



Benemérita Universidad **A**utónoma de **P**uebla

Facultad de Ciencias de la Computación

**Herramientas Colaborativas basadas en Servicios Web
para una Plataforma de Apoyo al Aprendizaje en la
Facultad de Ciencias de la Computación**

TESIS

**Que para obtener el título de:
Licenciado en Ciencias de la Computación**

Presenta:

Larri Juárez Romero

Asesor:

Dr. Mario Anzures García

Coasesor:

Dr. Mario Rossainz López

Heroica Puebla de Zaragoza, Noviembre de 2010

Índice

Resumen.....	i
Capítulo 1. Introducción.....	1
Capítulo 2. Fundamentos teóricos.....	3
2.1. Aplicaciones Colaborativas.....	3
2.1.1. Herramientas Colaborativas.....	3
2.1.1.1. Asíncronas.....	4
2.1.1.1.1. Foro.....	4
2.1.1.1.2. Espacio para compartir archivos.....	4
2.1.1.1.3. Buzón de mensajes.....	5
2.1.1.1.4. Blog.....	5
2.1.1.2. Síncronas.....	6
2.1.1.2.1. Chat.....	6
2.1.1.2.2. Videoconferencia.....	7
2.2. Servicios Web.....	8
2.2.1. Metodología (SOA).....	9
2.2.2. Características.....	10
2.2.3. Protocolos.....	11
2.2.3.1. WSDL.....	11
2.2.3.2. SOAP.....	18
2.2.3.3. UDDI.....	19
2.2.4. Interoperabilidad.....	21
2.2.4.1. Definiciones.....	22
2.2.4.2. XML.....	23
2.3. Bases de datos.....	25
2.3.1. Modelo Entidad-Relación.....	26
2.3.2. Modelo Relacional.....	33
2.3.2.1. Estructura de las bases de datos relacionales.....	33
2.3.2.2. Algebra Relacional.....	35
2.3.3. Normalización.....	40
2.3.4. SQL.....	42
2.3.4.1. Definiciones.....	43
2.3.4.2. SQL Server 2000.....	44
2.4. Desarrollo web.....	49
2.4.1. Definiciones .NET.....	49
2.4.1.1. CLR.....	50
2.4.1.2. Biblioteca de Clases.....	51
2.4.1.3. ASP .NET.....	52
2.4.1.4. C# .NET.....	53
2.4.2. JavaScript.....	55
2.4.3. DOM.....	58
2.5. Ingeniería de Software.....	61
2.5.1. Etapas del ciclo de vida en la Ingeniería de Software.....	62

2.5.2. Modelos de la Ingeniería de Software.....	63
2.5.2.1. Modelo lineal secuencial.....	63
2.5.2.2. Modelo de construcción de prototipos.....	64
2.5.2.3. Modelo espiral.....	65
2.5.2.4. Técnicas de cuarta generación.....	66
2.5.2.5. Modelo fuente.....	66
2.5.3. Análisis y diseño orientado a objetos.....	67
2.5.3.1. Definiciones.....	68
2.5.3.2. Análisis orientado a objetos.....	70
2.5.3.3. Diseño orientado a objetos.....	72
2.5.3.4. UML.....	73
Capítulo 3. Análisis y Diseño de las herramientas colaborativas de SisAAC.....	82
3.1. Análisis.....	82
3.1.1. Casos de uso.....	82
3.2. Diseño.....	88
3.2.1. Diagramas de Clases.....	88
3.2.2. Diagramas de Secuencia.....	91
3.2.3. Diagrama Entidad-Relación.....	92
Capítulo 4. Implementación de las herramientas colaborativas de SisAAC.....	99
4.1. Herramienta colaborativa “Foro”.....	102
4.2. Herramienta colaborativa “Buzón de mensajes”.....	105
4.3. Herramienta colaborativa “Espacio para compartir archivos”.....	107
4.4. Herramienta colaborativa “Blog”.....	110
Capítulo 5. Conclusiones.....	112
Apéndice A Manual de Usuario.....	113
Apéndice B Manual Técnico.....	126
Referencias.....	152
Índice de Figuras.....	155

RESUMEN

La creciente demanda en la utilización de herramientas que permitan trabajar en grupo en áreas de conocimiento específico, la interacción que un usuario puede o debe tener con otros usuarios dentro de un ambiente compartido y la dificultad que conlleva encontrar estas herramientas integradas en una misma plataforma, que este dedicada exclusivamente al ámbito académico; son las razones principales por las cuales en este trabajo de tesis se desarrolla un conjunto de herramientas colaborativas que estarán integradas en una misma plataforma. Con lo cual se pueda brindar un entorno en donde un usuario tenga a su disposición las herramientas necesarias para que su aprendizaje no se vea limitado por factores físicos y de tiempo.

Esta tesis plantea el desarrollo de distintos Servicios Web, que son utilizados para la implementación de cuatro herramientas colaborativas para trabajo en grupo. Tanto los Servicios Web como las herramientas colaborativas son desarrollados siguiendo la arquitectura que implementa SOA (*Services Oriented Architecture*) y en la parte de la programación será utilizado el Framework.NET en donde ASP.NET será el entorno de programación web el cual proporcionara los elementos necesarios para la creación de las herramientas colaborativas. C#.NET será el lenguaje principal que se utilice para generar el código administrado de los Servicios Web y la invocación de estos en el *code behind* de las páginas web y ADO.NET será utilizado para poder establecer la comunicación entre los Servicios Web y la base de datos generada en SQL Server 2008.

CSS (*Cascade Style Sheet*) y DOM (*Document Object Model*) se limitan a ser auxiliares en la maquetación de las páginas web, en tanto que el uso de JavaScript va mas allá de otorgarle simple dinamismo a las páginas web pues por medio de él también se hace la invocación de los Servicios Web, mostrando así que la interoperabilidad se puede alcanzar en cualquier nivel del desarrollo de una aplicación.

Cuando se habla de una arquitectura orientada a servicios se está hablando de un conjunto de servicios residentes en internet o en una intranet, usando Servicios Web. SOA es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos de negocios. Además permite la creación de sistemas escalables y brinda una forma bien definida para exponer e invocar los Servicios Web, lo cual facilita la interoperabilidad entre diferentes aplicaciones y plataformas.

Los Servicios Web son un conjunto de protocolos que sirven para intercambiar datos entre aplicaciones. Las distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes y ejecutados sobre cualquier plataforma, pueden utilizar los Servicios Web para intercambiar datos en redes de computadoras.

La interoperabilidad entre aplicaciones que se ejecutan en distintas plataformas se consigue mediante la utilización de estándares tales como: WSDL, SOAP y UDDI; que permiten la descripción de, la comunicación entre y publicación de los Servicios Web siendo estos protocolos un estándar basado en XML.

Capítulo 1. Introducción

En las últimas décadas internet, la web y el enorme crecimiento de diferentes tecnologías han originado un ciberespacio que permite a la gente compartir información sin importar su localización geográfica, favoreciendo la obtención de conocimiento. Este nuevo ciberespacio requiere sistemas de software que soporten, contribuyan y fortalezcan la colaboración, comunicación y coordinación entre las personas, es decir, el trabajo en grupo o mejor conocido como Groupware. Se han desarrollado una variedad de herramientas que permiten el desarrollo de aplicaciones colaborativas o Groupware, que van desde aplicaciones asíncronas, síncronas y plataformas de todo tipo.

En la actualidad existen en la web una infinidad de aplicaciones colaborativas síncronas y asíncronas. Ejemplos de aplicaciones síncronas son el chat y videoconferencia, y de asíncronas son foros y blogs. Cada una de ellas permite la posibilidad de trabajar de manera conjunta sobre cualquier ámbito, ya sea social, profesional o académico. Sin embargo, la mayoría de este tipo de herramientas las podemos encontrar por separado, siendo muy difícil o costoso integrarlas en una misma plataforma, principalmente por problemas de interoperabilidad.

La comunidad académica de la Facultad de Ciencias de la Computación hace uso de diferentes aplicaciones colaborativas tanto para sus tareas académicas como administrativas, obviamente también las utilizan para aspectos personales. Aplicaciones como el buzón de mensajes, los foros, los blogs y espacios para compartir archivos son las más utilizadas por esta comunidad. Sin embargo, estas aplicaciones deben ser utilizadas por separado, es difícil encontrarlas de manera conjunta, así como orientadas al ámbito académico de su interés, es decir, temas relacionadas con la currícula de la FCC. Además, la mayoría de las aplicaciones utilizadas no están basadas en Servicios Web.

En este trabajo de tesis, se desarrollarán un conjunto de Servicios Web que permitirán crear cuatro herramientas colaborativas asíncronas centradas en la colaboración e interacción de grupos de estudiantes para aprender un tema de su agrado o interés. Los Servicios Web pueden ser reutilizados y adaptados en diferentes plataformas, porque son los dos más grandes beneficios de SOA (*Services Oriented Architecture*). Por una parte, es más fácil reutilizar un servicio web que soporta la funcionalidad común de varias aplicaciones que reutilizar aplicaciones completas a través de diferentes escenarios. Por otra, la adaptación puede ser alcanzada mediante la sustitución, modificación o incluso incorporación de uno o varios Servicios Web para ajustar la aplicación a las características del nuevo escenario colaborativo.

Estas herramientas se integrarán en la plataforma denominada SisAAC (*Sistema de Apoyo al Aprendizaje Colaborativo*) que será creada en la tesis denominada “Una Plataforma de Apoyo al Aprendizaje basada en Servicios Web”. Estos dos trabajos de tesis se complementan dando como resultado una plataforma de apoyo al aprendizaje colaborativo que brinda las herramientas colaborativas antes mencionadas; por tal razón fue necesario un trabajo en conjunto.

El presente trabajo de tesis tiene por objetivo general el desarrollar un conjunto de Servicios Web que proporcionen diversas herramientas colaborativas. Siendo objetivos específicos de este: el desarrollo un buzón de mensajes, un foro, un blog, un espacio para compartir archivos y la integración de estas herramientas en la plataforma SisAAC.

La presente tesis se organiza de la siguiente manera:

Capítulo 2. Describe los fundamentos teóricos en los que se basa esta tesis, definiendo conceptualmente los paradigmas y tecnologías necesarios para el desarrollo de los Servicios Web, las herramientas colaborativas y las interfaces con las que interactuara el usuario.

Capítulo 3. Explica la forma en que fue realizado el análisis y diseño de las herramientas colaborativas propuestas para esta tesis, utilizando herramientas UML (*Unified Modeling Language*) tales como: diagramas de clases, casos de uso y diagramas de secuencia. También se describe como se hace el diseño del diagrama Entidad-Relación y su conversión al Modelo Relacional a través de una herramienta CASE (*Computer Assisted Software Engineering*).

Capítulo 4. Detalla la implementación de los Servicios Web correspondientes a las cuatro herramientas colaborativas, esto, en cuanto al uso de la plataforma de desarrollo .NET de Microsoft (ASP, ADO y C#), haciendo mención del uso de JavaScript, DOM (*Document Object Model*) y CSS (*Cascade Style Sheet*).

Capítulo 5. Finalmente en este capítulo se presentan las conclusiones que se han deducido del trabajo realizado.

Capítulo 2. Fundamentos teóricos

Los fundamentos teóricos de las secciones de este capítulo, abordan temas tales como: ingeniería de software, bases de datos y Servicios Web por nombrar solo algunos, que más adelante en esta tesis servirán para sustentar y argumentar el uso de las metodologías y tecnologías que han sido elegidas para el desarrollo de las herramientas colaborativas que serán definidas en un capítulo posterior.

2.1. Aplicaciones Colaborativas

Las aplicaciones colaborativas o Groupware permiten realizar el trabajo en grupo. El término Groupware es una contracción de las palabras *Group* y *Software* y fue acuñado por Peter y Truddy Johnson-Lenz en 1978. Es el software para los sistemas CSCW (*Computer Supported Cooperative Work -Trabajo Cooperativo Soportado por Computadora-*), que permite a los usuarios interactuar directamente entre ellos, ya sea preparando un documento, consultando una base de datos o, incluso jugando, utilizando el ordenador como herramienta de interacción. Las aplicaciones Groupware (también conocidas como aplicaciones colaborativas) se definen como sistemas basados en computadoras que asisten a grupos de personas implicadas en una tarea (o meta), proporcionando una interfaz para un entorno compartido [Ellis 1991].

Para algunos autores, los términos de CSCW y Groupware se refieren a lo mismo; sin embargo, la principal diferencia es que CSCW es la disciplina que analiza el trabajo en grupo mientras groupware es la tecnología que lo permite [Martínez 2005].

Groupware son aquellas herramientas que permiten el trabajo en grupo, mediante las cuales distintas personas situadas en diferentes localizaciones del mundo pueden trabajar en un mismo proyecto, comunicarse, cooperar, resolver problemas, competir o incluso negociar, para la elaboración de una tarea común.

En la siguiente sección se explicaran tanto las aplicaciones síncronas como las asíncronas, en particular nos centraremos en algunas aplicaciones colaborativas asíncronas debido a que son las que se desarrollaran en esta tesis, tales como: buzón de mensajes, espacio para compartir archivos, blogs y foros.

2.1.1. Herramientas Colaborativas

Las herramientas colaborativas permiten el trabajo en grupo vía internet y hoy en día son muy utilizadas por la mayoría de la gente por el auge de la web. Estas herramientas, básicamente permiten acceder a ciertos servicios que facilitan a los usuarios comunicarse y trabajar conjuntamente sin importar que no estén reunidos en un mismo lugar físico. En general, con ellas se puede compartir información en determinados formatos (audio, texto, video, etc.), y en algunos casos producir conjuntamente nuevos materiales producto de la colaboración. Muchas de ellas proveen avanzadas funcionalidades que facilitan tareas como publicación de información, búsquedas, filtros, accesos, privilegios, etc.

Por lo general las herramientas colaborativas tienen un funcionamiento similar, trabajan bajo el concepto de cliente-servidor, donde los clientes son los usuarios que hacen uso de un servicio y el servidor se encarga de ofrecer este servicio. Comúnmente se utilizan programas clientes para acceder a estos servicios, tal es el caso de un navegador que hace

las veces de cliente para acceder a un servicio que puede ser por ejemplo un servidor web que presenta una página web.

Las herramientas colaborativas requieren dos tipos de programas o software para poder establecer una “comunicación” entre los usuarios que hacen uso de un servicio.

- **Servidor:** Primero tenemos los programas servidores, que son aquellos que permiten ofrecer un servicio, generalmente este se ejecutara en un servidor web, accesible desde internet, una intranet o una red privada o algunas de sus variaciones. Su función es coordinar el flujo de información y alojar en muchos casos dicha información.
- **Cliente:** En segundo lugar tenemos los programas clientes, que son el software que posee cada usuario en su computadora y que le permite acceder a los servicios, y que son entonces capaces de ver información publicadas por otros usuarios o publicar la propia.

2.1.1.1. Asíncronas

Son herramientas de comunicación que permiten compartir la información en diferente tiempo, el uso de herramientas asíncronas nos permiten el almacenamiento así como la visualización de la información para que esté disponible para todos, independientemente de el momento en que se conecten [Martínez 2005]. De esta forma la información queda recogida en las bases de datos del sistema, las más comunes: los Grupos de Noticias, Listas de distribución, Correo Electrónico, y los Foros de Discusión.

2.1.1.1.1. Foro

El Foro [Foro 2010] es una herramienta web donde un grupo de personas mantienen conversaciones en torno a un tema en común y específico. Es muy común que los foros tengan una serie de normas que sus miembros deben seguir para integrar el grupo, las que tienen como finalidad mantener un clima ameno entre sus integrantes para poder intercambiar opiniones e información de la mejor manera posible. Forman parte del foro, el administrador, los moderadores y los miembros en general. El primero, quien suele definir diversos foros sobre una plataforma, es el que inicia las discusiones con una pregunta disparadora y una vez acabado el tema, puede dar una mirada general de lo que se estuvo viendo hasta el momento. Mientras que los moderadores son miembros, generalmente designados por el administrador, que tienen ciertos privilegios como por ejemplo eliminar o modificar ciertas discusiones, a fin de evitar enfrentamientos entre los usuarios y, además, para que todo se mantenga dentro de las normas del foro. En el caso concreto de la aplicación desarrollada en este trabajo de tesis, que es para fines académicos se ha omitido la designación de un administrador y moderadores, ya que cualquier usuario contara con los privilegios para crear algún foro, asumiendo que el comportamiento de la comunidad siempre se mantendrá dentro del reglamento.

2.1.1.1.2. Espacio para compartir archivos

El espacio para compartir archivos [BSCW 2010], [BSCW-w 2010], es un área web donde las personas que deciden colaborar en un proyecto común pueden almacenar y gestionar documentos en común. Estos documentos pueden ser de muy diversas clases:

textos, imágenes, archivos multimedia, páginas web, etc. Los documentos compartidos no se encuentran físicamente almacenados en las computadoras de los colaboradores, sino en un servidor situado en internet, de modo que están permanentemente accesibles. Como ejemplo cito al sistema BSCW (*Basic Support for Cooperative Work*) o, también, Be Smart (*Cooperate Worlwide*), que se originó en el Instituto Fraunhofer de Tecnología de la Información Aplicada FIT de Alemania a mediados de la década de 1990. BSCW es una herramienta informática de tipo Groupware cuyo objetivo es facilitar el trabajo en grupo mediante el uso únicamente de un navegador web y de una conexión a internet.

2.1.1.1.3. Buzón de mensajes

Un buzón de mensajes [UCOL 2010], es una herramienta de comunicación asíncrona que sirve para almacenar información de tipo texto que le ha sido enviada a un usuario. Esta herramienta utiliza un servidor para almacenar físicamente los mensajes a los cuales el usuario tendrá acceso hasta el momento en el cual determine eliminarla. A pesar de que el uso de la herramienta de buzón de mensajes (e-mail) es ampliamente conocido y utilizado. Cuando se trata de un ambiente académico en el cual, la herramienta de buzón de mensajes no se basa en el uso del protocolo SMTP para correo electrónico, no hay herramientas prefabricadas que permitan la creación y administración de estos, ya que se trata de mensajería interna. Por tanto, la implementación de una herramienta especializada en el ámbito académico que proporcione este servicio emulando un buzón de correo electrónico brindara la facilidad de compartir información entre los usuarios sin la necesidad de recurrir a un servicio externo.

2.1.1.1.4. Blog

Un blog [Crespo 2006] o en español también una bitácora, es un sitio web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente. Este término inglés *blog* o *weblog* proviene de las palabras *web* y *log* ('log' en inglés = diario). Se utiliza preferentemente cuando el autor escribe sobre su vida propia como si fuese un diario, pero publicado en la web. Un blog es, entre otras cosas, una interesante y flexible herramienta de interacción para docentes y alumnos, ya que permite otras formas de producir y publicar contenidos sin la necesidad de ser experto en informática. Es el paso a la multiplicidad de formatos y lenguajes, ser parte de un entramado social en el cual los contenidos se dinamizan, resinifican y renuevan permanentemente basado en la idea de comunidades y redes; forma redes y genera espacios colaborativos y de análisis, acercándonos a la descentralización de la información y estimulando los procesos de creación colaborativa y desdibujando el clásico pensamiento lineal.

