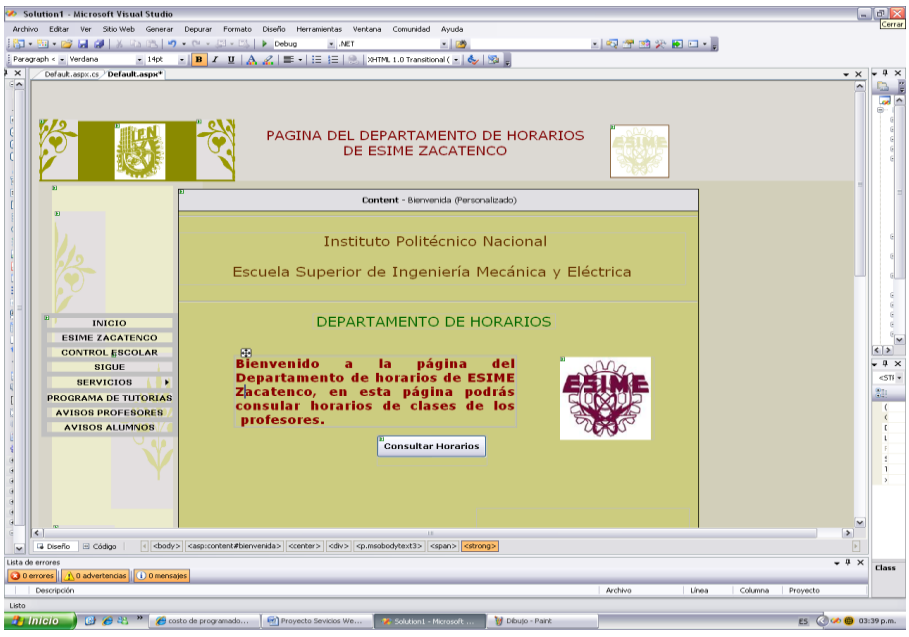


Fig. 6.8 Página independiente default.aspx insertada dentro de la página maestra.

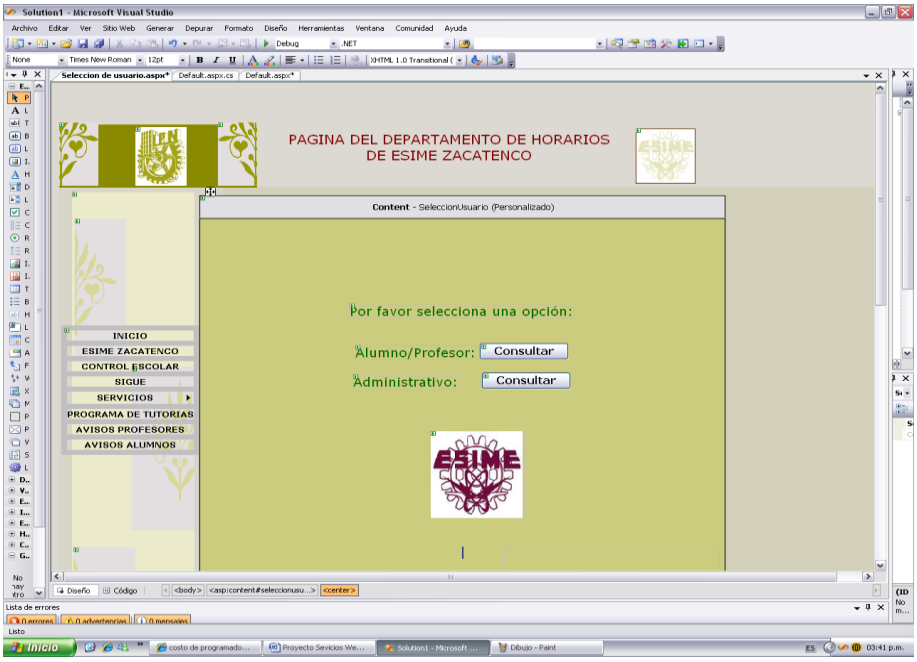


Fig. 6.9 Página independiente (web form: Selección de Usuario.aspx) insertada dentro de la página maestra.