Aportaciones de los blogs:

- Aportan distintos niveles de redacción y escritura, si se toman como si fueran una redacción periodística.
- Permiten valorar nuevas formas de aprender, lo que algunos autores han llamado la lecto-escritura.

- Son herramientas colaborativas asíncronas que potencia la cohesión del grupo y la interacción profesor-alumno.
- Permiten disponer de un espacio para encontrar fuentes documentales para el área de conocimiento.

2.1.1.2. Síncronas

Las herramientas síncronas son plataformas integradas de comunicación en tiempo real, que permiten la compartición de información de uno a uno o de muchos a muchos, el uso de herramientas síncronas permite que la colaboración sea más eficaz en el sentido que permite obtener la información y el consenso en un grupo de una manera más rápida. Este tipo de colaboración ofrece las principales ventajas de una colaboración cara a cara [Martínez 2005], por ejemplo los Chat y Videoconferencia.

2.1.1.2.1. Chat

El chat [Chat 2005] es un sistema mediante el cual dos o más personas pueden comunicarse a través de internet, en forma simultánea, es decir en tiempo real, por medio de texto, audio y hasta video, sin importar su localización geográfica. Un chat está conformado por una o varias salas o canales, los cuales son cuartos virtuales en donde la gente se reúne para comunicarse e intercambiar ideas sobre un tema en particular, o puedes platicar en privado con personas conocidas o desconocidas. Técnicamente podemos decir que los chat *room* son lugares virtuales que se encuentran en un servidor de internet el cual administra todos los mensajes, tanto los que se envían como los que se reciben.

El chat fue evolucionando y paso desde los precarios BBS (*Bulletin Board System*, una de las formas más primitivas de establecer conexiones entre computadoras) a los universos virtuales, que incluyen audio y video. Al chat se lo conoce también como IRC (*Internet Relay Chat*). El chat funciona del mismo modo que la mayoría de los servicios en internet, ya sea la web, el e-mail o los mensajeros instantáneos tipo el ICQ (*I seek you -te busco-*). Estos usan el esquema conocido con el nombre de cliente/servidor. Cada uno de los integrantes de una sala se conecta mediante un programa que recibe el nombre de "cliente". Todos los clientes, a su vez se conectan a un servidor que muestra toda la información de los otros conectados.

El estilo del chat al que se ingresa depende del tipo del cliente que se utilice. Si el cliente elegido es un browser o navegador (como el Internet Explorer o el Mozilla) se tiene acceso a un chat en el que la web es un tablero de operaciones; y si se opta por los llamados clientes IRC (como el mIRC, el Pirch, etc.) se tiene acceso a lo que se conoce como chat vía "IRC puro". También se pueden usar los mensajeros instantáneos como el ICQ o InstantTerra, o recurrir a los "mundos virtuales" y las videoconferencias.

El web chat es la modalidad de conversación en tiempo real por la red que permite acceder a una sala mediante un navegador (previo ingreso a un sitio). Es el estilo de chat que más atrae a la gente. El funcionamiento del web chat es sencillo. Basta con entrar a un sitio, dirigirse a la zona dedicada al chat y elegir una sala y un *nick* (nombre de usuario) e ingresar. El tablero de operaciones es fácil de maniobrar. Lo básico del sistema son tres ventanas: La primera, y la más grande, muestra los diálogos de los usuarios; la segunda, la lista de personas conectadas (ubicada generalmente a la derecha), y la tercera es un

pequeño espacio donde se debe escribir lo que se quiera decir al resto de las personas que ingresaron a ese web chat.

Hay sitios que convocan a diferentes personalidades de la cultura, la política, el deporte o el espectáculo, que es muy usado por los sitios locales AOL, Ciudad Internet, elsitio.com, UOL y Terra. Consiste en invitar a una figura de cualquiera de esos ámbitos para que chatee con la gente que ingrese en la sala en ese momento. Sin embargo, este sistema todavía tiene algunas desventajas que los administradores de chat no pueden superar: si hay demasiada gente suele producirse un caos incontrolable y ni el invitado ni los usuarios disfrutan del momento.

2.1.1.2.2. Videoconferencia

Se le llama Videoconferencia [Videoconf 2005] al sistema que nos permite llevar a cabo el encuentro de varias personas ubicadas en sitios distantes, y establecer una conversación como lo harían si todas se encontraran reunidas en una sala de juntas.

Como sucede con todas las tecnologías nuevas, los términos que se emplean no se encuentran perfectamente definidos. La palabra "Teleconferencia" está formada por el prefijo "tele" que significa distancia, y la palabra "conferencia" que se refiere a encuentro, de tal manera que combinadas establecen un encuentro a distancia.

En los Estados Unidos la palabra teleconferencia es usada como un término genérico para referirse a cualquier encuentro a distancia por medio de la tecnología de comunicaciones; de tal forma que frecuentemente es adicionada la palabra video a "teleconferencia" o a "conferencia" para especificar exactamente a qué tipo de encuentro se está haciendo mención. De igual forma se suele emplear el término "audio conferencia" para hacer mención de una conferencia realizada mediante señales de audio.

El término "videoconferencia" ha sido utilizado en los Estados Unidos para describir la transmisión de video en una sola dirección usualmente mediante satélites y con una respuesta en audio a través de líneas telefónicas para proveer una liga interactiva con la organización.

En Europa la palabra teleconferencia se refiere específicamente a las conferencias o llamadas telefónicas, y la palabra "videoconferencia" es usada para describir la comunicación en dos sentidos: audio y video. Esta comunicación en dos sentidos de señales de audio y de video es lo que nosotros llamaremos "videoconferencia".

Existen algunos términos que pueden crear confusión con respecto a videoconferencia, como puede ser el término "televisión interactiva"; este término ha sido empleado para describir la interacción entre una persona y un programa educativo previamente grabado en un disco compacto (Láser disc) pero no requiere de la transmisión de video.

La videoconferencia puede ser dividida en dos áreas:

- Videoconferencia Grupal o Videoconferencia sala a sala con comunicación de video comprimido a velocidades desde 64 Kbps (E0, un canal de voz) hasta 2.048 Mbps (E1, 30 canales de voz).
- Videotelefonía, la cual está asociada con la Red Digital de Servicios Integrados mejor conocida por las siglas "ISDN" operando a velocidades de 64 y 128 Kbps. Esta forma de videoconferencia está asociada a la comunicación personal o videoconferencia escritorio a escritorio.

Para fines de estudio y de diseño los sistemas de videoconferencia suelen subdividirse en tres elementos básicos que son:

- La red de comunicaciones: Para poder realizar cualquier tipo de comunicación es necesario contar primero con un medio que transporte la información del transmisor al receptor y viceversa o paralelamente (en dos direcciones). En los sistemas de videoconferencia se requiere que este medio proporcione una conexión digital bidireccional y de alta velocidad entre los dos puntos a conectar. El número de posibilidades que existen de redes de comunicación es grande, pero se debe señalar que la opción particular depende enteramente de los requerimientos del usuario
- La Sala de Videoconferencia: Es el área especialmente acondicionada en la cual se alojará el personal de videoconferencia, así como también, el equipo de control, de audio y de video, que permitirá el capturar y controlar las imágenes y los sonidos que habrán de transmitirse hacia el(los) punto(s) remoto(s). El nivel de confort de la sala determina la calidad de la instalación. La sala de videoconferencia perfecta es la sala que más se asemeja a una sala normal para conferencias; aquellos que hagan uso de esta instalación no deben sentirse intimidados por la tecnología requerida, más bien deben sentirse a gusto en la instalación. La tecnología no debe notarse o debe de ser transparente para el usuario.
- Codec: Las señales de audio y video que se desean transmitir se encuentran por lo general en forma de señales analógicas, por lo que para poder transmitir esta información a través de una red digital, ésta debe de ser transformada mediante algún método a una señal digital, una vez realizado esto se debe de comprimir y multiplexar estas señales para su transmisión. El dispositivo que se encarga de este trabajo es el CODEC (Codificador/Decodificador) que en el otro extremo de la red realiza el trabajo inverso para poder desplegar y reproducir los datos provenientes desde el punto remoto. Existen en el mercado equipos modulares que junto con el CODEC, incluyen los equipos de video, de audio y de control, así como también equipos periféricos como pueden ser: Tabla de anotaciones, Convertidor de gráficos informáticos, Cámara para documentos, Proyector de video-diapositivas, PC, Videgrabadora, Pizarrón electrónico, etc.

2.2. Servicios Web

Un servicio Web es un conjunto de estándares y protocolos que sirven para intercambiar datos entre aplicaciones distribuidas. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los diferentes Servicios Web para intercambiar datos en redes de ordenadores como el internet, cuyo objetivo es ofrecer servicios; los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web.

Los Servicios Web son parte del nuevo paradigma denominada cómputo orientado a servicios, que utiliza servicios como base fundamental para el desarrollo de aplicaciones de rápido y bajo costo y una fácil composición de aplicaciones distribuidas incluso en entornos heterogéneos.

La promesa del SOC (*Service Oriented Computing*) es un mundo de cooperación de servicios donde los componentes de una aplicación son ensamblados con un pequeño

esfuerzo en una red de servicios que pueden ser débilmente acoplados para crear dinámicas de negocios flexibles y aplicaciones ágiles que pueden abarcar organizaciones y plataformas de computo.

En SOC se identifican los siguientes niveles de integración:

- La integración a nivel de Servicios Básicos *e-services*: Servicios basados en internet hacen énfasis en la integración de servicios con otras aplicaciones disponibles en internet.
- Servicios compuestos *e-workflow*: Está formado por servicios básicos

Para construir un modelo basado en servicios el computo orientado a servicios utiliza la arquitectura orientada a servicios (SOA) la cual es la metodología para reorganizar aplicaciones de software previamente desarrolladas, incorporándolas en un conjunto de servicios interconectados, cada uno accesible a través de interfaces y protocolos de mensajería.

Los Servicios Web son componentes software que permiten a los usuarios usar aplicaciones de negocio que comparten datos con otros programas modulares, vía internet. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos web estándar, como XML, SOAP, UDDI o WSDL. El objetivo final es la creación de un directorio online de Servicios Web, que pueda ser localizado de un modo sencillo y que tenga una alta fiabilidad.

2.2.1. Metodología (SOA)

SOA (*Services Oriented Architecture -Arquitectura Orientada a Servicios-*) [Bean 2010] es un protocolo que permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma estándar de exposición e invocación de servicios (comúnmente pero no exclusivamente Servicios Web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros, reduce los costos y el tiempo de desarrollo, es decir pueden reutilizarse fácilmente y convertirse en nuevas aplicaciones compuestas, reduce costos de mantenimiento, los servicios reutilizables reducen el grado de complejidad interna del sistema, además permite aumentar la calidad de los servicios en múltiples ciclos de prueba de diferentes consumidores de servicios, SOA permite reducir los costos de integración por lo que los servicios estandarizados pueden trabajar en conjunto, permitiendo que las aplicaciones dispares se conecten con rapidez y facilidad, reduce el riesgo en general relacionado con el cumplimiento.

Los elementos que intervienen en una Arquitectura Orientada a servicios y sus relaciones es la siguiente (ver Figura 2.1).

- Proveedor de servicios: se encarga de implementar un servicio y publicarlo en un repositorio.
- Repositorio de Servicios: En el repositorio se almacenan las descripciones de servicios, para que ser solicitados por un cliente.
- Consumidor: El cual solicitará un servicio previamente publicado en el repositorio, obteniendo una descripción, que será utilizada para obtener los mensajes necesarios para invocar el servicio deseado.
- Servicio: es una funcionalidad diseñada, codificada y encapsulada de modo que sea realizable por otros servicios y aplicaciones, no es importante la tecnología en que

este constituido; cada vez que hablamos del encapsulamiento de las funcionalidades de enrutamiento y transformación nos referimos a ESB (*Enterprise Service BUS*) la cual se convierte en una de las piezas más importantes de SOA, ya que evita que cada servicio tenga que preocuparse de cómo comunicarse con el resto.

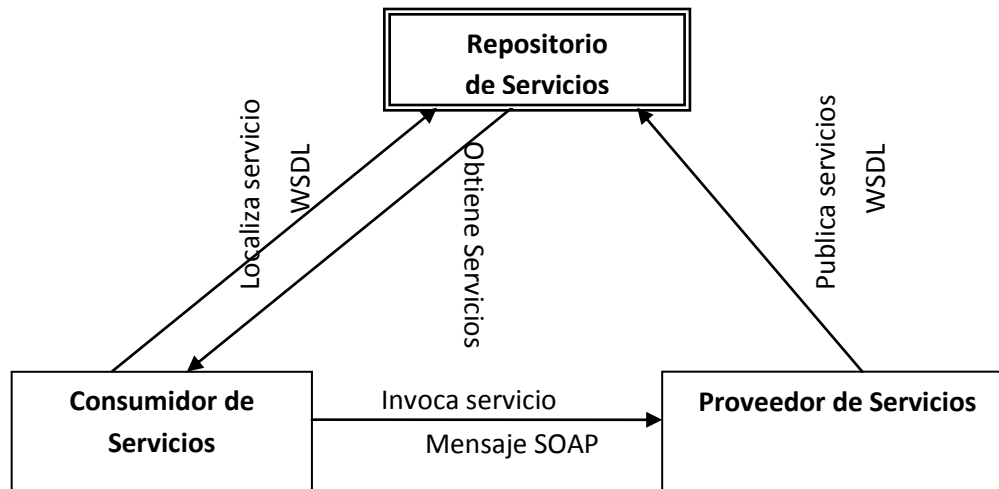


Figura 2.1. Arquitectura orientada a servicios básica

2.2.2. Características

Las principales características [Jiménez 2009] de un servicio basado en una arquitectura orientada a servicios son:

- Bajo acoplamiento. Esto quiere decir que los diversos servicios no tengan una dependencia muy alta entre ellos, pudiendo funcionar de manera autónoma si es necesario.
- Contrato entre servicios. Los servicios se comprometen a cumplir un acuerdo establecido para sus comunicaciones.
- Autonomía. Los servicios tienen el control sobre la lógica que encapsulan.
- Abstracción. Los servicios ocultan determinada lógica que no está descrita en el contrato.
- Reutilización. La lógica es dividida en servicios con la intención de promover la reutilización.
- Composición. Los servicios pueden ser agrupados para dar lugar a servicios compuestos.
- Sin estado. Los servicios deben minimizar la cantidad de información que retienen para realizar una determinada actividad.
- Descubrimiento. Los servicios deben ser suficientemente descriptivos para poder ser encontrados con mecanismos de descubrimiento.
- Mejora en la Integración de los Componentes. Siguiendo un patrón de comunicación y de datos similar se pueden acoplar nuevos servicios a una aplicación.
- Facilita la reusabilidad. El mismo servicio puede servir para distintas aplicaciones en la organización.

- Agilidad organizacional. El coste y el esfuerzo para responder y adaptarse a los cambios tecnológicos es más pequeño. El coste para la integración de nuevos servicios dentro de una aplicación heredada (llamadas comúnmente *legacy systems*) es mucho menor. Esto se consigue con la implementación de Servicios Web que denominaremos *ADAPTADORES*, y que logran proporcionar a las aplicaciones heredadas no basadas en SOA una interfaz para comunicarse con la arquitectura basada en SOA.

2.2.3. Protocolos

Los estándares son definiciones obtenidas por consenso que aprueban, reconocen y promueven organizaciones internacionales para el intercambio de información o del conocimiento. Generalmente estos organismos están formados por el conjunto de empresas más representativas de un sector o de un campo de la producción.

Los estándares permiten que las industrias desarrollen componentes con las garantías suficientes de: interoperabilidad, funcionalidad y calidad. Ayudan a desarrollar los bloques básicos sobre los cuales pueden ser construidos desarrollos tecnológicos. Los estándares son extremadamente importantes en la computación, ya que permiten que se combinen productos de diferentes fabricantes para el desarrollo de sistemas, tanto de software como de hardware. Sin estándares, solo los productos de una misma compañía podrán ser utilizados de forma conjunta. Actualmente existen estándares para diversos protocolos de comunicación, formatos de datos, lenguajes de programación, etc. Los organismos más importantes de estandarización son: ANSI, IEEE, ISO y W3C.

Los Servicios Web se construyen sobre estándares y a su vez pretenden ser un estándar mediante el cual sea posible construir sistemas a partir de piezas heterogéneas, desarrolladas por diversos fabricantes, funcionando en distintos sistemas y construidas con distintas tecnologías, con la finalidad de proporcionar la interoperabilidad entre servicios. La es la condición mediante la cual los sistemas heterogéneos pueden intercambiar procesos o datos.

Los principales estándares para el desarrollo de Servicios Web son WSDL, SOAP y UDDI, que serán descritos en las siguientes secciones

2.2.3.1. WSDL

WSDL (*Web Services Description Language -Lenguaje de descripción de Servicios Web-*) [WSDL 2001], es un documento escrito en XML. El documento describe un servicio Web en el que se especifica la ubicación del servicio y las operaciones (o métodos) que el servicio expone.

Un documento WSDL es simplemente un conjunto de definiciones. Hay un elemento “definitions” en la raíz, y dentro de las definiciones. La gramática es la siguiente:

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
  <import namespace="uri" location="uri"/>*
  <wsdl:documentation .... /> ?
  <wsdl:types> ?
    <wsdl:documentation .... />?
    <xsd:schema .... />*
    <!-- extensibility element --> *
  </wsdl:types>
  <wsdl:message name="nmtoken"> *
```

```

    <wsdl:documentation .... />?
    <part name="nmtoken" element="qname"? type="qname"?/> *
</wsdl:message>
<wsdl:portType name="nmtoken">*
  <wsdl:documentation .... />?
  <wsdl:operation name="nmtoken">*
    <wsdl:documentation .... /> ?
    <wsdl:input name="nmtoken"? message="qname">?
      <wsdl:documentation .... /> ?
    </wsdl:input>
    <wsdl:output name="nmtoken"? message="qname">?
      <wsdl:documentation .... /> ?
    </wsdl:output>
    <wsdl:fault name="nmtoken" message="qname"> *
      <wsdl:documentation .... /> ?
    </wsdl:fault>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="nmtoken" type="qname">*
  <wsdl:documentation .... />?
  <!-- extensibility element --> *
  <wsdl:operation name="nmtoken">*
    <wsdl:documentation .... /> ?
    <!-- extensibility element --> *
    <wsdl:input> ?
      <wsdl:documentation .... /> ?
      <!-- extensibility element -->
    </wsdl:input>
    <wsdl:output> ?
      <wsdl:documentation .... /> ?
      <!-- extensibility element --> *
    </wsdl:output>
    <wsdl:fault name="nmtoken"> *
      <wsdl:documentation .... /> ?
      <!-- extensibility element --> *
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="nmtoken"> *
  <wsdl:documentation .... />?
  <wsdl:port name="nmtoken" binding="qname"> *
    <wsdl:documentation .... /> ?
    <!-- extensibility element -->
  </wsdl:port>
  <!-- extensibility element -->
</wsdl:service>
<!-- extensibility element --> *
</wsdl:definitions>

```

Un documento WSDL describe un servicio web utilizando estos elementos principales:

- <types>: Los tipos de datos que utiliza el servicio web.
- <message>: Los mensajes que utiliza el servicio web.
- <portType>: Las operaciones realizadas por el servicio web.
- <binding>: Los protocolos de comunicación utilizados por el servicio web.
- <operation>: Una descripción abstracta de una acción soportada por un servicio.
- <port>: Un único punto definido como una combinación de un enlace una dirección de red.

- `<service>`: Una colección de puntos finales relacionados.

Types: El elemento WSDL `<types>` encierra las definiciones de los tipos de datos que tienen relevancia para el intercambio de mensajes. Para una máxima interoperabilidad y neutralidad de plataforma, WSDL prefiere el uso de XSD como el tipo de sistema canónico, y lo trata como sistema de tipo intrínseco.

```
<definitions .... >
  <types>
    <xsd:schema .... />*
  </types>
</definitions>
```

El sistema de tipos XSD se puede utilizar para definir los tipos en un mensaje, independientemente de si el formato de alambre resultante es realmente XML, o si el esquema resultante XSD valida el formato de alambre en particular. Esto es especialmente interesante si habrá varios enlaces para el mismo mensaje, o si sólo hay un enlace. En estos casos, el enfoque recomendado para la codificación de tipos abstractos utilizando XSD es el siguiente:

- Usar la forma de elemento (no de atributo).
- No incluir atributos o elementos que son propios de la codificación de alambre.
- Los tipos Array se deben extender del tipo Array como se define en el esquema de codificación SOAP v1.1. Utilice el nombre `ArrayOfXXX` para los tipos de matriz (donde XXX es el tipo de los elementos de la matriz). El tipo de los elementos de la matriz y las dimensiones de la matriz se especifica mediante un valor predeterminado para el atributo `soapenc:arrayType`. En el momento de escribir esto, la especificación XSD no tiene un mecanismo para especificar el valor predeterminado de un atributo que contiene un valor QName. Para superar esta limitación, WSDL introduce el atributo **arrayType** (del espacio de nombres `http://schemas.xmlsoap.org/wsdl/`) que tiene la semántica para proporcionar el valor predeterminado. Si se revisa XSD para dar soporte a esta funcionalidad, el mecanismo de revisión se debe utilizar en favor de el atributo `arrayType` definido por WSDL.
- Utilizar el tipo `xsd:anyType` para representar un campo/parámetro que puede tener cualquier tipo.

Sin embargo, es de esperar que un solo tipo de sistema gramatical pueda y deba ser usada para describir todos los tipos abstractos presentes y futuros, WSDL permite agregar tipos de sistema a través de la extensibilidad de los elementos. Un elemento de extensibilidad puede aparecer debajo de el elemento `types` para identificar el tipo de definición de sistema que está siendo usado y proveer un elemento contenedor XML para la definición del tipo. La función de estos elementos puede ser comparada con el elemento `schema` del esquema de lenguaje XML.

```
<definitions .... >
  <types>
    <!-- type-system extensibility element --> *
  </types>
</definitions>
```

Message: El elemento WSDL <message> define los elementos de datos de una operación. Cada mensaje puede consistir en una o más partes, las partes se puede comparar con los parámetros de una llamada a la función en un lenguaje de programación tradicional. La sintaxis para definir un mensaje es la siguiente:

```
<definitions...>
  <message name="nmtoken">*
    <part name="nmtoken" element="qname"? type="qname"?>*
  </message>
</definitions>
```

El atributo name del mensaje provee un unico nombre entre todos los mensajes definidos en el documento adjunto del WSDL.

PortType: El elemento WSDL <portType> es el elemento más importante del documento WSDL en él se describe un servicio web, las operaciones que se pueden realizar, y los mensajes que están involucrados. Este elemento se puede comparar con una biblioteca de funciones (o un módulo, o una clase) en un lenguaje de programación tradicional.

```
<wsdl:definitions .... >
  <wsdl:portType name="nmtoken">
    <wsdl:operation name="nmtoken" .... /> *
  </wsdl:portType>
</wsdl:definitions>
```

El atributo name de PortType provee un único nombre entre todos los PortType definidos dentro del documento XML.

Una operación es invocada a través del atributo name. WSDL tiene cuatro formas primitivas de transmisión que una terminal puede soportar:

- One-way. La terminal recibe un mensaje.
- Request-response. La terminal recibe un mensaje, y envía un mensaje correlacionado.
- Solicit-response. La terminal envía un mensaje, y recibe un mensaje correlacionado.
- Notification. La terminal envía un mensaje.

WSDL se refiere a estas primitivas como *operations* (operaciones). A pesar de que request/response o solicit/response pueden ser modelados abstractamente usando dos mensajes one-way, es preferible modelar estas como operaciones primitivas porque:

- Son muy comunes.
- La secuencia puede ser correlacionada sin tener que introducir más flujo complejo de información.
- Algunas terminales solo pueden recibir mensajes si estas son el resultado de una solicitud respondida síncronamente.

A pesar de que la estructura de WSDL soporta los enlaces para estas cuatro formas de transmisión primitivas, WSDL solo define enlaces para las primitivas One-Way y Request-Response. Se espera que las especificaciones que definen el protocolo para Solicit-

Response o Notification puedan también incluir enlaces en las extensiones WSDL que permitan utilizar esas primitivas.

La gramática para la operación one-way es la siguiente:

```
<wsdl:definitions .... > <wsdl:portType .... > *
  <wsdl:operation name="nmtoken">
    <wsdl:input name="nmtoken"? message="qname"/>
  </wsdl:operation>
</wsdl:portType >
</wsdl:definitions>
```

El elemento input especifica el formato abstracto del mensaje para la operación one-way.

La gramática para la operación request-response es la siguiente:

```
<wsdl:definitions .... >
  <wsdl:portType .... > *
    <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
      <wsdl:input name="nmtoken"? message="qname"/>
      <wsdl:output name="nmtoken"? message="qname"/>
      <wsdl:fault name="nmtoken" message="qname"/>*
    </wsdl:operation>
  </wsdl:portType >
</wsdl:definitions>
```

Los elementos input y output especifican el formato abstracto del mensaje para request y response, respectivamente. El elemento opcional Fault especifica el formato abstracto del mensaje para algún mensaje de error que pueda producirse como resultado de la operación.

Tenga en cuenta que una operación request-response es una noción abstracta, un enlace particular debe ser consultado para determinar cómo los mensajes se envíen: dentro de una sola comunicación (por ejemplo, una HTTP request/response), o como dos comunicaciones independientes (por ejemplo, dos peticiones HTTP).

La gramática para la operación solicit-response es la siguiente:

```
<wsdl:definitions .... >
  <wsdl:portType .... > *
    <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
      <wsdl:output name="nmtoken"? message="qname"/>
      <wsdl:input name="nmtoken"? message="qname"/>
      <wsdl:fault name="nmtoken" message="qname"/>*
    </wsdl:operation>
  </wsdl:portType >
</wsdl:definitions>
```

Los elementos input y output especifican el formato abstracto del mensaje para solicit y response, respectivamente. El elemento opcional Fault especifica el formato abstracto del mensaje para algún mensaje de error que pueda producirse como resultado de la operación.

Tenga en cuenta que una operación solicit-response es una noción abstracta, un enlace particular debe ser consultado para determinar cómo los mensajes se envíen: dentro de una sola comunicación (por ejemplo, una HTTP request/response), o como dos comunicaciones independientes (por ejemplo, dos peticiones HTTP).

La gramática para la operación solicit-response es la siguiente:

```
<wsdl:definitions .... >
  <wsdl:portType .... > *
    <wsdl:operation name="nmtoken">
      <wsdl:output name="nmtoken"? message="qname"/>
    </wsdl:operation>
  </wsdl:portType >
</wsdl:definitions>
```

El element output especifica el formato abstracto del mensaje para la operación notification.

Binding: El elemento WSDL <binding> define el formato del mensaje y los detalles del protocolo para las operaciones y los mensajes definidos por un PortType en particular. Puede haber cualquier número de enlaces para un portType dado. La gramática de un enlace es el siguiente

```
<wsdl:definitions .... >
  <wsdl:binding name="nmtoken" type="qname"> *
    <!-- extensibility element (1) --> *
    <wsdl:operation name="nmtoken"> *
      <!-- extensibility element (2) --> *
      <wsdl:input name="nmtoken"? > ?
        <!-- extensibility element (3) -->
      </wsdl:input>
      <wsdl:output name="nmtoken"? > ?
        <!-- extensibility element (4) --> *
      </wsdl:output>
      <wsdl:fault name="nmtoken"> *
        <!-- extensibility element (5) --> *
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

El atributo name provee un único nombre entre todos los enlaces contenidos en el documento WSDL. La extensibilidad de los elementos de enlace es usada para especificar la gramática concreta los mensajes de entrada, salida y fault. Una operación elemental dentro de un enlace específico vincula la información para la operación con el mismo nombre dentro del PortType del enlace. Dado que los nombres de operaciones no tienen la obligación de ser único (por ejemplo, en el caso de la sobrecarga de los nombres de los métodos), el atributo del elemento nombre en la operación enlazada no puede ser suficiente para identificar de forma única una operación. En ese caso, la operación debe estar identificada proporcionando el atributo name de los correspondientes elementos wsdl:input y wsdl:output.

Un binding debe especificar exactamente un protocolo.

Un binding no debe especificarla información de la dirección.

Port: El elemento WSDL <port>: define una terminal individual que especifica una sola dirección para un enlace.

```
<wsdl:definitions .... >
```

```

<wsdl:service .... > *
  <wsdl:port name="nmtoken" binding="qname"> *
    <!-- extensibility element (1) -->
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

El atributo name provee un único nombre de entre todos los puertos definidos dentro del documento WSDL.

El atributo binding se refiere al vínculo utilizado para ligar las reglas definidas por WSDL.

Un port no debe especificar más que una dirección.

Un port no debe especificar ninguna información de enlace que no sea de la dirección.

Service: El elemento WSDL <service>: define a grupos de servicios como un conjunto de puertos relacionados entre sí.

```

<wsdl:definitions .... >
  <wsdl:service name="nmtoken"> *
    <wsdl:port .... />*
  </wsdl:service>
</wsdl:definitions>

```

El atributo name provee un único nombre de entre todos los servicios definidos dentro del documento WSDL.

Los puertos en un servicio tiene la siguiente relación:

- Ninguno de los puertos se comunican entre sí (por ejemplo, la salida de un puerto no es la entrada de otro).
- Si un servicio tiene varios puertos que comparten un tipo de puerto, pero emplean diferentes enlaces o direcciones, los puertos son las alternativas. Cada puerto ofrece un comportamiento equivalente
- Mediante el examen de sus puertos, podemos determinar los tipos de servicios portuarios. Esto permite que un consumidor de un documento WSDL para determinar si se desea comunicar a un determinado servicio basado en si es o no compatible con varios tipos de puerto. Esto es útil si hay alguna relación implícita entre las operaciones de los tipos de puertos, y que todo el conjunto de tipos de puertos deben estar presentes con el fin de realizar una tarea en particular.

El directorio UDDI también ofrece varias formas para buscar servicios, por ejemplo, se puede buscar a los proveedores de un servicio en una zona geográfica concreta o en una empresa de un tipo determinado. El directorio UDDI proporcionaría toda la información necesaria, contactos, enlaces y datos técnicos que permitirán decidir cuáles son los servicios que cumplen las condiciones requeridas. UDDI es una pieza importante en el área de los Servicios Web. Sin él, las organizaciones que tienen Servicios Web no podrían dar a conocer fácilmente a sus posibles clientes que es lo que ofrecen, y los clientes no podrán conocer los datos que necesitan para acceder a esos servicios.

2.2.3.2. SOAP

SOAP acrónimo de *Simple Object Access Protocol* (Protocolo Simple de Acceso a objetos), permite el intercambio libre y estructurado de mensajes a través de internet, definiendo un mecanismo estándar de comunicación e intercambio de datos entre aplicaciones. La especificación SOAP describe:

1. Estructura del Mensaje. El contenido de un mensaje SOAP. Un mensaje SOAP es un documento XML que consiste de los siguientes elementos:
 - a. Sobre (*Envelope*). Es el elemento principal del documento XML. Provee un contenedor para información de control, la dirección de un mensaje, y el propio mensaje.
 - b. Encabezado (*Header*). Es un mecanismo genérico para agregar características adicionales a un mensaje SOAP. Puede contener información de control como atributos de calidad de un servicio, entre otros.
 - c. Cuerpo (*Body*). Es un contenedor para información requerida por el destinatario del mensaje. Contiene la identificación del mensaje y sus parámetros.
2. Estilos de Comunicación. El estilo en que se intercambian los mensajes SOAP. Existen 2 estilos de comunicación que soporta SOAP:
 - a. Llamadas Procedimientos Remotos (RPC). Es una invocación síncrona a un método que devuelve un resultado. Primero, el cliente envía una solicitud SOAP a un servidor de aplicaciones. Entonces el servidor, devuelve una respuesta a la solicitud SOAP del cliente.
 - b. Documento. Es también conocido como el estilo orientado a mensaje. Con este estilo, un documento XML se empaqueta como el contenido del mensaje SOAP.
3. Reglas de Codificación. Reglas que definen un mecanismo de serialización para el intercambio de los tipos de datos definidos en una aplicación. En un entorno distribuido, el estilo de codificación define cómo los valores de los datos definidos en una aplicación se traducen en un formato particular, esto es, serializado/deserializado.

En el caso de SOAP, el estilo de codificación determina cómo se representan los valores de los datos en los mensajes SOAP. SOAP soporta 2 estilos de codificación que son:

1. Codificación SOAP (*SOAP Encoding*). Define un conjunto de reglas para representar la estructura de datos permitiendo serializar/deserializar los valores de los tipos de datos a y desde un modelo de datos SOAP.
2. XML literal. Permite convertir directamente un árbol de nodos DOM-XML a un mensaje SOAP y viceversa. XML literal sólo usa los tipos de datos definidos en XSD (*XML Schema*).

En el núcleo de los Servicios Web se encuentra el protocolo simple de acceso a datos SOAP, que proporciona un mecanismo estándar de empaquetar mensajes. SOAP ha recibido gran atención debido a que facilita una comunicación del estilo RPC entre un cliente y un servidor remoto. Pero existen multitud de protocolos creados para facilitar la comunicación entre aplicaciones, incluyendo RPC de Sun, DCE de Microsoft, RMI de Java y ORPC de CORBA.

Una de las razones principales es que SOAP ha recibido un increíble apoyo por parte de la industria es que SOAP es el primer protocolo de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo. Compañías que en raras ocasiones cooperan entre sí están ofreciendo su apoyo a este protocolo. Algunas de las mayores Compañías que soportan SOAP son Microsoft, IBM, SUN, Microsystems y SAP.

Algunas de las Ventajas de SOAP son:

- No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.
- No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- No está atado a ninguna infraestructura de objeto distribuido: La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.
- Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de transporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- Permite la interoperabilidad entre múltiples entornos: SOAP se desarrollo sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

2.2.3.3. UDDI

UDDI (*Universal Description, Discovery and Integration*) [UDDI 2010] es un esfuerzo de la industria impulsada por todas las principales plataformas y proveedores de software, como Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP y Sun, así como una gran comunidad de operadores de mercado y líderes empresariales, más de 220 empresas son miembros de la comunidad UDDI. La idea de UDDI es muy simple: dado que existen varias compañías que proveen Servicios Web, ¿cómo encontrar un servicio que satisfaga las necesidades de un cliente? La solución que ofrece UDDI es utilizar un registro

distribuido como un mecanismo común para la publicación de las descripciones de los Servicios Web. Para esto, UDDI hace uso de estándares establecidos, entre los cuales, HTTP, XML y SOAP poseen la mayor importancia.

UDDI es uno de los principales edificios construidos para llevar a cabo los Servicios Web. UDDI provee un mecanismo para que los clientes encuentren de forma dinámica otros Servicios Web creando una plataforma interoperable estándar que permite a las compañías usar de forma rápida, fácil y dinámica los Servicios Web. Usando la interfaz de UDDI, pueden conectarse dinámicamente las empresas con los servicios proporcionados por socios externos. Para ello es necesario registrarse en UDDI y los registros pueden tener diversos propósitos y usarse en distintos contextos. Existen 2 tipos de clientes: compañías que desean publicar un servicio (y su interfaz de uso) y clientes que desean obtener cierta clase de servicios por medio de una conexión. UDDI se monta sobre SOAP y asume que las consultas y las respuestas son objetos de UDDI enviados como mensajes de SOAP.

UDDI es un marco independiente de la plataforma para la descripción de los servicios, el descubrimiento de las empresas, y la integración de servicios empresariales a través de internet.

- UDDI es un directorio para almacenar información acerca de los Servicios Web.
- UDDI es un directorio de Servicios Web e interfaces descritas por WSDL.
- UDDI se comunica a través de SOAP.
- UDDI está integrado en la plataforma Microsoft .NET.

Un registro en el directorio UDDI es un archivo XML que describe una empresa y los servicios que ofrece. Dicho registro consta de tres partes.

1. Las páginas blancas describen a la empresa que ofrece el servicio: nombre, dirección, contactos, etc.
2. Las páginas amarillas incluyen a los servicios en categorías industriales basadas en clasificaciones estándar como el *North American Industry Classification System* y la *Standard Industrial Classification*.
3. Las páginas verdes describen la interfaz del servicio con suficiente detalle para quienes quieran escribir una aplicación que utilice el servicio Web. Los servicios se definen mediante un documento UDDI llamado *ModelType* o *tModel*. En muchos casos, el *tModel* contiene un archivo WSDL que describe la interfaz para un servicio Web.

Toda esta información se almacena en un registro (UBR, UDDI *business registry*), al cual irán todos los clientes a consultar los servicios que necesitan, y obtener un documento WSDL para poder comunicarse con el servicio solicitado. La forma de almacenar la información también se basa en XML.

Antes de UDDI, no había ninguna norma de internet para que las empresas llegaran a sus clientes y socios con información sobre sus productos y servicios. Tampoco había un método de cómo integrarlos en sus respectivos sistemas y procesos.

En un primer momento se creó un repositorio (UBR- UDDI *business registry*) universal donde se almacenaría la información referente a las distintas organizaciones de forma clasificada y basándose en XML. Una vez establecido el repositorio y su formato las empresas empiezan a registrar sus servicios y la información relevante de las mismas en él. A este repositorio accederán las distintas organizaciones que busquen una serie de servicios

de otras empresas, obteniendo principalmente en sus accesos una serie de documentos (WSDL), en los que viene el “interfaz” en XML, de los servicios publicados por las empresas. Con este documento las organizaciones pueden establecer en tiempo de ejecución sus mensajes (SOAP) que serán enviados a la organización que provee el servicio y estos mensajes se ciñen a la estructura marcada por el documento WSDL obtenido anteriormente. De esta forma se ejecuta el ciclo para el intercambio de información entre las organizaciones interesadas.

El siguiente gráfico (Figura 2.2) ilustra todo este proceso:

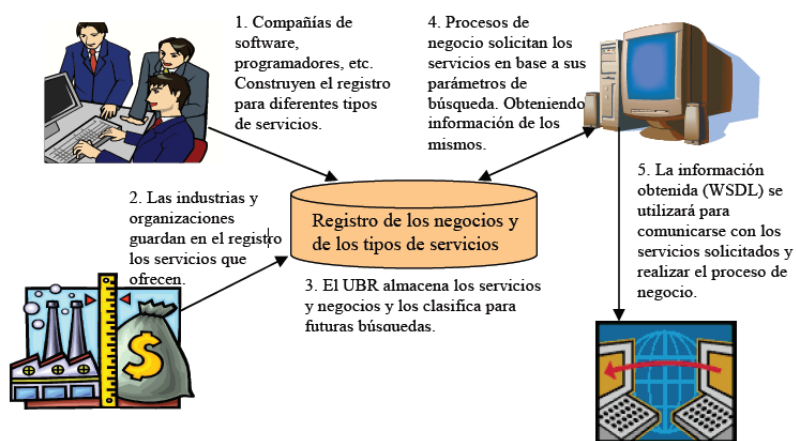


Figura 2.2. Proceso de publicación y obtención de servicios mediante UDDI (UBR).

En la actualidad los Servicios Web, permiten un alto grado de integración en las aplicaciones. Esta integración puede tener varios niveles, desde un nivel sencillo de integración, denominado servicios básicos o “e-servicios”, en el que las aplicaciones se integran con Servicios Web. Hasta un nivel complejo de integración, denominado servicios compuestos o “e-workflow”, en el que la integración se produce en servicios compuestos de varios servicios sencillos (varios “e-services” forman el “e-workflow”).

Los e-servicios o servicios básicos están basados en las tecnologías anteriormente mencionadas (XML, SOAP, WSDL, UDDI), mientras que los “e-workflow” (*procesos de negocio*), al estar compuestos por varios *e-services*, también se basarán en estas tecnologías pero además, tendrán que tener mecanismos para controlar que todo el proceso de negocio discorra por los cauces adecuados, es decir mecanismos que coordinen todo el *proceso de negocio*.

2.2.4. Interoperabilidad

La Interoperabilidad [Harder 1999], [Goodchild 1977], [Albrecht 1999] es la habilidad de dos sistemas de intercambiar información, del proceso e interpretación correcta de esta información requiere un grado de compatibilidad entre los sistemas, para permitir el intercambio y la interpretación correcta de los datos. Idealmente, los sistemas cooperativos deben apegarse a un estándar de dominio de aplicación y computacional. Sin embargo, este nivel de estandarización puede ser imposible de lograr en la práctica, debido a la proporción de cambios tecnológicos, la falta de estándares universalmente aceptados, la existencia de sistemas legados, o simplemente por razones de autonomía de cada sistema de información. Así, en muchos casos, la única manera de alcanzar la interoperabilidad es

haciendo publicas las interfaces, esquemas y formatos usados para el intercambio de información, haciendo su semántica tan explícita como sea posible, para que así puedan ser manejados apropiadamente por los sistemas cooperativos.

2.2.4.1. Definiciones

Dentro de la arquitectura SOA para la implementación de Servicios Web, la interoperabilidad es tal vez el principio más importante. SOA debe ofrecer importantes beneficios de interoperabilidad, y permitir la ejecución de servicios Web distribuidos en múltiples plataformas de software y arquitecturas de hardware.

La Organización para la Interoperabilidad de Servicios Web (*Web Services Interoperability Organization*), tiene por objetivo fomentar y promover la Interoperabilidad de Servicios Web (*Web Services Interoperability -WS-I*) sobre cualquier plataforma, sobre aplicaciones, y sobre lenguajes de programación. Su intención es ser un integrador de estándares para ayudar al avance de los servicios web de una manera estructurada y coherente. La WS-I [WS-I 2004], ha organizado los estándares que afectan a la interoperabilidad de los Servicios Web en una pila basada en funcionalidades.

Hay varios estándares que necesitan ser coordinados para llevar a buen término las cuestiones de interoperabilidad de servicios. Dichos estándares se están desarrollando en paralelo y de manera independiente. Para superar estas cuestiones, la WS-I ha desarrollado el concepto de "perfil" (*profile*). La WS-I define un perfil como:

Un conjunto de definiciones/especificaciones comúnmente aceptadas por la industria y a partir del apoyo a estándares basados en XML, junto con un conjunto de indicaciones que recomiendan cómo se deben usar las especificaciones para desarrollar servicios web interoperables entre sí.

Dado que dicha organización, de carácter abierto, está compuesta por las principales compañías de desarrollo de software, como IBM, Microsoft y Sun Microsystems, hay un claro compromiso por incluir todos estos estándares en el mundo de la programación actual y futura.

Los beneficios que se obtienen de la utilización de los Servicios Web, y con ello de la arquitectura SOA son:

- Constituyen un mecanismo de reutilización de componentes software, realizando una funcionalidad o actividad de negocio, que se ofrece a las aplicaciones que los invocan a través de una interfaz bien definida. Añadiendo además flexibilidad y reduciendo el esfuerzo de mantenimiento de las aplicaciones.
- Pueden registrarse y publicarse en la Web, haciendo uso de protocolos y estándares abiertos.
- La utilización de los Servicios Web constituye una forma de estructurar las aplicaciones en componentes de negocio, garantizando la transparencia en la localización de los servicios

Como ya se dijo la interoperabilidad es la capacidad de dos o más sistemas o componentes para intercambiar la información. A partir de esto existen diferentes formas de clasificar la interoperabilidad. Por ejemplo:

1. Dado que la web es una red altamente distribuida donde cada vez más se encuentran, por una parte, sistemas de información heterogénea y, por otra, usuarios que demandan un completo acceso a la información disponible; se clasifica con base en distintos niveles:
 - a. Infraestructura
 - b. Sintaxis
 - c. Estructura
 - d. Semántica
2. Para el caso de las arquitecturas basadas en servicios, se clasifica en:
 - a. Aplicación y herramienta
 - b. Contenido
 - c. Infraestructura
 - d. Usuario.
3. Para los sistemas que apoyan la educación, se clasifica principalmente en dos:
 - a. Infraestructura
 - b. Contenido

2.2.4.2. XML

XML (*eXtensible Markup Language*) [XML 1999], [Dikes 2005], [XML 2005], [Amiano 2006] es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que tiene un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Ejemplo de documento XML:

```
<?xml versión ="1.0" encoding="ISO-8859-1"?>
<libro>
  <titulo></titulo>
  <Capítulo>
    <titulo></titulo>
    <sección>
      <titulo></titulo>
    </sección>
  </Capítulo>
</libro>
```

El desarrollo de XML comenzó 1996 y ha sido una recomendación de la W3C desde febrero de 1998, lo cual le podría hacer sospechar que ésta es una tecnología bastante inmadura. De hecho, la tecnología no es muy nueva. Antes de XML estuvo SGML, desarrollado a principios de los '80, estándar ISO desde 1986, y ampliamente usado para grandes proyectos de documentación. El desarrollo de HTML empezó en 1990. Los diseñadores de XML simplemente tomaron las mejores partes de SGML, guiados por la experiencia con HTML, y produjeron algo que es no menos poderoso que SGML, y vastamente más regular y simple de usar. Algunas evoluciones, sin embargo, son difíciles

de distinguir de revoluciones y hay que decir que mientras SGML es mayormente usado para documentación técnica y mucho menos para otras clases de datos, con XML pasa exactamente lo opuesto.

Hay una importante aplicación de XML que es un formato de documento: XHTML de la W3C, el sucesor de HTML. XHTML tiene muchos de los mismos elementos de HTML. La sintaxis ha sido ligeramente cambiada para conformarse a las reglas de XML. Un documento "basado en XML" hereda la sintaxis de XML y la restringe de ciertas maneras (por ejemplo, XHTML permite "<p>", pero no "<r>"); también suma significado a esa sintaxis (XHTML dice que "<p>" significa "párrafo", y no "precio", "persona", o cualquier otra cosa).

XML le permite definir un formato de documento combinando y reutilizando otros formatos. Puesto que dos formatos desarrollados independientemente podrían tener elementos o atributos con el mismo nombre, se debe tener cuidado al combinarlos (¿"<p>" significa "párrafo" de este formato o "persona" de aquél otro?). Para eliminar la confusión de nombres al combinar formatos, XML provee un mecanismo de espacio de nombre. XSL (*Extensible Style Language*) y RDF (*Resource Description Framework*) son buenos ejemplos de formatos basados en XML que usan espacios de nombres. XML Schema está diseñado para reflejar este soporte de la modularidad al nivel de definir la estructura de documentos XML, por medio de la facilidad para combinar dos esquemas para producir un tercero que cubre una estructura de documento combinada.

El RDF de la W3C es un formato de texto XML que soporta aplicaciones de descripción de recursos y metadatos, tales como listas de temas musicales, colecciones de fotos, y bibliografías. Por ejemplo, RDF podría permitirle identificar las personas en un álbum de fotos Web usando información de una lista de contactos personales; entonces, su cliente de correo podría enviar automáticamente a esas personas un mensaje diciendo que sus fotos están en la Web. Lo mismo que HTML integró los documentos, los sistemas de menú y las aplicaciones de formularios para lanzar la Web original, RDF integra las aplicaciones y los agentes en una Web Semántica. Del mismo modo que las personas necesitan estar de acuerdo en los significados de las palabras que emplean en su comunicación, las computadoras necesitan mecanismos para acordar los significados de los términos para comunicarse efectivamente. Las descripciones formales de los términos en una cierta área (compras o manufactura, por ejemplo) se llaman ontologías, y son una parte necesaria de la Web Semántica. RDF, las ontologías, y la representación del significado de modo que las computadoras puedan ayudar a las personas a hacer el trabajo, son tópicos de la actividad de la Web Semántica.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información.

Entre las tecnologías XML disponibles se pueden destacar:

- **XSL:** Lenguaje Extensible de Hojas de Estilo, cuyo objetivo principal es mostrar cómo debería estar estructurado el contenido, cómo debería ser diseñado el contenido de origen y cómo debería ser paginado en un medio de presentación como puede ser una ventana de un navegador Web o un dispositivo móvil, o un conjunto de páginas de un catálogo, informe o libro. XSL funciona como un lenguaje avanzado para crear hojas de estilos. Es capaz de transformar, ordenar y filtrar datos XML, y darles formato basándolo en sus valores.

- XPath: Lenguaje de Rutas XML, es un lenguaje para acceder a partes de un documento XML. XPath identifica partes de un documento XML concreto, como pueden ser sus atributos, elementos, etc.
- XLink: Lenguaje de Enlace XML, es un lenguaje que permite insertar elementos en documentos XML para crear enlaces entre recursos XML. XLink por su lado, describe un camino estándar para añadir hiperenlaces en un archivo XML. Es decir, es un mecanismo de vinculación a otros documentos XML. Funciona de forma similar a un enlace en una página Web, es decir, funciona como lo haría `<a href = "">`, sólo que href es un enlace unidireccional. Sin embargo, XLink permite crear vínculos bidireccionales, lo que implica la posibilidad de moverse en dos direcciones. Esto facilita la obtención de información remota como recursos en lugar de simplemente como páginas Web.
- XPointer: Lenguaje de Direccionamiento XML, es un lenguaje que permite el acceso a la estructura interna de un documento XML, esto es, a sus elementos, atributos y contenido. XPointer funciona como una sintaxis que apunta a ciertas partes de un documento XML, es como una extensión de XPath. Se utiliza para llegar a ciertas partes de un documento XML. Primero, XLink permite establece el enlace con el recurso XML y luego es XPointer el que va a un punto específico del documento. Su funcionamiento es muy similar al de los identificadores de fragmentos en un documento HTML ya que se añade al final de una URI y después lo que hace es encontrar el lugar especificado en el documento XML. Al ser XPointer una extensión de XPath, XPointer tiene todas las ventajas de XPath y además permite establecer un rango en un documento XML, es decir, con XPointer es posible establecer un punto final y un punto de inicio, lo que incluye todos los elementos XML dentro de esos dos puntos.
- XQL: Lenguaje de Consulta XML, es un lenguaje que facilita la extracción de datos desde documentos XML. Ofrece la posibilidad de realizar consultas flexibles para extraer datos de documentos XML en la Web. XQL, lenguaje de consultas, se basa en operadores de búsqueda de un modelo de datos para documentos XML que puede realizar consultas en infinidad de tipos de documentos como son documentos estructurados, colecciones de documentos, bases de datos, estructuras DOM, catálogos, etc.

2.3. Bases de datos

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos [Silberschatz 2002].

En esencia, un sistema de bases de datos no es más que un sistema para archivar datos en computador. La base de datos en si puede considerarse como una especie de archivero electrónico; dicho de otra manera, es un lugar donde se almacena un conjunto de archivos de datos computarizados [Date 1993].

2.3.1. Modelo Entidad-Relación

Existen tres conceptos básicos que emplea el modelo de datos Entidad-Relación (E-R): conjuntos de entidades, conjuntos de relaciones y atributos.

1. Conjuntos de entidades: Podemos decir inicialmente que una entidad es una «cosa» u «objeto» en el mundo real que es distinguible de todos los demás objetos. Una entidad puede ser concreta, como una persona o un libro, o puede ser abstracta, como un préstamo, unas vacaciones o un concepto. A partir de esto decimos que un conjunto de entidades es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos. Las entidades individuales que constituyen un conjunto se llaman la *extensión* del conjunto de entidades.
2. Conjuntos de relaciones: Una relación es una asociación entre diferentes entidades. Un conjunto de relaciones es un conjunto de relaciones del mismo tipo. Formalmente es una relación matemática con $n \geq 2$ de conjuntos de entidades (posiblemente no distintos). Si $E_1, E_2, \dots, E_n$ son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de:

$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

donde $(e_1, e_2, \dots, e_n)$ es una relación.

3. Atributos: Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. La designación de un atributo para un conjunto de entidades expresa que la base de datos almacena información similar concerniente a cada entidad del conjunto de entidades; sin embargo, cada entidad puede tener su propio valor para cada atributo. Para cada atributo hay un conjunto de valores permitidos, llamados el dominio, o el conjunto de valores, de ese atributo. Un atributo, como se usa en el modelo E-R, se puede caracterizar por los siguientes tipos de atributo.
 - Atributos simples y compuestos. Los atributos simples son aquellos que no están divididos en subpartes. Los atributos compuestos, en cambio, se pueden dividir en subpartes (es decir, en otros atributos). Usar atributos compuestos en un esquema de diseño es una buena elección si el usuario desea referirse a un atributo completo en algunas ocasiones y, en otras, a algún componente del atributo.
 - Atributos monovalorados y multivalorados. Los atributos monovalorados son aquellos en los cuales solo existe un solo valor para una entidad específica. Los atributos multivalorados como su nombre lo indica, son aquellos que pueden tener cero, uno o más valores.
 - Atributos derivados. El valor para este tipo de atributo se puede derivar de los valores de otros atributos o entidades relacionados.

Un esquema de desarrollo E-R puede definir ciertas restricciones a las que los contenidos de la base de datos se deben adaptar.

La correspondencia de cardinalidades, o razón de cardinalidad, expresa el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones. La correspondencia de cardinalidades es la más útil describiendo conjuntos de relaciones binarias, aunque ocasionalmente contribuye a la descripción de conjuntos de relaciones que implican más de dos conjuntos de entidades. Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B , la correspondencia de cardinalidades debe ser una de las siguientes:

- Uno a uno. Una entidad en A se asocia con *a lo sumo* una entidad en B , y una entidad en B se asocia con *a lo sumo* una entidad en A .
- Uno a varios. Una entidad en A se asocia con cualquier número de entidades en B (ninguna o varias). Una entidad en B , sin embargo, se puede asociar con *a lo sumo* una entidad en A .
- Varios a uno. Una entidad en A se asocia con *a lo sumo* una entidad en B . Una entidad en B , sin embargo, se puede asociar con cualquier número de entidades (ninguna o varias) en A .
- Varios a varios. Una entidad en A se asocia con cualquier número de entidades (ninguna o varias) en B , y una entidad en B se asocia con cualquier número de entidades (ninguna o varias) en A .

La correspondencia de cardinalidades apropiada para un conjunto de relaciones particular depende obviamente de la situación del mundo real que el conjunto de relaciones modela.

La participación de un conjunto de entidades E en un conjunto de relaciones R se dice que es total si cada entidad en E participa al menos en una relación en R .

Si sólo algunas entidades en E participan en relaciones en R , la participación del conjunto de entidades E en la relación R se llama parcial.

Conceptualmente las entidades y relaciones individuales son distintas; desde una perspectiva de bases de datos, sin embargo, la diferencia entre ellas se debe expresar en término de sus atributos. Por lo tanto, los valores de los atributos de una entidad deben ser tales que permitan *identificar unívocamente* a la entidad. En otras palabras, no se permite que ningún par de entidades tengan exactamente los mismos valores de sus atributos.

Una clave permite identificar un conjunto de atributos lo suficiente para distinguir las entidades entre sí. Las claves también ayudan a identificar unívocamente a las relaciones y así a distinguir las relaciones entre sí.

Una superclave es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en el conjunto de entidades. Una superclave puede contener atributos innecesarios. Si K es una superclave, entonces también lo es cualquier superconjunto de K . A menudo interesan las superclaves tales que los subconjuntos propios de ellas no son superclave. Tales superclaves mínimas se llaman claves candidatas.

Una clave primaria es aquella que denota una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades. Una clave (primaria, candidata y superclave) es una propiedad del conjunto de entidades, más que de las entidades individuales. Cualesquiera dos entidades individuales en el conjunto no pueden tener el mismo valor en sus atributos clave

al mismo tiempo. La designación de una clave representa una restricción en el desarrollo del mundo real que se modela. La clave primaria se debería elegir de manera que sus atributos nunca, o muy raramente, cambien.

Las nociones de conjunto de entidades y conjunto de relaciones no son precisas, y es posible definir un conjunto de entidades y las relaciones entre ellas de diferentes formas.

Un error común es usar la clave primaria de un conjunto de entidades como un atributo de otro conjunto de entidades, en lugar de usar una relación. Otro error relacionado que se comete es designar a los atributos de la clave primaria de los conjuntos de entidades relacionados como atributos del conjunto de relaciones. Esto no se debería hacer, ya que los atributos de la clave primaria son ya implícitos en la relación.

No siempre está claro si es mejor expresar un objeto mediante un conjunto de entidades o mediante un conjunto de relaciones. Una posible guía para determinar si usar un conjunto de entidades o un conjunto de relaciones es designar un conjunto de relaciones para describir una acción que ocurre entre entidades. Este enfoque puede también ser útil para decidir si ciertos atributos se pueden expresar más apropiadamente como relaciones. Las relaciones en las bases de datos son generalmente binarias. Algunas relaciones que parecen no ser binarias podrían ser representadas mejor con varias relaciones binarias. De hecho, siempre es posible reemplazar un conjunto de relaciones no binarias (n -aria, para $n > 2$) por un número de diferentes conjuntos de relaciones binarias.

La razón de cardinalidad de una relación puede afectar a la situación de los atributos de la relación. Los atributos de los conjuntos de relaciones uno a uno o uno a varios pueden estar asociados con uno de los conjuntos de entidades participantes, en lugar de con el conjunto de relaciones. Los atributos de un conjunto de relaciones uno a varios se pueden colocar sólo en el conjunto de entidades de la parte «varios» de la relación. Por otra parte, para los conjuntos de entidades uno a uno, los atributos de la relación se pueden asociar con cualquiera de las entidades participantes. La elección de la colocación del atributo es más clara para los conjuntos de relaciones varios a varios.

Un conjunto de entidades puede no tener suficientes atributos para formar una clave primaria. Tal conjunto de entidades se denomina conjunto de entidades débiles. Un conjunto de entidades que tiene una clave primaria se denomina conjunto de entidades fuertes. Para que un conjunto de entidades débiles tenga sentido, debe estar asociada con otro conjunto de entidades, denominado el conjunto de entidades identificadoras propietarias. Cada entidad débil debe estar asociada con una entidad identificadora; es decir, se dice que el conjunto de entidades débiles depende existencialmente del conjunto de entidades identificadoras. Se dice que el conjunto de entidades identificadoras es propietaria del conjunto de entidades débiles que identifica. La relación que asocia el conjunto de entidades débiles con el conjunto de entidades identificadoras se denomina relación identificadora. La relación identificadora es varios a uno del conjunto de entidades débiles al conjunto de entidades identificadoras y la participación del conjunto de entidades débiles en la relación es total.

Aunque un conjunto de entidades débiles no tiene clave primaria, no obstante se necesita conocer un medio para distinguir todas aquellas entidades del conjunto de entidades que dependen de una entidad fuerte particular. El discriminante de un conjunto de entidades débiles es un conjunto de atributos que permite que esta distinción se haga. La clave primaria de un conjunto de entidades débiles se forma con la clave primaria del conjunto de

entidades identificadoras, más el discriminante del conjunto de entidades débiles. El conjunto de entidades identificadoras no debería tener atributos descriptivos, ya que cualquier atributo requerido puede estar asociado con el conjunto de entidades débiles.

Un conjunto de entidades débiles puede participar en relaciones distintas de relaciones identificadoras. Un conjunto de entidades débiles puede participar como propietario en una relación identificadora con otro conjunto de entidades débiles. También es posible tener un conjunto de entidades débiles con más de un conjunto de entidades identificadoras. Una entidad débil en concreto podría ser identificada por una combinación de entidades, una de cada conjunto de entidades identificadoras. La clave primaria de la entidad débil consistiría de la unión de las claves primarias de los conjuntos de entidades identificadoras más el discriminante del conjunto de entidades débiles.

Un conjunto de entidades débiles se indica en los diagramas E-R mediante un rectángulo dibujado con una línea doble y la correspondiente relación de identificación mediante un rombo dibujado con línea doble. Un conjunto de entidades débiles se puede modelar más adecuadamente como un atributo si sólo participa en la relación identificadora y si tiene pocos atributos. Alternativamente, una representación de conjunto de entidades débiles será más adecuada para modelar una situación en la que el conjunto participe en otras relaciones además de la relación identificadora y donde el conjunto de entidades débiles tenga muchos atributos.

El refinamiento a partir de un conjunto de entidades inicial en sucesivos niveles de subgrupos de entidades representa un proceso de diseño descendente en el que las distinciones se hacen explícitas. El proceso de diseño puede ser también de una forma ascendente, en el que varios conjuntos de entidades se sintetizan en un conjunto de entidades de nivel más alto basado en características comunes. La generalización, es una relación contenedora que existe entre el conjunto de entidades de nivel más alto y uno o más conjuntos de entidades de nivel más bajo. Los conjuntos de entidades de nivel más alto y nivel más bajo también se pueden llamar superclase y subclase, respectivamente. La generalización procede de observar que varios conjuntos de entidades que comparten algunas características comunes (se describen mediante los mismos atributos y participan en los mismos conjuntos de relaciones). Basada en sus similitudes, la generalización sintetiza estos conjuntos de entidades en uno solo, el conjunto de entidades de nivel más alto. La generalización se usa para resaltar las similitudes entre los conjuntos de entidades de nivel más bajo y para ocultar las diferencias; también permite economizar la representación para que los atributos compartidos no estén repetidos.

Una limitación del modelo E-R es que no resulta posible expresar relaciones entre relaciones. La agregación es una abstracción a través de la cual las relaciones se tratan como entidades de nivel más alto.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama E-R. Los diagramas son simples y claros, tal diagrama consta de los siguientes componentes principales (ver Figura 2.3):

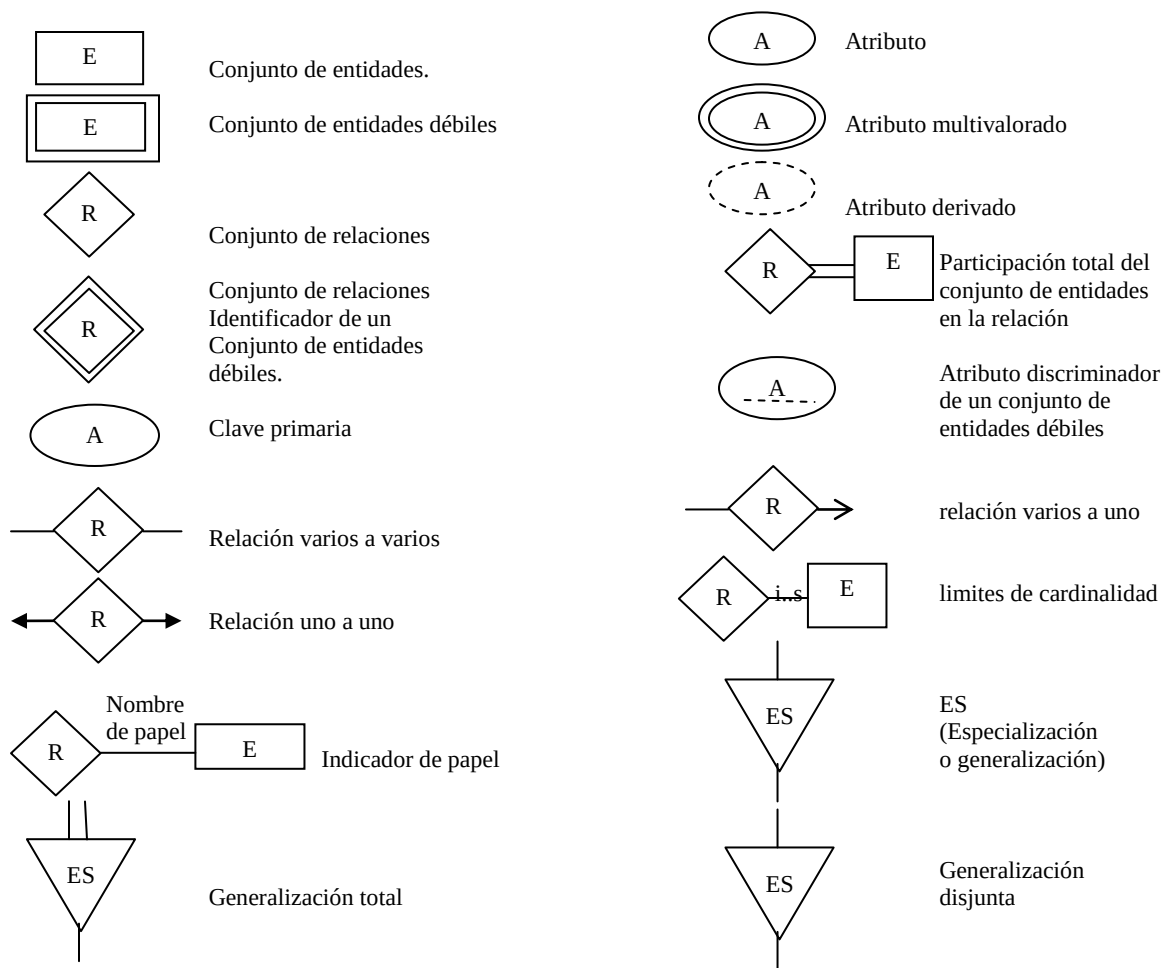


Figura 2.3. Símbolos usados en la notación E-R.

Considérese un diagrama entidad-relación, que consta de dos conjuntos de entidades, *cliente* y *préstamo*, relacionadas a través de un conjunto de relaciones binarias *prestatario* (ver Figura 2.4). Los atributos asociados con *cliente* son *id-cliente*, *nombre-cliente*, *calle-cliente*, y *ciudad-cliente*. Los atributos asociados con *préstamo* son *número-préstamo* e *importe*. Como se muestra en la figura, los atributos de un conjunto de entidades que son miembros de la clave primaria están subrayados. El conjunto de relaciones *prestatario* puede ser varios a varios, uno a varios, varios a uno o uno a uno. Para distinguir entre estos tipos, se dibuja o una línea dirigida ($\rightarrow$) o una línea no dirigida ($—$) entre el conjunto de relaciones y el conjunto de entidades en cuestión.

Una línea dirigida desde el conjunto de relaciones *prestatario* al conjunto de entidades *préstamo* especifica que *prestatario* es un conjunto de relaciones uno a uno, o bien varios a uno, desde *cliente* a *préstamo*; *prestatario* no puede ser un conjunto de relaciones varios a varios ni uno a varios, desde *cliente* a *préstamo*. Una línea no dirigida desde el conjunto de relaciones *prestatario* al conjunto de relaciones *préstamo* especifica que *prestatario* es o bien un conjunto de relaciones varios a varios, o bien uno a varios, desde *cliente* a *préstamo*.

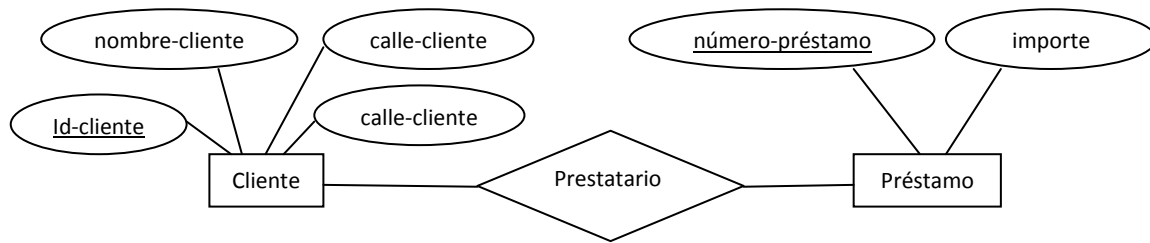


Figura 2.4. Diagrama E-R correspondiente a clientes y préstamos.

El modelo de datos E-R da una flexibilidad sustancial en el diseño de un esquema de bases de datos. Entre las decisiones que se toman para la modelación están las siguientes:

- Si se usa un atributo o un conjunto de entidades para representar un objeto.
- Si un concepto del mundo real se expresa más exactamente mediante un conjunto de entidades o mediante un conjunto de relaciones.
- Si se usa una relación ternaria o un par de relaciones binarias.
- Si se usa un conjunto de entidades fuertes o débiles; un conjunto de entidades fuertes y sus conjuntos de entidades débiles dependientes se pueden considerar como un «objeto» en la base de datos, debido a que la existencia de las entidades débiles depende de la entidad fuerte.
- Si el uso de la generalización es apropiado; la generalización, o una jerarquía de relaciones ES, contribuye a la modularidad por permitir que los atributos comunes de conjuntos de entidades similares se representen en un único lugar en un diagrama E-R.
- Si el uso de la agregación es apropiado; la agregación agrupa una parte de un diagrama E-R en un único conjunto de entidades, permitiendo tratar el conjunto de entidades de la agregación como una unidad única sin importar los detalles de su estructura interna.

La fase inicial del diseño de bases de datos, es caracterizar completamente las necesidades de datos esperadas por los usuarios de la base de datos. El resultado de esta fase es una *especificación de requisitos del usuario*. El esquema desarrollado en esta fase de diseño conceptual proporciona una visión detallada del desarrollo. En términos del modelo E-R, el esquema especifica todos los conjuntos de entidades, conjuntos de relaciones, atributos y restricciones de correspondencia. También se examina el diseño para eliminar características redundantes. Lo importante en este punto es describir los datos y las relaciones, más que especificar detalles del almacenamiento físico. Un esquema conceptual completamente desarrollado indicará también los requisitos funcionales. En una especificación de requisitos funcionales los usuarios describen los tipos de operaciones (o transacciones) que se realizarán sobre los datos. Algunos ejemplos de operaciones son la modificación o actualización de datos, la búsqueda y recuperación de datos específicos y el borrado de datos.

Un esquema conceptual completamente desarrollado indicará también los requisitos funcionales de la empresa. En una especificación de requisitos funcionales los usuarios describen los tipos de operaciones (o transacciones) que se realizarán sobre los datos. Algunos ejemplos de operaciones son la modificación o actualización de datos, la búsqueda

y recuperación de datos específicos y el borrado de datos. En esta fase de diseño conceptual se puede hacer una revisión del esquema para encontrar los requisitos funcionales.

El proceso de trasladar un modelo abstracto de datos a la implementación de la base de datos consta de dos fases de diseño finales.

En la fase de diseño lógico, el diseñador traduce el esquema conceptual de alto nivel al modelo de datos de la implementación del sistema de base de datos que se usará. El diseñador usa el esquema resultante específico a la base de datos en la siguiente fase de diseño físico, en la que se especifican las características físicas de la base de datos. Estas características incluyen la forma de organización de los archivos y las estructuras de almacenamiento interno.

La especificación inicial de los requisitos de usuario se puede basar en entrevistas con los usuarios de la base de datos y en el análisis propio del diseñador del desarrollo.

La descripción que surge de esta fase de diseño sirve como base para especificar la estructura conceptual de la base de datos. La especificación de los requisitos de datos sirve como punto de partida para la construcción de un esquema conceptual para la base de datos.

Una base de datos que se ajusta a un esquema de bases de datos E-R se puede representar por una colección de tablas. Para cada conjunto de entidades de la base de datos y para cada conjunto de relaciones de la base de datos hay una única tabla a la que se asigna el nombre del conjunto de entidades o del conjunto de relaciones correspondiente. Cada tabla tiene varias columnas, cada una de las cuales tiene un nombre único.

Los modelos E-R y el de bases de datos relacionales son representaciones abstractas y lógicas de empresas del mundo real. Debido a que los dos modelos emplean principios de diseño similares, se puede convertir un diseño E-R en un diseño relacional. Convertir una representación de bases de datos de un diagrama E-R a un formato de tablas es la base para la derivación de un diseño de bases de datos relacional desde un diagrama E-R. Aunque existen diferencias importantes entre una relación y una tabla, una relación se puede considerar informalmente como una tabla de valores.

Las restricciones especificadas en un diagrama E-R, tales como las claves primarias y las restricciones de cardinalidad, se corresponden con restricciones sobre las tablas generadas a partir del diagrama E-R.

A continuación se da un ejemplo de cómo se puede representar un esquema E-R mediante tablas.

Sea E un conjunto de entidades fuertes con los atributos descriptivos $a_1, a_2, \dots, a_n$. Esta entidad se representa mediante una tabla llamada E con n columnas distintas, cada una de las cuales corresponde a uno de los atributos de E . Cada fila de la tabla corresponde a una entidad del conjunto de entidades E . Como ilustración considérese el conjunto de entidades *préstamo* del diagrama E-R mostrado en la Figura 2.x. Este conjunto de entidades tiene dos atributos: *número-préstamo* e *importe*. Se representa este conjunto de entidades mediante una tabla llamada *préstamo*, con dos columnas, como se muestra en la Figura 2.5. La fila

(P-17,1.000)

de la tabla *préstamo* significa que el número de préstamo P-17 tiene un importe de préstamo de 1.000 €. Se puede añadir una nueva entidad a la base de datos insertando una fila en una tabla. También se pueden borrar o modificar las filas. D_1 denota el conjunto de todos los números de préstamo y D_2 denota el conjunto de todos los saldos. Cualquier fila de la tabla *préstamo* debe consistir en una tupla (v_1, v_2) , donde v_1 es un número de préstamo (es decir, v_1 está en el conjunto D_1) y v_2 es un importe (es decir, v_2 está en el conjunto D_2). En general, la tabla *préstamo* contendrá sólo un subconjunto del conjunto de todas las filas

posibles. El conjunto de todas las filas posibles de *préstamo* es el *producto cartesiano* de D_1 y D_2 , denotado por $D_1 \times D_2$. En general, si se tiene una tabla de n columnas, se denota el producto cartesiano de $D_1, D_2, \dots, D_n$ por

$$D_1 \times D_2 \times \dots \times D_{n-1} \times D_n$$

número-préstamo	importe
P-11	900
p-14	1.500
p-15	1.500
p-16	1.300
p-17	1.000
p-23	2.000
p-93	500

Figura 2.5. La tabla préstamo.

2.3.2. Modelo Relacional

El Modelo Relacional es una forma de representar los datos mediante tablas y la forma para manipular estos datos mediante operadores [Date 1993].

El Modelo Relacional se ha establecido como el principal modelo de datos para las aplicaciones de procesamiento de datos, esto debido a su simplicidad, que facilita el trabajo en comparación con otros modelos anteriores como el de red y el jerárquico.

2.3.2.1. Estructura de las bases de datos relacionales

Una base de datos relacional consiste en un conjunto de tablas, a cada una de las cuales se le asigna un nombre exclusivo. Cada fila de la tabla representa una relación entre un conjunto de valores. Dado que cada tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación, del que toma su nombre el modelo de datos relacional.

Considérese la tabla *cuenta* de la Figura 2.6, la cual tiene tres cabeceras de columna: *número-cuenta*, *nombre-sucursal* y *saldo*. Siguiendo la terminología del modelo relacional se puede hacer referencia a estas cabeceras como. Para cada atributo hay un conjunto de valores permitidos, llamado dominio de ese atributo. Para el atributo *nombre-sucursal*, por ejemplo, el dominio es el conjunto de los nombres de las sucursales. Supongamos que D_1 denota el conjunto de todos los números de cuenta, D_2 el conjunto de todos los nombres de sucursal y D_3 el conjunto de los saldos. Todas las filas de *cuenta* deben consistir en una tupla (v_1, v_2, v_3) , donde v_1 es un número de cuenta (es decir, v_1 está en el dominio D_1), v_2 es un nombre de sucursal (es decir, v_2 está en el dominio D_2) y v_3 es un saldo (es decir, v_3 está en el dominio D_3). En general, *cuenta* sólo contendrá un subconjunto del conjunto de todas las filas posibles. Por tanto, *cuenta* es un subconjunto de:

$$D_1 \times D_2 \times D_3$$

En general, una tabla de n atributos debe ser un subconjunto de:

$$D_1 \times D_2 \times \dots \times D_{n-1} \times D_n$$

Los matemáticos definen las relaciones como subconjuntos del producto cartesiano de la lista de dominios. Esta definición se corresponde de manera casi exacta con la definición de

tabla dada anteriormente. La única diferencia es que aquí se han asignado nombres a los atributos, mientras que los matemáticos sólo utilizan «nombres» numéricos, utilizando el entero 1 para denotar el atributo cuyo dominio aparece en primer lugar en la lista de dominios, 2 para el atributo cuyo dominio aparece en segundo lugar, etcétera. Como las tablas son esencialmente relaciones, se utilizarán los términos matemáticos relación y tupla en lugar de los términos tabla y fila. Una variable tupla es una variable que representa a una tupla; en otras palabras, una tupla que representa al conjunto de todas las tuplas.

numero-cuenta	nombre-sucursal	saldo
C-101	Centro	500
C-102	Navacerrada	400
C-201	Galapagar	900
C-215	Becerril	700
C-217	Galapagar	750
C-222	Moralzarzal	700
C-305	Collado Mediano	350

Figura 2.6. La relación cuenta.