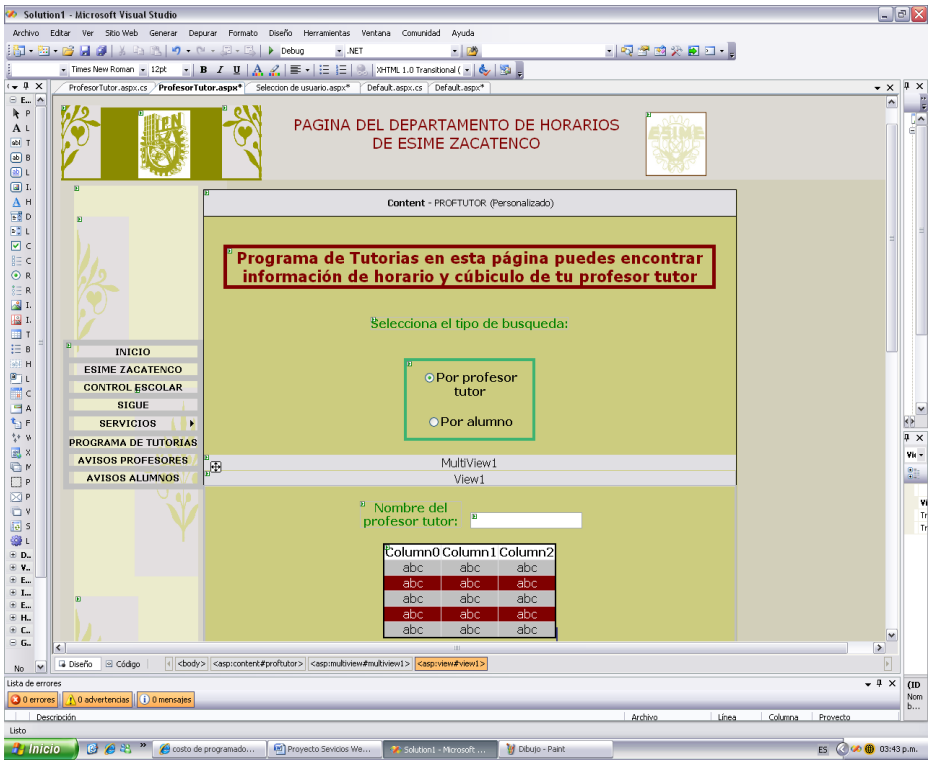


Fig. 6.10 Página independiente (web form : ProfesorTutor.aspx) insertada dentro de la página maestra.

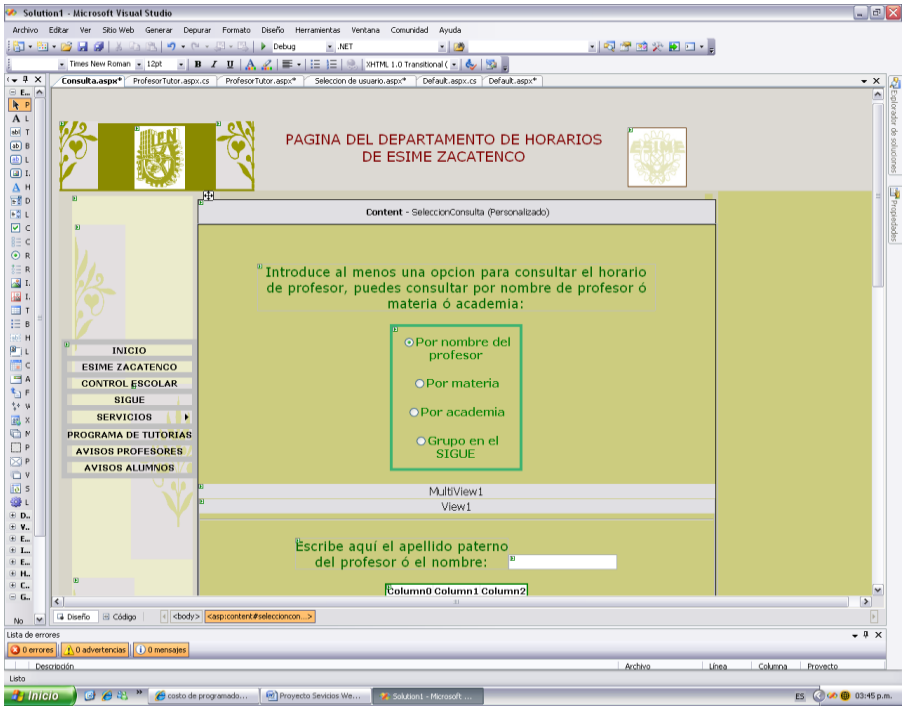


Fig. 6.11 Página independiente (web form: Consulta.aspx) insertada dentro de la página maestra.

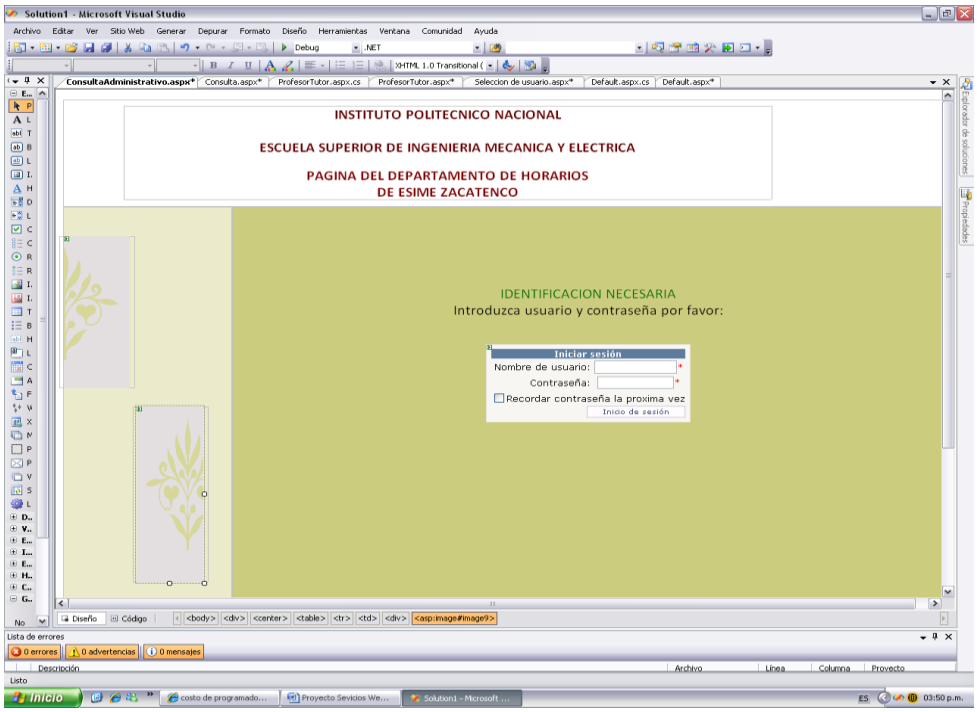


Fig. 6.12 Página independiente (web form: ConsultaAdministrativo.aspx) insertada dentro de la página maestra.

Después se construyen las páginas que no utilizan paginas maestras ver figura 6.12, se agregan los controles ASP, HTML y de datos que se requieren para poder visualizar datos dese un acceso remoto a datos, se les da formato y estilo para hacerlos más amigables. En este caso se utilizarán los siguientes controles:

- Label
- Radio button list
- Grid View
- Menu
- Div
- Text Box
- Multiview
- View
- Long in
- Login name

El siguiente paso, es habilitar los eventos de los controles, esto se puede hacer mediante la interfaz gráfica, del IDE o haciéndolo mediante código, por ejemplo el evento postback de páginas que tienen controles “radio button list” para que en el momento que el cliente (el usuario en su navegador) efectué un evento click sea enviada una respuesta al servidor y este le envié un nuevo código html para visualizar en el explorador. Una vez habilitados los eventos necesarios de los controles se escribe el código para que funcione con la lógica planteada en el diseño de navegación, exceptuando los controles que invocaran a los servicios web.

Para probar la lógica de la navegación ejecutamos la aplicación y se prueba manualmente la correcta navegación; (ver figura 6.13 - 6.15).

Fig. 6.13 Bienvenida a la página de horarios, dar click en Consultar Horarios.

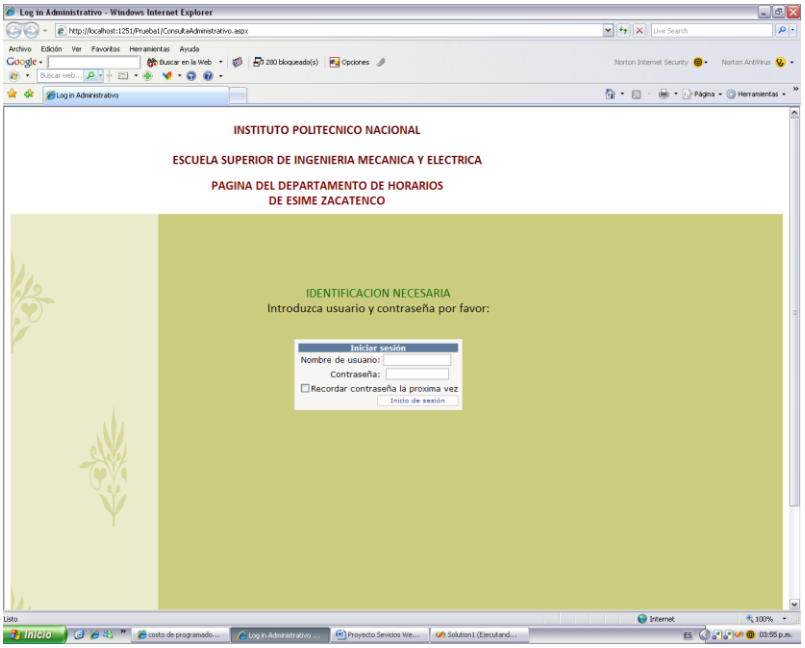


Fig. 6.14 Sección de la validación de administrador.

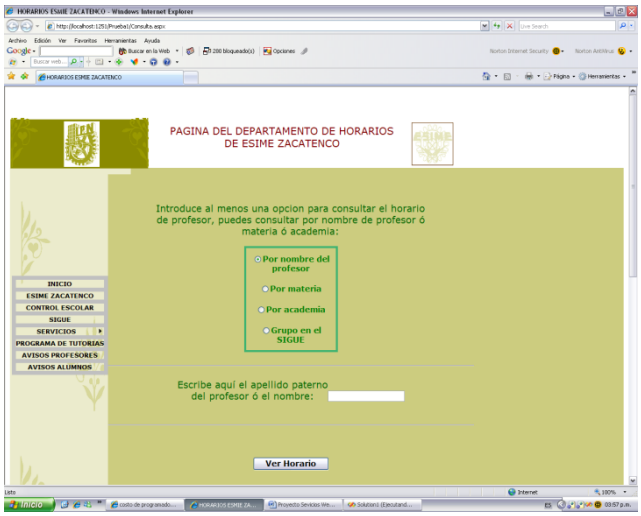


Fig.6.15 Validando RadioButtons en la sección de búsqueda por profesor o materia o academia o Grupo en el SIGUE.

Fig. 6.16 Validando RadioButtons en la sección del programa de tutorías por profesor o por alumno.

El siguiente paso es crear el servicio web en el proyecto web Service que se creó previamente. El código del web Service se coloca el en archivo asmx.cs (véase Anexo 1) que es el archivo que contiene el código de clases y métodos del servicio web.

WEB FORMS (aspx):

Nombre de la forma:	Default.aspx
Servicios web consumidos:	----------
Descripción:	Página principal de bienvenida a la página del departamento de horarios de ESIME Zacatenco.

Nombre de la forma:	Selección de Usuario.aspx
Servicios web consumidos:	----------
Descripción:	Presenta la opción de elegir la consulta en modo alumno/profesor o administrativo.

Nombre de la forma:	Consulta.aspx
Servicios web consumidos:	Service.asmx
Descripción:	Presenta la consulta de horario del profesor, puede consultar por nombre de profesor ó materia ó academia ó grupo en el SIGUE.

Nombre de la forma:	ConsultaAdministrativo.aspx
Servicios web consumidos:	----------
Descripción:	Presentar la identificación necesaria para consultar como administrador.

Nombre de la forma:	ProfesorTutor.aspx
Servicios web consumidos:	Service.asmx
Descripción:	Presenta la consulta de horario y cubículo del profesor tutor para el programa de tutorías.

SERVICIO WEB (asmx):

Servicio web:	Service.asmx
Métodos:	HelloWorld()
Parámetros que recibe:	----------
Regresa:	Cadena de caracteres.
Descripción:	Método por defecto que permite hacer una prueba de inicio, este método no efectúa ninguna operación importante para la aplicación.