En la relación *cuenta* de la Figura 2.x hay siete tuplas. Supóngase que la variable tupla t hace referencia a la primera tupla de la relación. Se utiliza la notación $t[\text{número-cuenta}]$ para denotar el valor de t en el atributo *número-cuenta*. Por tanto, $t[\text{número-cuenta}] = \text{«C-101»}$ y $t[\text{nombre-sucursal}] = \text{«Centro»}$. De manera alternativa, se puede escribir $t[1]$ para denotar el valor de la tupla t en el primer atributo (*número-cuenta*), $t[2]$ para denotar *nombre-sucursal*, etcétera.

Dado que las relaciones son conjuntos se utiliza la notación matemática $t \in r$ para denotar que la tupla t está en la relación r . El orden en que aparecen las tuplas es irrelevante, dado que una relación es un *conjunto* de tuplas. Se exigirá que, para todas las relaciones r , los dominios de todos los atributos de r sean atómicos.

Un dominio es atómico si los elementos del dominio se consideran unidades indivisibles. Por ejemplo, el conjunto de los enteros es un dominio atómico, pero el conjunto de todos los conjuntos de enteros es un dominio no atómico. La diferencia es que no se suele considerar que los enteros tengan subpartes, pero sí se considera que los conjuntos de enteros las tienen; por ejemplo, los enteros que forman cada conjunto. Lo importante no es lo que sea el propio dominio, sino la manera en que se utilizan los elementos del dominio en la base de datos.

Un valor de dominio que es miembro de todos los dominios posibles es el valor nulo, que indica que el valor es desconocido o no existe.

Cuando se habla de bases de datos se debe diferenciar entre el esquema de la base de datos, o diseño lógico de la misma, y el ejemplar de la base de datos, que es una instantánea de los datos de la misma en un momento dado. El concepto de relación se corresponde con el concepto de variable de los lenguajes de programación. El concepto de esquema de la relación se corresponde con el concepto de definición de tipos de los lenguajes de programación. Resulta conveniente dar un nombre a los esquemas de las relaciones, igual que se dan nombres a las definiciones de tipos en los lenguajes de programación. Se adopta el convenio de utilizar nombres en minúsculas para las relaciones y nombres que comiencen por una letra mayúscula para los esquemas de las relaciones.

Siguiendo esta notación se utilizará *Esquema-cuenta* para denotar el esquema de la relación de la relación *cuenta*. Por tanto,

Esquema-cuenta = (número-cuenta, nombre-sucursal, saldo)

Se denota el hecho de que *cuenta* es una relación de *Esquema-cuenta* mediante *cuenta* (*Esquema-cuenta*). En general, los esquemas de las relaciones incluyen una lista de los atributos y de sus dominios correspondientes.

El concepto de ejemplar de relación se corresponde con el concepto de valor de una variable en los lenguajes de programación. El valor de una variable dada puede cambiar con el tiempo; de manera parecida, el contenido del ejemplar de una relación puede cambiar con el tiempo cuando la relación se actualiza. Sin embargo, se suele decir simplemente «relación» cuando realmente se quiere decir «ejemplar de la relación».

Los conceptos de *superclave*, de *clave candidata* y de *clave primaria*, también son aplicables en el modelo relacional. Ejemplo: sea R el esquema de una relación. Si se dice que un subconjunto K de R es una *superclave* de R para las relaciones $r(R)$ en las que no hay dos tuplas diferentes que tengan los mismos valores en todos los atributos de K .

Es decir, si t_1 y t_2 están en R y $t_1 \neq t_2$, entonces $t_1[K] \neq t_2[K]$. Si el esquema de una base de datos relacional se basa en las tablas derivadas de un esquema E-R es posible determinar la clave primaria del esquema de una relación a partir de las claves primarias de los conjuntos de entidades o de relaciones de los que se deriva el esquema:

- Conjunto de entidades fuertes. La clave primaria del conjunto de entidades se convierte en la clave primaria de la relación.
- Conjunto de entidades débiles. La tabla y, por tanto, la relación correspondientes a un conjunto de entidades débiles incluyen:
 - a) Los atributos del conjunto de entidades débiles.
 - b) La clave primaria del conjunto de entidades fuertes del que depende el conjunto de entidades débiles.

La clave primaria de la relación consiste en la unión de la clave primaria del conjunto de entidades fuertes y el discriminante del conjunto de entidades débil.

- Conjunto de relaciones. La unión de las claves primarias de los conjuntos de entidades relacionados se transforma en una superclave de la relación. Si la relación es de varios a varios, esta superclave es también la clave primaria.

2.3.2.2. Álgebra Relacional

Un lenguaje de consulta es aquel con el cual los usuarios solicitan información a la base de datos, estos pueden ser procedimentales y no procedimentales.

En los lenguajes procedimentales el usuario le dice concretamente al sistema que hacer para que lleve a cabo una serie de operaciones en la base de datos para calcular el resultado deseado, mientras que en los lenguajes no procedimentales el usuario describe la información deseada sin dar un procedimiento concreto para obtener esa información.

El álgebra relacional es un lenguaje de consulta *procedimental*, el cual consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación.

Existen dos tipos de operaciones fundamentales sobre las cuales opera el álgebra relacional las cuales son: operaciones unarias y operaciones binarias.

Las operaciones *selección*, *proyección* y *renombramiento* se denominan operaciones unarias porque operan sobre una sola relación, a continuación se describen estas operaciones.

- La operación selección, selecciona tuplas que satisfacen una condición dada. Se utiliza la letra griega sigma minúscula (σ) para denotar la selección. La condición aparece como subíndice de σ . La relación del argumento se da entre paréntesis a continuación de σ . Por tanto, para seleccionar las tuplas de una relación R las cuales cumplan con cierta condición se escribe:

$$\sigma_{\text{nombre_atributo} = \text{«parametro»}}(R)$$

Esto se puede interpretar de la siguiente manera: *selecciona aquellas tuplas de la relación R en donde nombre_atributo sea igual a parametro*.

- La operación proyección devuelve solo aquellos atributos especificados. La proyección se denota por la letra griega mayúscula pi (Π). La lista de atributos seleccionados se colocan como subíndice de Π . La relación se escribe a continuación entre paréntesis. Por tanto para proyectar algunos atributos de una relación R hay que escribir:

$$\Pi_{\text{atributo1, atributo2, ..., atributon}}(R)$$

Esto se puede interpretar de la siguiente manera: *lista todas las filas de la relación R mostrando solo los atributos: atributo₁, atributo₂, ..., atributo_n*.

- La operación renombramiento, denotado por la letra griega rho minúscula (ρ), permite obtener una relación con un nombre definido por el usuario. La expresión $\rho_x(E)$ devuelve el resultado de la expresión E con el nombre x . Otra forma de la operación renombramiento es la siguiente. Supóngase que una expresión del álgebra relacional E tiene aridad n . Por tanto, la expresión

$$\rho_x(A_1, A_2, \dots, A_n)(E)$$

devuelve el resultado de la expresión E con el nombre x y con los atributos con el nombre cambiado a $A_1, A_2, \dots, A_n$.

Las operaciones *unión*, *diferencia de conjuntos*, *producto cartesiano* se denominan operaciones binarias, por que operan sobre pares de relaciones, a continuación se describen estas operaciones.

- La operación unión funciona de la misma forma que en la teoría de conjuntos, toma dos relaciones y las unifica, aunque hay que considerar que, para que una operación unión $R \cup S$ sea válida hay que exigir que se cumplan dos condiciones:
 1. Las relaciones r y s deben ser de la misma aridad, es decir, deben tener el mismo número de atributos.
 2. Los dominios de los atributos i -ésimos de R y de S deben ser iguales para todo i .

La sintaxis para esta operación es la siguiente:

$$\Pi_{\text{nombre-atributo}}(R) \cup \Pi_{\text{nombre-atributo}}(S)$$

- La operación diferencia de conjuntos, denotada por el símbolo $-$, permite buscar las tuplas que estén en una relación pero no en la otra. La expresión $R - S$ da como resultado una relación que contiene las tuplas que están en R pero no en S . La sintaxis para esta operación es la siguiente:

$$\Pi_{\text{nombre-atributo}}(R) - \Pi_{\text{nombre-atributo}}(S)$$

- La operación producto cartesiano, denotada por un aspa ($\times$), permite combinar información de cualesquiera dos relaciones. Por ejemplo, el producto cartesiano de las relaciones r_1 y r_2 como $r_1 \times r_2$. Sin embargo, dado que el mismo nombre de atributo puede aparecer tanto en r_1 como en r_2 , debe haber una forma para distinguir entre ambos atributos, lo cual se logra adjuntando al atributo el nombre de la relación de la que proviene originalmente. La sintaxis de esta operación es la siguiente:

$$\Pi_{\text{atributoC}}(\sigma_{R.\text{atributoP} = S.\text{atributoP}}(\sigma_{\text{atributoX} = \text{«parametro»}}(R \times S)))$$

La cual se puede interpretar como: *lista todas las filas resultado de la Selección de aquellas tuplas donde R.atributoP fue igual a S.atributoP de la selección resultante de $R \times S$ en donde AtributoX fue igual al parámetro dado, mostrando solo el atributo “atributoC”*

Las operaciones fundamentales del álgebra relacional son suficientes para expresar cualquier consulta del álgebra relacional. Sin embargo, si uno se limita únicamente a las operaciones fundamentales, algunas consultas resultan de expresión confusa y complicada. Por tanto, se definen otras que simplifican estas consultas, siendo que para cada operación nueva se facilita una expresión equivalente utilizando sólo las operaciones fundamentales.

- La intersección de conjuntos ($\cap$) puede sustituirse por un par de operaciones de diferencia de conjuntos:

$$r \cap s = r - (r - s)$$

Por tanto, esta operación no añade potencia al álgebra relacional, sencillamente, es más conveniente escribir $r \cap s$ que $r - (r - s)$.

- La *reunión natural* es una operación binaria que permite combinar ciertas selecciones y un producto cartesiano en una sola operación. Se denota por el símbolo de la «reunión». La operación reunión natural forma un producto cartesiano de sus dos argumentos, realiza una selección forzando la igualdad de los atributos que aparecen en ambos esquemas de relación y, finalmente, elimina los atributos duplicados. La reunión natural de r y de s , denotada por $r \bowtie s$ es una relación del esquema $R \cup S$ definida formalmente de la manera siguiente:

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{r.A1 = s.A1 \wedge r.A2 = s.A2 \wedge \dots \wedge r.An = s.An}(r \times s))$$

- La operación división, denotada por $\div$, resulta adecuada para las consultas que incluyen la expresión «para todos». Formalmente, sean $r(R)$ y $s(S)$ relaciones y $S \subseteq R$; es decir, todos los atributos del esquema S están también en el esquema R . La relación $r \div s$ es una relación del esquema $R - S$ (es decir, del esquema que contiene todos los atributos del esquema R que no están en el esquema S). Una tupla t está en $r \div s$ si y sólo si se cumplen estas dos condiciones:

1. t está en $\Pi_{R - S}(r)$
2. Para cada tupla t_S de s hay una tupla t_r de r que cumple las dos condiciones siguientes:
 - a) $t_r[S] = t_S[S]$
 - b) $t_r[R - S] = t$

Dados una operación división y los esquemas de las relaciones, se puede, de hecho, definir la operación división en términos de las operaciones fundamentales. Sean $r(R)$ y $s(S)$ dadas, con $S \subseteq R$:

$$r \div s = \Pi_{R - S}(r) - \Pi_{R - S}((\Pi_{R - S}(r) \times s) - \Pi_{R - S, S}(r))$$

- La operación asignación, denotada por $\leftarrow$, actúa de manera parecida a la asignación de los lenguajes de programación.

$temp1 \leftarrow \Pi_{R-S}(r)$

$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$

La evaluación de una asignación no hace que se muestre ninguna relación al usuario. Por el contrario, el resultado de la expresión a la derecha de $\leftarrow$ se asigna a la variable relación a la izquierda de $\leftarrow$. Esta variable relación puede utilizarse en expresiones posteriores:

$$resultado = temp1 - temp2$$

Las operaciones básicas del álgebra relacional se han ampliado de varias maneras. Una ampliación sencilla es permitir operaciones aritméticas como parte de la proyección. Una ampliación importante es permitir *operaciones de agregación*, como el cálculo de la suma de los elementos de un conjunto, o su media. Otra ampliación importante es la operación *reunión externa*, que permite a las expresiones del álgebra relacional trabajar con los valores nulos que modelan la información que falta. A continuación se describen estas operaciones.

- La operación proyección generalizada amplía la operación proyección permitiendo que se utilicen funciones aritméticas en la lista de proyección. La operación proyección generalizada tiene la forma

$$\Pi_{F1, F2, \dots, Fn}(E)$$

donde E es cualquier expresión del álgebra relacional y $F1, F2, \dots, Fn$ son expresiones aritméticas que incluyen constantes y atributos en el esquema de E . Como caso especial la expresión aritmética puede ser simplemente un atributo o una constante.

- Las funciones de agregación son funciones que toman una colección de valores y devuelven como resultado un único valor. Estas funciones pueden ser: *sum*, *avg*, *count*, *min*, *max*, etc. El símbolo $\mathcal{G}$ es la letra G en el tipo de letra caligráfico; se lee «G caligráfica». La operación del álgebra relacional $\mathcal{G}$ significa que se debe aplicar agregación, y el subíndice indica la operación de agregación a aplicar. La forma general de la operación de agregación G es la siguiente:

$$G_1, G_2, \dots, G_n \mathcal{G}_{F_1(A_1), F_2(A_2), \dots, F_m(A_m)}(E)$$

donde E es cualquier expresión del álgebra relacional; $G_1, G_2, \dots, G_n$ constituye una lista de atributos que indican cómo se realiza la agrupación, cada F_i es una función de agregación y cada A_i es el nombre de un atributo.

- La operación **reunión externa** es una ampliación de la operación reunión para trabajar con la información que falta. Esta operación tiene tres formas diferentes: *reunión externa por la derecha*, denotada por $\bowtie$ y *reunión externa completa*, denotada por $\Join$. Las tres formas de la reunión externa calculan la reunión y añaden tuplas adicionales al resultado de la misma.

1. La reunión externa por la izquierda denotada por $\ltimes$, toma todas las tuplas de la relación de la izquierda que no coincidan con ninguna tupla de la relación de la derecha, las rellena con valores nulos en todos los demás atributos de la relación de la derecha y las añade al resultado de la reunión natural.

2. La reunión externa por la derecha denotada por $\bowtie$, es simétrica de la reunión externa por la izquierda. Las tuplas de la relación de la derecha que no coincidan con ninguna tupla de la relación de la izquierda se rellenan con valores nulos y se añaden al resultado de la reunión natural.
3. La reunión externa completa denotada por $\Join$ realiza tanto la reunión externa por la izquierda como la reunión externa por la derecha, rellorando las tuplas de la relación de la izquierda que no coincidan con ninguna tupla de la relación de la derecha y las tuplas de la relación de la derecha que no coincidan con ninguna tupla de la relación de la izquierda, y añadiéndolas al resultado de la reunión.

Las modificaciones de la base de datos se expresan utilizando la operación de asignación ($\leftarrow$) que fue descrita anteriormente, estas operaciones se describen a continuación:

- La operación de inserción, agrega una nueva tupla en la relación, solo se debe tener en cuenta que la tupla a insertar sea de la misma aridad que la relación. En el álgebra relacional las inserciones se expresan mediante:

$$r \leftarrow r \cup E$$

donde r es una relación y E es una expresión del álgebra relacional.

- La operación de actualización, modifica ya sea todos o solo algunos valores de la tupla a la cual se hace referencia. Se puede utilizar el operador proyección generalizada para realizar esta tarea:

$$r \leftarrow \Pi_{F_1, F_2, \dots, F_n}(r)$$

donde cada F_i es el i -ésimo atributo de r , si el i -ésimo atributo no está actualizado, o, si hay que actualizar el atributo, una expresión, que sólo implica constantes y los atributos de r , que da el nuevo valor del atributo.

- La operación de borrado es semejante a una consulta solo que en vez de mostrar datos al usuario, elimina una o varias tuplas de la base de datos. En el álgebra relacional los borrados se expresan mediante:

$$r \leftarrow r - E$$

donde r es una relación y E es una consulta del álgebra relacional.

Las relaciones que no forman parte del modelo lógico pero se hacen visibles a los usuarios como relaciones virtuales se denominan vistas. Se puede trabajar con gran número de vistas sobre cualquier conjunto dado de relaciones reales. Las vistas se definen utilizando la instrucción create view.

Para definir una vista hay que darle un nombre e indicar la consulta que la va a calcular. La forma de la instrucción create view es

create view v as <expresión de consulta>

donde <expresión de consulta> es cualquier expresión legal de consulta del álgebra relacional. El nombre de la vista se representa mediante v .

Algunos sistemas de bases de datos permiten que se guarden las relaciones de vistas, pero se aseguran de que, si las relaciones reales utilizadas en la definición de la vista cambian, la vista se mantenga actualizada. Estas vistas se denominan vistas materializadas. El proceso de mantener actualizada la vista se denomina mantenimiento de vistas. Las ventajas de la materialización de una vista para las consultas deben considerarse frente a los costes de almacenamiento y la sobrecarga añadida por las actualizaciones.

2.3.3. Normalización

La normalización es una técnica para diseñar la estructura lógica de los datos de una base de datos en el modelo relacional, desarrollada por E. F. Codd en 1972.

Es una estrategia de diseño de abajo a arriba: se parte de los atributos y éstos se van agrupando en relaciones (tablas) según su afinidad. Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.

Uno de los conceptos fundamentales en la normalización es el de *dependencia funcional*. Una dependencia funcional es una relación entre atributos de una misma relación (tabla).

Si X e Y son atributos de la relación R , se dice que Y es funcionalmente dependiente de X (se denota por $X \rightarrow Y$) si cada valor de X tiene asociado un solo valor de Y (X e Y pueden constar de uno o varios atributos). A X se le denomina “determinante”, ya que X determina el valor de Y . Se dice que el atributo Y es “completamente dependiente” de X si depende funcionalmente de X y no depende de ningún subconjunto de X .

La dependencia funcional es una noción semántica. Si hay o no dependencias funcionales entre atributos no lo determina una serie abstracta de reglas, sino, más bien, los modelos mentales del usuario y las reglas de negocio de la organización para la que se desarrolla la base de datos. Cada dependencia funcional es una clase especial de regla de integridad y representa una relación de uno a muchos.

En el proceso de normalización se debe ir comprobando que cada relación (tabla) cumple una serie de reglas que se basan en la clave primaria y las dependencias funcionales. Cada regla que se cumple aumenta el grado de normalización. Si una regla no se cumple, la relación se debe descomponer en varias relaciones que sí la cumplan.

La normalización se lleva a cabo en una serie de pasos. Cada paso corresponde a una forma normal que tiene unas propiedades. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto (más formal) y, por lo tanto, son menos vulnerables a las anomalías de actualización. El modelo relacional sólo requiere un conjunto de relaciones en la primera forma normal. Las restantes formas normales son opcionales. Sin embargo, para evitar las anomalías de actualización, es recomendable llegar al menos a la tercera forma normal. A continuación se definen las cinco formas normales y la forma normal de Boyce-Codd.

- Primera forma normal (1FN): Una relación está en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna. Si una relación no está en 1FN, hay que eliminar de ella los grupos repetitivos. Un grupo repetitivo será el atributo o grupo de atributos que tiene múltiples valores para cada tupla de la relación. Hay dos formas de eliminar los grupos repetitivos. En la primera, se repiten los atributos con un solo valor para cada valor del grupo repetitivo. De este modo, se introducen redundancias ya que se duplican valores, pero estas redundancias se eliminarán después mediante las restantes formas

normales. La segunda forma de eliminar los grupos repetitivos consiste en poner cada uno de ellos en una relación aparte, heredando la clave primaria de la relación en la que se encontraban. En resumen, un conjunto de relaciones se encuentra en 1FN si ninguna de estas relaciones tiene grupos repetitivos.

- Segunda forma normal (2FN): Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria. La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN. Las relaciones que no están en 2FN pueden sufrir anomalías cuando se realizan actualizaciones. Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen).
- Forma Normal de Boyce-Codd (FNBC): Una de las formas normales mas deseables que se pueden obtener es la forma normal de Boyce-Codd (FNBC). Un esquema de relación R está en FNBC respecto a un conjunto de dependencias funcionales F si, para todas las dependencias funcionales de F^+ de la forma $\alpha \rightarrow \beta$, donde $\alpha \subseteq R$ y $\beta \subseteq R$, se cumple al menos una de las siguientes condiciones:
 1. $\alpha \rightarrow \beta$ es una dependencia funcional trivial (es decir, $\beta \subseteq \alpha$)
 2. α es una superclave del esquema R .

Un diseño de base de datos está en FNBC si cada miembro del conjunto de esquemas de relación que constituye el diseño está en FNBC.

- Tercera forma normal: FNBC exige que todas las dependencias no triviales sean de la forma $\alpha \rightarrow \beta$ donde α es una superclave. 3FN relaja ligeramente esta restricción permitiendo dependencias funcionales no triviales cuya parte izquierda no sea una superclave. Un esquema de relación R está en tercera forma normal (3FN) respecto a un conjunto F de dependencias funcionales si, para todas las dependencias funcionales de F^+ de la forma $\alpha \rightarrow \beta$, donde $\alpha \subseteq R$ y $\beta \subseteq R$, se cumple al menos una de las siguientes condiciones:
 1. $\alpha \rightarrow \beta$ es una dependencia funcional trivial.
 2. α es una superclave de R .
 3. Cada atributo A de $\beta - \alpha$ está contenido en alguna clave candidata de R .

Obsérvese que cualquier esquema que satisfaga FNBC satisface también 3FN, ya que cada una de sus dependencias funcionales satisfará una de las dos primeras alternativas. Por tanto, FNBC es una restricción más estricta que 3FN.

- Cuarta forma normal: La 4NF se asegura de que las dependencias multivaluadas independientes estén correcta y eficientemente representadas en un diseño de base de datos. La 4NF es el siguiente nivel de normalización después de la forma normal de Boyce-Codd (BCNF). Una tabla está en 4NF si y solo si esta en Tercera forma normal o en BCNF (Cualquiera de ambas) y no posee dependencias multivaluadas no triviales. Una tabla con una dependencia multivaluada es una donde la existencia

de dos o más relaciones independientes muchos a muchos causa redundancia; y es esta redundancia la que es suprimida por la cuarta forma normal. Sea R un esquema de relación. La dependencia multivaluada $X \twoheadrightarrow Y$ vale en R si los pares de tuplas t_1 y t_2 en R , tal que $t_1[X] = t_2[X]$ existen las tuplas t_3 y t_4 en R tales que:

$$t_1[X] = t_2[X] = t_3[X] = t_4[X]$$

$$t_3[Y] = t_1[Y]$$

$$t_3[R-X-Y] = t_2[R-X-Y]$$

$$t_4[Y] = t_2[Y]$$

$$t_4[R-X-Y] = t_1[R-X-Y]$$

En otras palabras se puede decir que: $X \twoheadrightarrow Y$ si dado un valor de X , hay un conjunto de valores de Y asociados y este conjunto de valores de Y no está relacionado (ni funcional ni multifuncionalmente) con los valores de $R - X - Y$ (donde R es el esquema), es decir Y es independiente de los atributos de $R-X-Y$. Una dependencia multivaluada de la forma $X \twoheadrightarrow Y$, es trivial cuando el conjunto de atributos $\{X, Y\}$ conforma el total de los atributos del esquema.

- Quinta forma normal (5FN): Una relación está en la Quinta Forma Normal (5FN) o Forma Normal de Proyección-Unión si está en 4FN y las únicas dependencias que existen son las dependencias de unión de una relación con sus proyecciones relacionándose entre las distintas proyecciones mediante la clave primaria o cualquier clave alternativa. La 5FN se emplea cuando en una misma relación tenemos mucha información redundante, con pocos atributos o cuando una relación posee una gran cantidad de atributos y se hace por ello inmanejable. Para conseguir que una relación 4FN con gran cantidad de atributos esté en 5FN, se parte la relación original en tantas relaciones como se desee, teniendo cada una de ellas en común con las demás los campos que forman la clave primaria en la relación original.

Aunque la FNBC y la 5FN no se apliquen en el diseño de la base de datos no debería afectar en la funcionalidad de esta, a pesar de que pueda ser menos perfecto.

2.3.4. SQL

El algebra relacional proporciona una notación formal y concisa para la representación de consultas, sin embargo los sistemas de bases de datos necesitan un lenguaje de consultas menos complejo para los usuarios.

IBM desarrolló la versión original en su Laboratorio de Investigación de San José (*San José Research Center*, actualmente Centro de Investigación de Almadén, *Almadén Research Center*). IBM implementó el lenguaje, originalmente denominado Sequel, como parte del proyecto System R, a principios de 1970. El lenguaje Sequel ha evolucionado desde entonces y su nombre ha pasado a ser SQL (*Structured Query Language*, -Lenguaje estructurado de consultas-).

2.3.4.1. Definiciones

SQL se ha establecido como el lenguaje estándar de bases de datos relacionales y a pesar de que SQL se considere un lenguaje de consultas, contiene muchas otras capacidades

además de la consulta en bases de datos. Incluye características para definir la estructura de los datos, para la modificación de los datos en la base de datos y para la especificación de restricciones de seguridad.

El lenguaje SQL tiene varios componentes:

- Lenguaje de definición de datos (LDD). El LDD de SQL proporciona órdenes para la definición de esquemas de relación, borrado de relaciones, creación de índices y modificación de esquemas de relación.
- Lenguaje interactivo de manipulación de datos (LMD). El LMD de SQL incluye un Lenguaje de consultas, basado tanto en el álgebra relacional como en el cálculo relacional de tuplas. Incluye también órdenes para insertar, borrar y modificar tuplas de la base de datos.
- Definición de vistas. El LDD de SQL incluye órdenes para la definición de vistas.
- Control de transacciones. SQL incluye órdenes para la especificación del comienzo y final de transacciones.
- SQL incorporado y SQL dinámico. SQL dinámico e incorporado define cómo se pueden incorporar las instrucciones SQL en lenguajes de programación de propósito general, tales como C, C++, Java, PL/I, Cobol, Pascal y Fortran.
- Integridad. El LDD de SQL incluye órdenes para la especificación de las restricciones de integridad que deben satisfacer los datos almacenados en la base de datos. Las actualizaciones que violen las restricciones de integridad se rechazan.
- Autorización. El LDD de SQL incluye órdenes para especificar derechos de acceso para las relaciones y vistas.

Una base de datos relacional consiste en un conjunto de relaciones, a cada una de las cuales se le asigna un nombre único. La estructura básica de una expresión SQL consiste en tres cláusulas: select, from y where.

- La cláusula select corresponde a la operación proyección del álgebra relacional. Se usa para listar los atributos deseados del resultado de una consulta.
- La cláusula from corresponde a la operación producto cartesiano del álgebra relacional. Lista las relaciones que deben ser analizadas en la evaluación de la expresión.
- La cláusula where corresponde al predicado selección del álgebra relacional. Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula from.

Sirva de ejemplo para estas tres cláusulas la siguiente consulta típica de SQL:

select $A_1, A_2, \dots, A_n$ **from** $r_1, r_2, \dots, r_m$ **where** P

Cada A_i representa un atributo, y cada r_i una relación. P es un predicado. La consulta es equivalente a la expresión del álgebra relacional:

$\Pi_{A_1, A_2, \dots, A_n} (\sigma_P (r_1 \times r_2 \times \dots \times r_m))$

SQL forma el producto cartesiano de las relaciones incluidas en la cláusula from, lleva a cabo la selección del álgebra relacional usando el predicado de la cláusula where y entonces proyecta el resultado sobre los atributos de la cláusula select.

SQL proporciona un mecanismo para renombrar tanto relaciones como atributos. Para ello utiliza la cláusula **as**, que tiene la forma siguiente:

nombre-antiguo as nombre-nuevo

la cláusula **as** puede aparecer tanto en **select** como en **from**.

select A_1 **as** $b_1, A_2, \dots, A_n$ **from** $r_1, r_2, \dots, r_m$ **where** P

SQL especifica las cadenas encerrándolas entre comillas simple, por ejemplo 'cadena'. Un carácter comilla que sea parte de una cadena se puede especificar usando dos caracteres comilla; por ejemplo, la cadena «El carácter ' se puede ver en esta cadena» se puede especificar como 'El carácter ' se puede ver en esta cadena'.

La operación más usada sobre cadenas es el encaje de patrones, para el que se usa el operador **like**. Para la descripción de patrones se utilizan los dos caracteres especiales siguientes:

- Tanto por ciento (%): El carácter % encaja con cualquier subcadena.
- Subrayado (_): El carácter _ encaja con cualquier carácter.

Los patrones son muy sensibles, esto es, los caracteres en mayúsculas no encajan con los caracteres en minúscula, o viceversa.

Para que los patrones puedan contener los caracteres especiales patrón (esto es, % y _), SQL permite la especificación de un carácter de escape. El carácter de escape se utiliza inmediatamente antes de un carácter especial patrón para indicar que ese carácter especial va a ser tratado como un carácter normal. El carácter de escape para una comparación **like** se define utilizando la palabra clave **escape**. Para ilustrar esto, considérense los siguientes patrones, los cuales utilizan una barra invertida (\) como carácter de escape:

- **like** 'ab\%cd%' **escape** '\' encaja con todas las cadenas que empiecen por ab%cd.
- **like** 'ab\\cd%' **escape** '\' encaja con todas las cadenas que empiecen por ab\cd .

SQL ofrece al usuario cierto control sobre el orden en el cual se presentan las tuplas de una relación. La cláusula **order by** hace que las tuplas resultantes de una consulta se presenten en un cierto orden. Sírvese de ejemplo la Figura 2.x, donde para listar en orden alfabético todas las sucursales donde los clientes tienen un saldo mayor a 500 se escribirá:

```
select nombre_sucursal  
from cuenta  
where saldo >500  
order by nombre-sucursal
```

2.3.4.2. SQL Server 2000

Una base de datos de Microsoft SQL Server [Delaney 2001] es una colección de objetos que contiene y maneja datos. Las características principales de SQL Server son:

- Es una colección de muchos objetos, como tablas, vistas, procedimientos almacenados y restricciones. El límite teórico es de $2^{31}-1$ (más de 2 billones) de objetos.
- Es propiedad de una única cuenta de usuario, pero puede contener objetos propiedad de otros usuarios.
- Tiene su propio conjunto de tablas del sistema que catalogan la definición de base de datos.
- Mantienen su propio conjunto de cuentas de usuario y seguridad.
- Es la unidad principal de recuperación y mantiene la coherencia lógica entre los objetos de la base de datos.

- Tiene su propio registro de las transacciones y administra las transacciones dentro de la base de datos.
- Puede participar en transacciones de confirmación en dos fases con otras bases de datos de SQL Server en el mismo servidor o en servidores distintos.
- Puede estar distribuida por distintas unidades de disco y archivos del sistema operativo.
- Puede tener un tamaño comprendido entre 1 Mb y un límite teórico 1.048.576 Tb.
- Puede expandirse y comprimirse de manera automática o mediante un comando.
- Puede tener objetos unidos en consultas con objetos de otras bases de datos en la misma instalación de SQL Server o en servidores vinculados.
- Puede tener opciones específicas activadas o desactivadas.
- Es conceptualmente similar, pero con más posibilidades, que el concepto de esquema SQL ANSI.

SQL Server 2000 incluye seis bases de datos:

- Master: La base de datos master se compone de tablas del sistema que realizan el seguimiento de la instalación del servidor como un todo, y de todas las demás bases de datos que se creen posteriormente.
- Model: La base de datos model es simplemente una base de datos plantilla. Cada vez que se crea una nueva base de datos, SQL Server hace una copia de model como base para la nueva base de datos.
- Tempdb: La base de datos temporal, tempdb, es un espacio de trabajo, es exclusiva entre todas las demás bases de datos porque se vuelve a crear, no se recupera, cada vez que se arranca SQL Server.
- Pubs: La base de datos pubs es una base de datos de ejemplo utilizada ampliamente en gran parte de la documentación de SQL Server.
- Northwind: La base de datos northwind es una base de datos de ejemplo desarrollada originalmente para su uso con Microsoft Access.
- Msdb: La base de datos msdb la utiliza el servicio Agente SQL, que realiza actividades programadas como copias de seguridad y tareas de duplicación.

Un archivo de base de datos, no es más que un archivo del sistema operativo. Una base de datos se distribuye en al menos dos archivos de bases de datos y estos archivos se especifican al crear o modificar una base de datos. SQL Server permite los tres tipos siguientes de archivos de bases de datos:

- Archivos de datos primarios: Toda base de datos tiene un archivo de datos primario que realiza el seguimiento de todos los demás archivos de la misma. El nombre de un archivo de datos primario tiene la extensión MDF.
- Archivos de datos secundarios: Una base de datos puede tener cero o varios archivos de datos secundarios, estos archivos tienen la extensión NDF.
- Archivos de registro: Toda base de datos tiene al menos un archivo de registro que contiene toda la información necesaria para recuperar todas las transacciones de la misma, estos archivos tienen la extensión LDF.

La forma en la cual se crea una base de datos en SQL Server por medio del comando CREATE DATABASE es la siguiente:

CREATE DATABASE Archive