Servicio web:	Service.asmx
Métodos:	ConsultarPorNombreDeProfesor()
Parámetros que recibe:	Cadena de búsqueda, cadena de conexión , nombre de tabla
Regresa:	DataSet
Descripción:	Hace una consulta al servidor SQL Server express, utilizando una cadena de conexión y un parámetro de búsqueda.

Servicio web:	Service.asmx
Métodos:	ConsultaPorMateria()
Parámetros que recibe:	Cadena de búsqueda, cadena de conexión , nombre de tabla
Regresa:	DataSet
Descripción:	Hace una consulta al servidor SQL Server express, utilizando una cadena de conexión y un parámetro de búsqueda.

Servicio web:	Service.asmx
Métodos:	ConsultarPorAcademia()
Parámetros que recibe:	Cadena de búsqueda, cadena de conexión , nombre de tabla
Regresa:	DataSet
Descripción:	Hace una consulta al servidor SQL Server express, utilizando una cadena de conexión y un parámetro de búsqueda.

Servicio web:	Service.asmx
Métodos:	ConsultaDeProfesorTutor()
Parámetros que recibe:	Cadena de búsqueda, cadena de conexión , nombre de tabla
Regresa:	DataSet
Descripción:	Hace una consulta al servidor SQL Server express, utilizando una cadena de conexión y un parámetro de búsqueda.

Servicio web:	Service.asmx
Métodos:	ConsultarHorarioGrupoSigue()
Parámetros que recibe:	Cadena de búsqueda, cadena de conexión , nombre de tabla
Regresa:	DataSet
Descripción:	Hace una consulta al servidor SQL Server express, utilizando una cadena de conexión y un parámetro de búsqueda.

Servicio web:	Service.asmx
Métodos:	ConsultarPorAlumnoProfesorTutor()
Parámetros que recibe:	Cadena de búsqueda, cadena de conexión , nombre de tabla
Regresa:	DataSet
Descripción:	Hace una consulta al servidor SQL Server express, utilizando una cadena de conexión y un parámetro de búsqueda.

Servicio web:	Service.asmx
Métodos:	InsertarNombreRegistroPrueba
Parámetros que recibe:	Cadena de nuevo valor y una cadena de conexión.
Regresa:	Un valor booleano.
Descripción:	Este método inserta nuevos valores.

6.6 Explicación del código

Los métodos que tienen la palabra “Consultar”, en el nombre de método hacen una consulta (query) al servidor SQL Server express, utilizando una cadena de conexión (una direccion y parametros para la conexión a un servidor) y un parametro de búsqueda. Utiliza los objetos SqlConnection, SqlDataAdapter y DataSet del framework el metodo envia un comando de SQL al servidor mediante el objeto SqlDataAdapter y regresa un objeto del tipo DataSet, el cual encapsula los datos seleccionados por la consulta, los cuales serán posteriormente procesados pro el control de la página “Grid View”. El resto de los metodos solo cambian el comando de SQL por SELECT, UPDATE Y DELETE, de esta manera el proceso de consulta es invisibe para el código de la página.

6.7 Integración del servicio web y pruebas

El último paso es; el de indicarle al proyecto de web site que apunte a la referencia del web Service cada vez que se le invoque y que interacute con el los eventos de la página para mostrar los datos en el control “Grid View”.

El procedimiento es el siguiente: con ayuda del IDE, agregar una refencia al web site hacia el web Service ver figura 6.17.

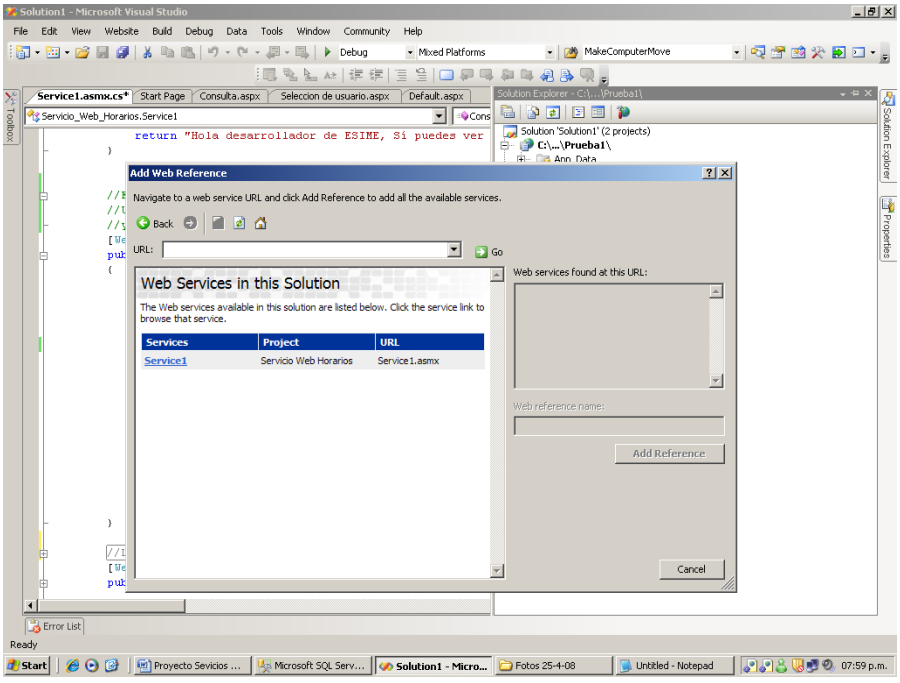


Fig. 6.17 Agregando referencia web Service al servicio web para su invocación.

El siguiente paso es escribir el código para que los controles Button y “Grid View” usen el objeto devuelto por la invocación del web Service, primeramente se crea una instancia y se invoca el servicio y el método especifico de acuerdo a la lógica de la página. El código para los controles mencionados se encuentra en el Anexo 2.

Una vez escrito el código, se compila el proyecto y se ejecuta para realizar la prueba. La prueba para verificar el código de invocación del web Service, serán las siguientes:

- Hacer una consulta por Profesor
- Hacer una consulta por Materia

Para la primera prueba se introdujo como parámetro a buscar el nombre de profesor, “HARO”, este nombre fue seleccionado de la base de datos ya que solo existe un registro que contiene esta subcadena, por lo tanto la aplicación deberá devolver solo una coincidencia.

El resultado de la prueba se puede ver en la figura 6.18.

Para la segunda prueba se introdujo como parámetro a buscar el nombre de materia, “Agentes inteligente”, el resultado debe devolver aquellas coincidencias correspondientes a profesores de noveno semestre.

El resultado de la prueba se puede ver en la figura 6.19.

Efectivamente se ha encontrado la aplicación, cumplió satisfactoriamente con la prueba, y es capaz de hacer lo que se había planeado para que se llevara a cabo.

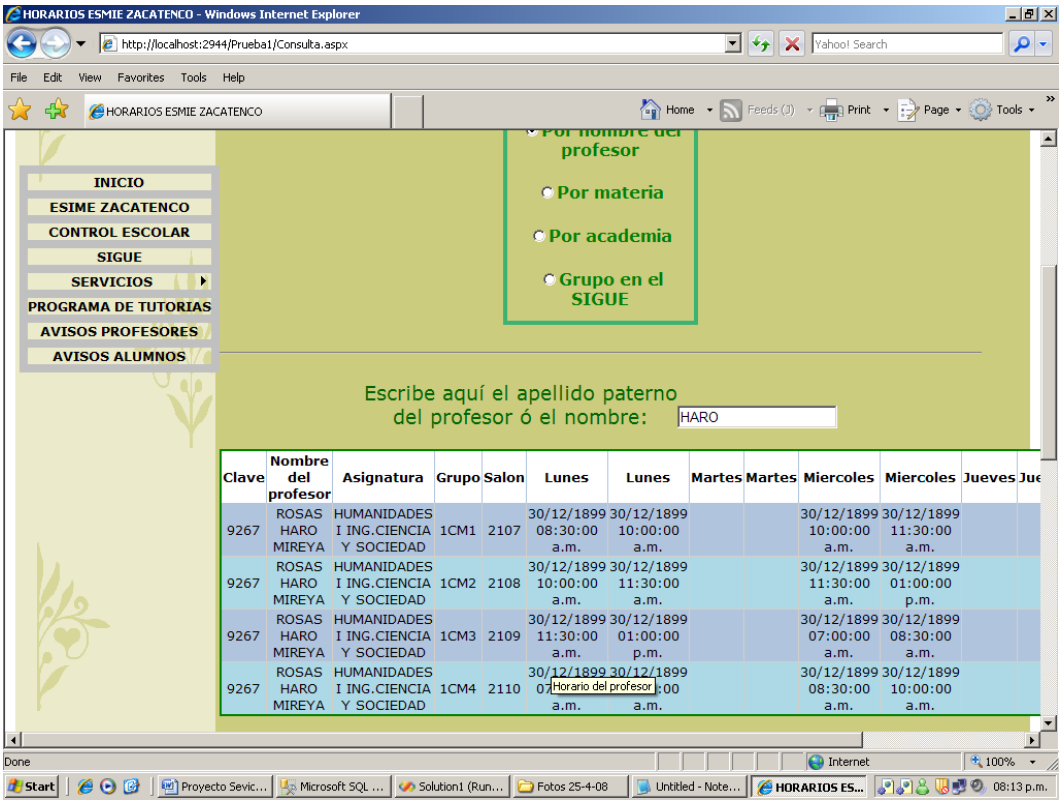


Fig. 6.18 Búsqueda con nombre de profesor, "HARO".

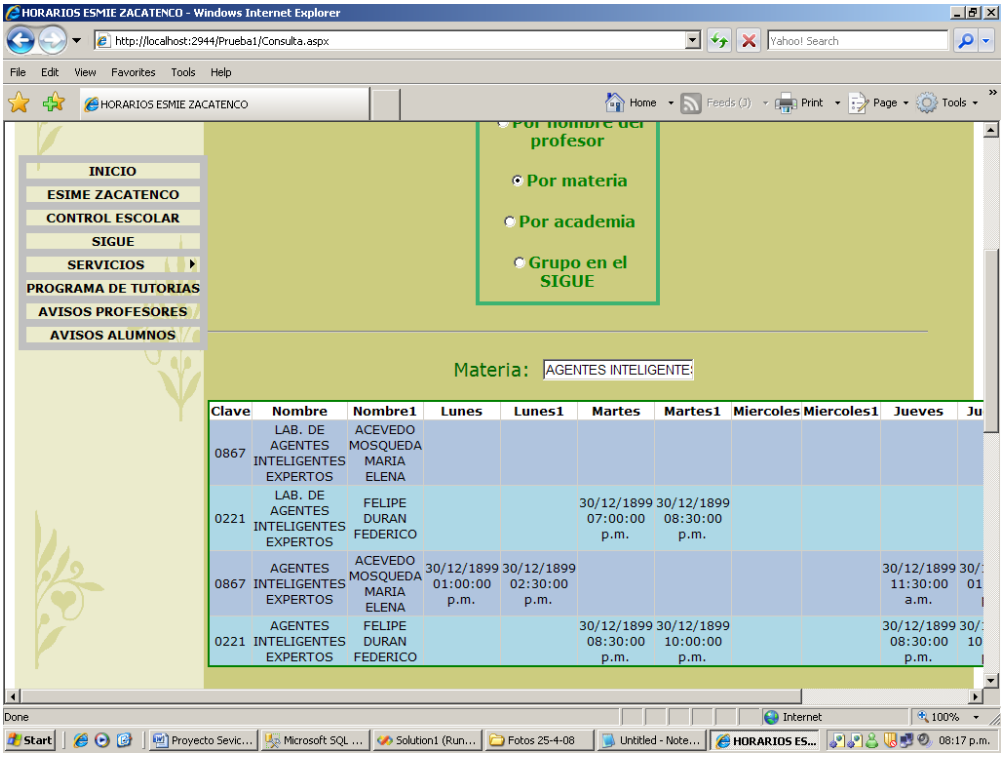


Fig. 6.20 Búsqueda con nombre de materia, "Agentes Inteligentes".