```

ON
PRIMARY
(NAME = Arch1,
FILENAME = 'c:\archivos de programa\microsoft sql server\mssql\data\archdat1.mdf',
SIZE = 100 MB,
FILEGROWTH = 20),
(NAME = Arch2,
FILENAME = 'c:\archivos de programa\microsoft sql server\mssql\data\archdat2.ndf',
SIZE = 100 MB,
MAXSIZE = 200,
FILEGROWTH = 20),
LOG ON
(NAME = Archlog1,
FILENAME = 'c:\archivos de programa\microsoft sql server\mssql\data\archlog1.ldf',
SIZE = 100 MB,
MAXSIZE = 200,
FILEGROWTH = 20),
LOG ON

```

Se puede utilizar la instrucción ALTER DATABASE para modificar la base de datos de alguna de las siguientes maneras:

- Cambiar el nombre de la base de datos.
- Agregar uno o varios archivos de datos nuevos a la base de datos.
- Agregar uno o varios archivos del registro nuevos a la base de datos.
- Eliminar un archivo o un grupo de archivos de la base de datos.
- Agregar u nuevo grupo de archivos a la base de datos.
- Modificar algún archivo existente.
- Modificar un grupo de archivos existente.

El siguiente ejemplo aumenta el tamaño de un archivo de base de datos.