CAPITULO 7
CONCLUSIONES

El uso de un directorio académico dentro de una institución escolar de nivel superior es fundamental en el aspecto de la localización de profesores por cualquier tipo de necesidad por parte de los alumnos. El uso del servicio web es una buena manera de resolver este aspecto en una administración escolar ya que maneja varios de los problemas planteados, como fue el caso de consultas de información docente a la comunidad estudiantil, esto por medio de tecnologías que son la tendencia en un futuro para trabajar sobre estas en cualquier ámbito laboral.

Conocimos que Visual Studio es una buena manera de crear esta solución ya que permite desarrollar aplicaciones y servicios web de cualquier tipo de manera eficiente y práctica, con sus herramientas dirigidas a este tipo de necesidades.

El haber resuelto el problema de un directorio usando un servicio Web nos dará la oportunidad a futuro si se requiere el poder dejar esta aplicación disponible hacia modificaciones que la mejoren, así como la utilización de esté, por parte de cualquier otra aplicación o servicio de algún departamento de la institución ya que aportara la utilidad y el uso necesario para la comunidad de ESIME Zacatenco.

7.1 Resultados Finales

Mejor desempeño el próximo semestre entre alumnos y profesores en el sentido de la localización ya sea para asesorías, tutorías, u otro tipo de necesidad escolar

Con el desarrollo de este servicio web obtuvimos como resultado el poder encontrar en un mismo dominio la información necesaria para un alumno como para un profesor o personal administrativo el cual se ha implementado para lograr una mejor funcionalidad lo que anteriormente se hacia en dos páginas web, ahora en una sola.

Conociendo las características y propiedades del servicio web esperamos que nuestro proyecto sea consumido o utilizado por otro tipo de aplicaciones en otros departamentos internos de ESIME Zacatenco.

Fuentes Electrónicas:

[1] (fuente: http://www.desarrolloweb.com/articulos/1883.php)

[2] http://www.manycomics.com/ingenieria-del-software/ciclo-vida-software/

[3] Extreme Programming and Open Source Software

http://www.advogato.org/article/202.html

Fuentes Bibliográficas:

- Tabor R., (2002), ***Servicios Web XML De Microsoft .NET,*** (2da. Edición). España: Prentice Hall

- Deitel H., Deitel P., ***Cómo programar C#,*** (2ª edición), México D.F.: Prentice Hall

- Pressman R.(2001), ***Software engineering: a practitioner's approach***, New York, NY, McGraw-Hill

- Balter A.(2006), **Teach *Yourself Microsoft® SQL Server™ 2005 Express in 24 Hours,*** E.U.A, Sams Publishing

- Darie C., Ruvalcaba Z. (2006), ***Build Your Own ASP.NET 2.0 Web Site Using C# & VB*** (segunda edición), E.U.A, Site Point

- [4] Beck K., 1999, ***"Extreme Programming Explained: Embrace Change"***,(1a edición), Addison-Wesley Pub Co

- [5] Gonzalo León S., 1996 ***INGENIERÍA DE SISTEMAS DE SOFTWARE***, Madrid: Publicaciones de Ingeniería de Sistemas.

- Sinay D., ***Web Services con C#,*** (primera edición) Buenos Aires, Argentina, Users Code

Anexo 1

```
using System;
using System.Data;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.ComponentModel;
using System.Data.SqlClient;
using System.Xml.Serialization;

namespace Servicio_Web_Horarios //espacio de nombres del servicio web
{
    /// <summary>
    /// Summary description for Service1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")] //Clase para describir la funcinalidad
del web service
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [ToolboxItem(false)]
    public class Service1 : System.Web.Services.WebService //Definición de la clase Service1
    {

        [WebMethod]
        public string HelloWorld() //Metodo por defecto que permite hacer una prueba de
inicio, este metodo no efectua ninguna operación importante para la aplicación
        {
            return "Hola desarrollador de ESIME, Sí puedes ver este mensaje singnifica que
este web services funciona bien.";
        }


        //El siguiente metodo hace una consulta al servidor SQL Server express, utilizando
una cadena de conexión y un parametro de busqueda
        //Utiliza los objetos SqlConnection,SqlDataAdapter y DataSet del framework. Enevia
un comando de SQL al servidor mediante el objeto SqlDataAdapter
        //y regresa un objeto del tipo DataSet, el cual encapsula los datos seleccionados
por la consulta
        [WebMethod]
        public DataSet ConsultarPorNombreDeProfesor(string sNombre, string
sCadenaDeConexion,string sNombreDeLaTabla)
        {

            string QRY = "SELECT HorSig2.MtoTit AS [Clave], Profesores.Nombre AS [Nombre del
profesor],Materias.Nombre AS [Asignatura],HorSig2.Sigue AS [Grupo], HorSig2.Salon,
HorSig2.LUI AS [Lunes], HorSig2.LUF AS [Lunes ] ,HorSig2.MAI AS [Martes],HorSig2.MAF AS
[Martes ], HorSig2.MII AS [Miercoles], HorSig2.MIF AS [Miercoles ],HorSig2.JUI AS [Jueves],
HorSig2.JUF AS [Jueves ], HorSig2.VII AS [Viernes],HorSig2.VIF AS [Viernes ]From
((Profesores INNER JOIN HorSig2 ON Profesores.Clave = HorSig2.MtoTit) INNER JOIN Materias ON
Materias.Materia = HorSig2.Materia) WHERE Profesores.Nombre LIKE '%" +  sNombre + "%';";


             oConexionAlServidor = new SqlConnection(sCadenaDeConexion);

            oConexionAlServidor.Open();

            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);

            DataSet oDatos = new DataSet();

            oAdaptadorDeDatos.Fill(oDatos, sNombreDeLaTabla);

            return oDatos;

        }
```

```
        //Los siguientes metodos realizan la misma tarea que el metodo
"ConsultarPorNombreDeProfesor", con la diferencia que el query al servidor se hace
        //para diferentes tablas.
        [WebMethod]
        public DataSet ConsultarPorMateria(string sNombreDeLaMateria, string
sCadenaDeConexion, string sNombreDeLaTabla)
        {
            //string sCadenaDeConexionAlServidor = sCadenaDeConexion;

            string QRY = "";


            QRY = "SELECT HorSig2.MtoTit AS [Clave], Materias.Nombre, Profesores.Nombre,
HorSig2.LUI AS [Lunes], HorSig2.LUF AS [Lunes] ,HorSig2.MAI AS [Martes],HorSig2.MAF AS
[Martes], HorSig2.MII AS [Miercoles], HorSig2.MIF AS [Miercoles],HorSig2.JUI AS [Jueves],
HorSig2.JUF AS [Jueves], HorSig2.VII AS [Viernes],HorSig2.VIF AS [Viernes]FROM (Materias
INNER JOIN HorSig2 ON Materias.Materia = HorSig2.Materia) INNER JOIN Profesores ON
Profesores.Clave = HorSig2.MtoTit WHERE Materias.Nombre LIKE '%" + sNombreDeLaMateria +
"%';";

            SqlConnection oConexionAlServidor = new SqlConnection(sCadenaDeConexion);

            oConexionAlServidor.Open();

            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);

            DataSet oDatos = new DataSet();

            oAdaptadorDeDatos.Fill(oDatos, sNombreDeLaTabla);

            return oDatos;
        }

        [WebMethod]
        public DataSet ConsultarPorAcademia(string sNombreAcademia, string
sCadenaDeConexion, string sNombreDeLaTabla)
        {

            string QRY = "";

            QRY = "SELECT HorSig2.MtoTit AS [Clave],Academias.Academia, Materias.Nombre AS
[Asignatura], Profesores.Nombre AS [Profesor], HorSig2.Sigue AS [Grupo],  HorSig2.LUI AS
[Lunes], HorSig2.LUF AS [Lunes ] ,HorSig2.MAI AS [Martes],HorSig2.MAF AS [Martes ],
HorSig2.MII AS [Miercoles], HorSig2.MIF AS [Miercoles ],HorSig2.JUI AS [Jueves], HorSig2.JUF
AS [Jueves ], HorSig2.VII AS [Viernes],HorSig2.VIF AS [Viernes ] FROM ((Materias INNER JOIN
HorSig2 ON Materias.Materia = HorSig2.Materia) INNER JOIN Profesores ON Profesores.Clave =
HorSig2.MtoTit) INNER JOIN Academias ON  Academias.Clave = Materias.Academia WHERE
Academias.Academia LIKE '%" + sNombreAcademia + "%';";

            SqlConnection oConexionAlServidor = new SqlConnection(sCadenaDeConexion);

            oConexionAlServidor.Open();

            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);

            DataSet oDatos = new DataSet();

            oAdaptadorDeDatos.Fill(oDatos, sNombreDeLaTabla);

            return oDatos;
        }

        [WebMethod]
        public DataSet ConsultaDeProfesorTutor(string sNombre, string sCadenaDeConexion,
string sNombreDeLaTabla)
        {
            string QRY = "";

            QRY = "SELECT  Profesores.Clave, Profesores.Nombre AS [Profesor tutor],
HorTutor.Lugar AS [Cúbiculo], HorTutor.LUI AS[Lunes], HorTutor.LUF AS [Lunes ], HorTutor.MAI
AS [Martes], HorTutor.MAF AS [Martes], HorTutor.MII AS [Miercoles], HorTutor.MIF AS
```

```
[Miercoles ], HorTutor.JUI AS [Jueves], HorTutor.JUF AS [Jueves ], HorTutor.VII AS
[Viernes], HorTutor.VIF AS [Viernes ]FROM Profesores INNER JOIN HorTutor ON Profesores.Rfc =
HorTutor.Profesor WHERE Profesores.Nombre LIKE '%" + sNombre + "%';";

            SqlConnection oConexionAlServidor = new SqlConnection(sCadenaDeConexion);

            oConexionAlServidor.Open();

            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);

            DataSet oDatos = new DataSet();

            oAdaptadorDeDatos.Fill(oDatos, sNombreDeLaTabla);

            return oDatos;

        }

        [WebMethod]
        public DataSet ConsultarHorarioGrupoSigue(string sGrupoSigue, string
sCadenaDeConexion, string sNombreDeLaTabla)
        {
            string QRY = "";

            QRY = "SELECT HorSig2.Sigue AS [Grupo],Materias.Nombre, Profesores.Nombre,
HorSig2.Salon, HorSig2.LUI AS [Lunes], HorSig2.LUF AS [Lunes ] ,HorSig2.MAI AS
[Martes],HorSig2.MAF AS [Martes ], HorSig2.MII AS [Miercoles], HorSig2.MIF AS [Miercoles
],HorSig2.JUI AS [Jueves], HorSig2.JUF AS [Jueves ], HorSig2.VII AS [Viernes],HorSig2.VIF AS
[Viernes ]FROM (HorSig2 INNER JOIN Materias ON HorSig2.Materia = Materias.Materia) INNER
JOIN  Profesores ON Profesores.Clave = HorSig2.MtoTit WHERE HorSig2.Sigue LIKE '%" +
sGrupoSigue + "%' ;";

            SqlConnection oConexionAlServidor = new SqlConnection(sCadenaDeConexion);

            oConexionAlServidor.Open();

            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);

            DataSet oDatos = new DataSet();

            oAdaptadorDeDatos.Fill(oDatos, sNombreDeLaTabla);

            return oDatos;

        }

        [WebMethod]
        public DataSet ConsultaPorAlumnoProfesorTutor(string sBoleta, string
sCadenaDeConexion, string sNombreDeLaTabla)
        {
            string QRY = "";

            QRY = "SELECT TutorAlumno.Boleta, TutorAlumno.Alumno,Profesores.Nombre AS
[Profesor tutor], HorTutor.Lugar AS [Cúbiculo],HorTutor.LUI AS[Lunes], HorTutor.LUF AS
[Lunes ], HorTutor.MAI AS [Martes], HorTutor.MAF AS [Martes], HorTutor.MII AS [Miercoles],
HorTutor.MIF AS [Miercoles ], HorTutor.JUI AS [Jueves], HorTutor.JUF AS [Jueves ],
HorTutor.VII AS [Viernes], HorTutor.VIF AS [Viernes ]FROM (TutorAlumno INNER JOIN HorTutor
ON TutorAlumno.Tutor = HorTutor.Profesor) INNER JOIN Profesores ON Profesores.Rfc =
TutorAlumno.Tutor WHERE TutorAlumno.Boleta LIKE '%" + sBoleta + "%';";

            SqlConnection oConexionAlServidor = new SqlConnection(sCadenaDeConexion);

            oConexionAlServidor.Open();

            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);

            DataSet oDatos = new DataSet();

            oAdaptadorDeDatos.Fill(oDatos, sNombreDeLaTabla);

            return oDatos;
```

```
        }

        //Este metodo realiza un insert
        [WebMethod]
        public bool InsertarNuevoRegistroPrueba(string sNuevoValor,string sCadenaDeConexion)
        {
            string QRY = "";
            QRY = "INSERT INTO Prueba (Clave,Profesor,Rfc) VALUES (" + sNuevoValor + ");";
            /*SqlConnection oConexionAlServidor = new SqlConnection(sCadenaDeConexion);
            oConexionAlServidor.Open();
            SqlDataAdapter oAdaptadorDeDatos = new SqlDataAdapter(QRY, sCadenaDeConexion);*/
            return false;

        }


    }
}
```

Anexo 2

```
protected void Button1_Click(object sender, EventArgs e)
    {
        string sConexion = "Data Source=PEPOY1\\SQLEXPRESS;Initial
Catalog=HorariosSQL;Integrated Security=True";

        sLabel = Label6.Text;
        if (TextBox1.Text == "" && TextBox2.Text == "" && TextBox4.Text == "" &&
TextBox3.Text == "")
        {
            Label6.Text = "ES NECESARIO INGRESAR ALMENOS UN CAMPO PARA BUSCAR EL HORARIO";
            Label6.ForeColor = System.Drawing.Color.Crimson;

        }

        else
        {


            //Label6.Text = sLabel;
            GridView1.Visible = GridView2.Visible = GridView3.Visible = GridView4.Visible =
true;
            localhost.Service1 B = new localhost.Service1();


            switch (RadioButtonList1.SelectedIndex)
            {
                case 0 :

                    if (TextBox1.Text != "")
                    {
                        GridView1.DataSource = B.ConsultarPorNombreDeProfesor(TextBox1.Text,
sConexion, "Profesores");
                        GridView1.DataBind();
                        if (GridView1.Columns.Count == 0 && GridView1.Rows.Count == 0)
                        {
                            Label5.Text = "NO SE ENCONTRARON COINCIDENCIAS";
                            Label5.Visible = true;
                        }

                    }

                    break;

                case 1:

                    if (TextBox2.Text != "")
                    {
                        GridView2.DataSource = B.ConsultarPorMateria(TextBox2.Text,
sConexion, "Materias");
                        GridView2.DataBind();
                        if (GridView2.Columns.Count == 0 && GridView2.Rows.Count == 0)
                        {
                            Label5.Text = "NO SE ENCONTRARON COINCIDENCIAS";
                            Label5.Visible = true;
                        }

                    }

                    break;

                case 2:

                    if (TextBox4.Text != "")
                    {
```

```
                        GridView3.DataSource = B.ConsultarPorAcademia(TextBox4.Text,
sConexion, "Academias");
                        GridView3.DataBind();
                        if (GridView3.Columns.Count == 0 && GridView3.Rows.Count == 0)
                        {
                            Label5.Text = "NO SE ENCONTRARON COINCIDENCIAS";
                            Label5.Visible = true;
                        }

                    }

                    break;

                case 3:

                    if (TextBox3.Text != "")
                    {
                        GridView4.DataSource = B.ConsultarHorarioGrupoSigue(TextBox3.Text,
sConexion, "HorSig2");
                        GridView4.DataBind();
                        if (GridView4.Columns.Count == 0 && GridView4.Rows.Count == 0)
                        {
                            Label5.Text = "NO SE ENCONTRARON COINCIDENCIAS";
                            Label5.Visible = true;
                        }
                    }

                    break;

            }


        }


    }
```