```

USE master
GO
ALTER DATABASE Test1
MODIFY FILE
(NAME = 'test1dat3',
SIZE = 20 MB)

```

Para crear una tabla, SQL Server utiliza la sintaxis CREATE TABLE estándar del ANSI SQL. Un ejemplo simple de este comando es:

```

CREATE TABLE
{
nombre char(30),
telefono char (12),
id_emp char (4)
}

```

SQL Server proporciona un gran número de tipos de datos tales como: int, smallint, tinyint, datetime, char, varchar, text, etc., seleccionar el tipo de datos adecuado es simplemente cuestión de asociar el dominio de valores que necesita almacenar con el tipo de datos correspondiente.

Existe en SQL Server una propiedad denominada IDENTITY (identidad), la cual se refiere a la propiedad de una columna para almacenar un tipo de datos numérico el cual será un valor auto incrementable e irrepetible. Al crearse una tabla en la cual se haya definido esta propiedad para alguna de sus columnas, se posee la facilidad de determinar el valor

inicial (semilla) y también el incremento que tendrá el valor de la columna al insertarse una nueva fila.

Las consultas en SQL Server siguen las reglas del ANSI SQL así como el álgebra relacional aunque con algunas variaciones, pero en general la forma de estructurarlas es intuitiva.

La instrucción SELECT (seleccionar) es el comando SQL que se utiliza, y es la forma fundamental de consultar datos. Las consultas con semántica incorrecta provocan errores en las aplicaciones, por eso es importante entender la correcta formulación de las consultas. El formato básico de SELECT se muestra a continuación, tomando en cuenta que se han utilizado corchetes ([]) para identificar los elementos opcionales:

<i>SELECT [DISTINCT][TOP n]</i>	<i>< Columnas elegidas, eliminando opcionalmente las filas duplicadas del conjunto de resultados o limitando el número de filas que se deben devolver >.</i>
<i>FROM</i>	<i>< nombre de tablas >.</i>
<i>[WHERE]</i>	<i>< Criterios que deben cumplirse para seleccionar una fila >.</i>
<i>[GROUP BY]</i>	<i>< Columnas para agrupar funciones agregadas >.</i>
<i>[HAVING]</i>	<i>< Criterios que deben cumplirse para las funciones agregadas >.</i>
<i>[ORDER BY]</i>	<i>< Especificación opcional de cómo deben ordenarse los resultados >.</i>

La siguiente consulta que sirve como ejemplo, recupera todas las columnas de una tabla ficticia denominada usuarios:

*SELECT * FROM usuarios*

Las combinaciones están basadas en las claves primarias y foráneas de las tablas implicadas, aunque no es necesario que las tablas tengan claves declaradas explícitamente. Para combinar las tablas se deben incluir estas en la cláusula FROM de la instrucción SELECT, especificando que las columnas que componen las claves tengan los mismos valores.

La sintaxis ANSI JOIN está basada en ANSI SQL-92. Las principales diferencias entre los dos tipos de formulaciones de las combinaciones son:

- ANSI JOIN utiliza realmente la palabra clave JOIN.
- ANSI JOIN separa las condiciones de combinación de las condiciones de búsqueda.

Aunque es un poco más larga, la sintaxis de JOIN explícita es más legible. A continuación se resumen los cinco tipos de operaciones ANSI JOIN:

- INNER JOIN devuelve filas de cualquiera de las tablas solo si tienen una fila correspondiente en la otra tabla. En otras palabras, INNER JOIN desecha las filas en las que no se cumple la condición de combinación específica, indicada en la cláusula ON.

TablaA INNER JOIN TablaB ON condición-combinación

- LEFT OUTER JOIN devuelve todas las filas para las que existe una conexión entre TablaA y TablaB. Además, devuelve todas las filas de TablaA para las que no existe una fila correspondiente en TablaB. En otras palabras, conserva las filas sin concordancia de TablaA. TablaA se llama a veces la tabla conservada. Al devolver las filas sin concordancia de TablaA, las columnas seleccionadas de TablaB se devuelven como NULL.

TablaA LEFT OUTER JOIN TablaB ON condición-combinación

- **RIGHT OUTER JOIN** devuelve todas las filas para las que existe una conexión entre TablaA y TablaB; además, devuelve todas las filas de TablaB para las que no existe una fila correspondiente en TablaA. En otras palabras, conserva las filas sin concordancia de TablaB y en este caso TablaB es la tabla conservada. Al devolver las filas sin concordancia de TablaB, las columnas seleccionadas de TablaA se devuelven como NULL.

TablaA RIGHT OUTER JOIN TablaB ON condición-combinación

- **FULL OUTER JOIN** devuelve todas las filas para las que existe una conexión entre TablaA y TablaB. Además, devuelve todas las filas de TablaA para las que no existe una fila correspondiente en TablaB, devolviendo todos los valores seleccionados desde TablaB como NULL. Además devuelve todas las filas de TablaB para las que no existe una fila correspondiente en TablaA, devolviendo todos los valores seleccionados desde TablaA como NULL. En otras palabras FULL OUTER JOIN actúa como una combinación de LEFT OUTER JOIN y RIGHT OUTER JOIN.

TablaA FULL OUTER JOIN TablaB ON condición-combinación

- **CROSS JOIN** devuelve todas las filas de TablaA combinadas con todas las filas de TablaB, no existe cláusula ON para indicar una columna de conexión entre las tablas. CROSS JOIN devuelve un producto cartesiano de las dos tablas.

TablaA CROSS JOIN TablaB

SQL tiene tres instrucciones básicas de modificación de datos: INSERT, UPDATE y DELETE que junto con SELECT se denominan DML (Data Manipulation Language - Lenguaje de Manipulación de Datos -), y que a continuación se describen:

- **INSERT** se utiliza para insertar una fila a una tabla, siendo su sintaxis la siguiente:

*INSERT [INTO] {nombre_tabla | nombre_vista} [(lista_columnas)]
VALUES lista_valores*

En SQL Server, la utilización de INTO siempre es opcional, sin embargo, el ANSI SQL exige utilizar INTO. Si está proporcionando un valor para cada columna de la tabla, y los valores aparecen en el orden exacto en el que están definidas las columnas, la lista de columnas también es opcional. La única excepción a esto es cuando la tabla tiene una columna de tipo IDENTITY, que no debe incluirse en la lista de valores.

- **UPDATE** se utiliza para cambiar los valores de las filas existentes, generalmente UPDATE contiene una cláusula WHERE que limita la actualización a un subconjunto de filas de la tabla o incluso una sola fila. Si no se especifica una cláusula WHERE, UPDATE modificara todas las filas de la tabla, su sintaxis es la siguiente:

*UPDATE {nombre_tabla | nombre_vista}
SET nombre_columna1 = {expresion1}
[,nombre_columna2 = {expresion2}]
[,nombre_columnan = {expresionn}]
WHERE condiciones_busqueda*

- **DELETE** se utiliza para eliminar filas de una tabla. Como solo se eliminan filas, no columnas, nunca hay que especificar nombres de columnas así como se hace en INSERT o UPDATE, pero en otros muchos casos DELETE actúa como UPDATE, ya que hay que especificar una cláusula WHERE para limitar la eliminación a ciertas filas ya que de lo contrario se eliminarán todas las filas de la tabla, pero a

pesar de esto la clausula WHERE continua siendo opcional. Su sintaxis es la siguiente:

DELETE [FROM] {nombre_tabla | nombre_vista} [WHERE {condición_búsqueda}]

La preposición FROM es estándar ANSI, pero su inclusión es siempre opcional en SQL Server.

2.4. Desarrollo web

En la actualidad existes diferentes plataformas para el desarrollo tanto de Servicios Web como de aplicaciones web, sin embargo muy pocas ofrecen un marco de desarrollo totalmente integrado. En este punto es en donde la elección de las herramientas correctas para el desarrollo de las aplicaciones y servicios requiere de un análisis de las diferentes opciones que se tienen ya que el uso de herramientas que no proporcionen la interoperabilidad necesaria puede ocasionar que no se obtengan los resultados esperados.

Microsoft ha desarrollado el .NET Framework con la finalidad precisamente de brindar un ambiente de desarrollo integrado, el cual proporciona: lenguajes de programación totalmente orientados a objetos, librerías de clases, un administrador de código en tiempo de ejecución y un entorno de programación de aplicaciones web.

Las secciones siguientes describen de forma general la forma en que trabaja el .NET Framework y también se describe el uso de de JavaScript y DOM, los cuales aunque no forman parte del .NET Framework, le proporcionan características específicas que este no implementa en su totalidad.

2.4.1. Definiciones .NET

El .NET Framework [Northrup 2006], [Snell 2009] de Microsoft es un proyecto 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, .NET.

El .NET Framework es un componente de software que puede ser añadido al sistema operativo Windows, provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta tecnología ofrece soluciones a los problemas de programación actuales, como son la administración de código o la programación para internet. Para aprovechar al máximo las características de .NET es necesario entender la arquitectura básica en la que esta implementada esta tecnología y así beneficiarse de todas las características que ofrece esta nueva plataforma.

El .Net Framework se compone de cuatro partes esenciales (ver Figura 2.7):

1. El entorno común de ejecución (CLR).
2. Un conjunto de bibliotecas de clases.
3. Un grupo de lenguajes de programación.
4. El entorno ASP .NET.



Figura 2.7. Esquema General del .NET Framework.

.NET Framework fue diseñado con tres objetivos en mente. Primero, debía lograr aplicaciones Windows mucho más estables, aunque también debía proporcionar una aplicación con un mayor grado de seguridad. En segundo lugar, debía simplificar el desarrollo de aplicaciones y Servicios Web que no solo funcionen en plataformas tradicionales, sino también en dispositivos móviles. Por último, el entorno fue diseñado para proporcionar un solo grupo de bibliotecas que pudieran trabajar con varios lenguajes.

.NET Framework fue diseñado con una idea en mente: potenciar el desarrollo de internet. Este nuevo incentivo para el desarrollo de internet se llama Servicios Web. Se puede pensar en los Servicios Web como en una página web que interactúa con programas en lugar de con personas. En lugar de enviar páginas web, un Servicio Web recibe una solicitud en formato XML, realiza una función en concreto y luego devuelve una respuesta al solicitante en forma de mensaje XML.

Aparte del desarrollo web, con .NET Framework también se pueden construir las tradicionales aplicaciones Windows. Estas aplicaciones creadas con .NET Framework se basan en *Windows Form*, que es una mezcla de los formularios de Visual Basic 6.0 y los formularios de Visual C++. Aunque los formularios parecen iguales a sus predecesores, están completamente orientados a objetos y basados en clases, de forma muy parecida a los formularios objeto de *Microsoft Foundation Class*.

La creación de aplicaciones con .NET también brindan muchas mejoras no disponibles en otros lenguajes, como la seguridad. Estas medidas de seguridad pueden determinar si una aplicación puede escribir o leer un archivo de disco. También permiten insertar firmas digitales en la aplicación para asegurarse de que la aplicación fue escrita por una fuente de confianza.

Antes de que se desarrollara .NET, cada lenguaje constaba de su propio entorno de ejecución.

2.4.1.1. CLR

El CLR acrónimo de *Common Language Runtime* [Northrup 2006] es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado.

El CLR es el verdadero núcleo del .NET Framework, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, *Microsoft Intermediate Language*). Para generar dicho código el compilador se basa en el *Common Language Specification* (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador.

Sin embargo, el código generado en MSIL no es código máquina y por tanto no puede ejecutarse directamente. Se necesita un segundo paso en el que una herramienta denominada compilador JIT (*Just-In-Time*) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.

De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.

La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y, el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente.

La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los Servicios Web XML.

.NET Framework puede alojarse en componentes no administrados que cargan *Common Language Runtime* en sus procesos e inician la ejecución de código administrado, con lo que se crea un entorno de software en el que se pueden utilizar características administradas y no administradas. En .NET Framework no sólo se ofrecen varios hosts de motor en tiempo de ejecución, sino que también se admite el desarrollo de estos hosts por parte de terceros.

Por ejemplo, ASP.NET aloja el motor en tiempo de ejecución para proporcionar un entorno de servidor escalable para el código administrado. ASP.NET trabaja directamente con el motor en tiempo de ejecución para habilitar aplicaciones de ASP.NET y Servicios Web XML.

2.4.1.2. Biblioteca de clases

La biblioteca de clases de .NET Framework [Northrup 2006] es una colección de tipos reutilizables que se integran estrechamente con *Common Language Runtime*. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework

sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Por ejemplo, las clases de colección de .NET Framework implementan un conjunto de interfaces que puede usar para desarrollar sus propias clases de colección. Éstas se combinarán fácilmente con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recolección de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

- Aplicaciones de consola
- Aplicaciones GUI de Windows (Windows Forms)
- Aplicaciones de Windows Presentation Foundation (WPF)
- Aplicaciones de ASP.NET
- Servicios Web.
- Servicios de Windows
- Aplicaciones orientadas a servicios utilizando Windows Communication Foundation (WCF).
- Aplicaciones habilitadas para el flujo de trabajo utilizando Windows Workflow Foundation (WF)

2.4.1.3. ASP .NET

ASP.NET acrónimo de *Active Server Pages* (Paginas Activas de Servidor) [Hart 2006], [Foggon 2006], [AL 2008], [Snell 2009] es un marco (*framework*) para programar aplicaciones web, de un modo similar al que se programan las aplicaciones *windows*. El componente principal son los *Web Forms* (formularios web) que permiten, entre otras cosas, separar la interfaz del usuario de la funcionalidad de la aplicación.

ASP .NET es una plataforma de programación Web unificada que proporciona los servicios necesarios para crear aplicaciones Web. Si bien ASP.NET proporciona también un modelo de programación y una estructura nuevos para crear aplicaciones más seguras, escalables y estables.

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

ASP.NET se ha diseñado para funcionar sin problemas con editores HTML WYSIWYG y otras herramientas de programación como Microsoft Visual Studio .NET. Todo esto, además de hacer más fácil la programación Web, ofrece todas las ventajas de estas

herramientas, con una GUI que se puede utilizar para ubicar controles de servidor en una página Web e integrar completamente la compatibilidad con la depuración.

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

- Los formularios Web Forms permiten crear páginas Web basadas en formularios muy eficaces. Al crear estas páginas, se pueden usar controles de servidor ASP.NET para crear elementos comunes de la interfaz de usuario y programarlos para que realicen las tareas comunes. Estos controles permiten crear con rapidez un formulario Web Form a partir de componentes integrados reutilizables o personalizados, con un código de página simplificado
- Un servicio Web XML proporciona los medios para obtener acceso a la funcionalidad del servidor de manera remota. Con los Servicios Web XML, las empresas pueden exponer interfaces de programación a sus datos o lógica empresarial, que, a su vez, pueden obtener y manipular las aplicaciones de cliente y servidor. Los Servicios Web XML permiten el intercambio de datos en escenarios cliente-servidor o servidor-servidor, utilizando estándares como los servicios de mensajería HTTP y XML para que los datos pasen los servidores de seguridad. Los Servicios Web XML no están ligados a ninguna tecnología de componentes ni a ninguna convención de llamada a objetos concretas. En consecuencia, pueden tener acceso a los Servicios Web XML los programas escritos en cualquier lenguaje, utilizando cualquier modelo de componentes y que se ejecuten en cualquier sistema operativo.

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

2.4.1.4. C# .NET

Microsoft diseño C# [Archer 2001] [Turtschi 2002] desde su base para aprovechar el nuevo entorno .NET Framework, sin embargo, C# es un lenguaje orientado a objetos (desde el principio), moderno y seguro. Las principales características de C# son:

- Clases: Todo el código y los datos en C# deben ser incluidos en una clase. No se puede definir una variable fuera de una clase y no se puede escribir ningún código que no esté en una clase. Las clases pueden tener constructores, que se ejecutan cuando se crea un objeto de la clase, y un destructor, que se ejecuta cuando un objeto de la clase es destruido. Las clases admiten herencias simples y todas las clases derivan al final de una clase base llamada Objeto. C# admite técnicas de versiones para ayudar a que sus clases evolucionen con el tiempo mientras

mantienen la compatibilidad con código que use versiones anteriores de sus clases. C# permite agrupar las clases en una colección de clases llamada espacio de nombre. Los espacios de nombre tienen nombres y pueden servir de ayuda para organizar colecciones de clases en agrupaciones lógicas.

- Tipos de datos: C# permite trabajar con dos tipos de datos: de valor y de referencia. Los datos de valor contienen valores reales. Los de referencia contienen referencias a valores almacenados en algún lugar de la memoria. Los tipos primitivos como `char`, `int` y `float`, junto con los valores y estructuras comentados, son tipos de valor. Los tipos de referencia tienen variables que tratan con objetos y matrices. C# viene con tipos de referencia predefinidos (*objet* y *string*), junto con tipos de valor predefinidos (*sbyte*, *short*, *int*, *long*, *byte*, *ushort*, *uint*, *ulong*, *float*, *double*, *bool*, *char* y *decimal*). También pueden definirse en el código tipos de valor y referencia propios.
- Funciones: Una función es un fragmento de código que puede ser invocado y que puede o no devolver un valor al código que lo invoca en un principio. Una función suele asociarse a fragmentos de código que devuelven información, mientras que un método no suele devolver información. Sin embargo para nuestros propósitos, generalizamos y nos referimos a las dos como funciones. Las funciones pueden tener cuatro tipos de parámetros:
 1. Parámetros de entrada: tiene valores que son enviados a la función, pero la función no puede cambiar esos valores.
 2. Parámetros de salida: no tienen valor cuando son enviados a la función, pero la función puede darles un valor y enviar el valor de vuelta al invocador.
 3. Parámetros de referencia: introducen una referencia en otro valor. Tiene un valor de entrada para la función y ese valor puede ser cambiado dentro de la función.
 4. Parámetros Params: definen un número variable de argumentos en una lista.

El CLR monitoriza el uso de memoria y recupera automáticamente más cuando la necesita. También libera memoria automáticamente cuando detecta que ya no está siendo usada (esto también se conoce como recolección de objetos no utilizados).

C# proporciona operadores que permiten escribir expresiones matemáticas y de bits. Muchos (pero no todos) de estos operadores pueden ser redefinidos, permitiéndole cambiar la forma en que trabajan estos operadores. C# admite una larga lista de expresiones que permiten definir varias rutas de ejecución dentro del código. Las instrucciones de flujo de control que usan palabras clave como `if`, `switch`, `while`, `for`, `break` y `continue` permiten al código bifurcarse por caminos diferentes, dependiendo de los valores de sus variables. Las clases pueden contener código y datos. Cada miembro de una clase tiene algo llamado ámbito de accesibilidad, que define la visibilidad del miembro con respecto a otros objetos. C# admite los ámbitos de accesibilidad `public`, `protected`, `internal`, `protected internal` y `private`.

- Interfaces: C# admite interfaces que son grupos de propiedades, métodos y eventos que especifican un conjunto de funcionalidad. Las clases C# pueden implementar interfaces, que informan a los usuarios de que la clase admite el conjunto de funcionalidades documentado por la interfaz. Pueden desarrollar implementaciones de interfaces sin interferir con ningún código existente. Una vez que la interfaz ha sido publicada, no se puede modificar, pero puede evolucionar mediante herencia. Las clases C# pueden implementar muchas interfaces, aunque las clases solo pueden derivarse de una clase base.

2.4.2. JavaScript

JavaScript [Goodman 2001], [Villa 2003], [Eguiluz 2009] es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java.

El código JavaScript se encierra entre etiquetas `<script>` y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta `<head>`) así como se muestra a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Titulo de la pagina</title>
<script type="text/javascript">
alert("Un mensaje de prueba");
</script>
</head>
<body>
</body>
</html>
```

Las instrucciones JavaScript también se pueden incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta `<script>`. Sin embargo estos archivos además de ser enlazados, se debe invocar a sus métodos por medio de los eventos de los elementos del documento HTML. Se pueden crear todos los archivos JavaScript que sean necesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

Ejemplo de un archivo JavaScript nombrado “codigo.js”:

```
Function mensaje{
alert("Un mensaje de prueba");
}
```

Ejemplo de un documento XHTML incorporando el archivo “código.js”:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript" src="codigo.js"></script>
</head>
<body onload="mensaje()">
<p>Un párrafo de texto.</p>
</body>
</html>
```

Además del atributo type, este método requiere definir el atributo src, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta <script> solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas <script> como sean necesarias.

Los archivos de tipo JavaScript son documentos normales de texto con la extensión .js, que se pueden crear con cualquier editor de texto como Notepad, Wordpad, EmEditor, UltraEdit, Vi, etc., siempre y cuando no se agregue formato al cuerpo del texto.

La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código XHTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas XHTML que lo enlazan.

Desde su aparición, JavaScript siempre fue utilizado de forma masiva por la mayoría de sitios de internet. La aparición de Flash disminuyó su popularidad, ya que Flash permitía realizar algunas acciones imposibles de llevar a cabo mediante JavaScript. Sin embargo, la aparición de las aplicaciones AJAX programadas con JavaScript le ha devuelto una popularidad sin igual dentro de los lenguajes de programación web.

En cuanto a las limitaciones, JavaScript fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los scripts. De esta forma, los scripts de JavaScript no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó el script. Los scripts tampoco pueden cerrar ventanas que no hayan abierto esos mismos scripts. Las ventanas que se crean no pueden ser demasiado pequeñas ni demasiado grandes ni colocarse fuera de la vista del usuario (aunque los detalles concretos dependen de cada navegador). Además, los scripts no pueden acceder a los archivos del ordenador del usuario (ni en modo lectura ni en modo escritura) y tampoco pueden leer o modificar las preferencias del navegador.

Por último, si la ejecución de un script dura demasiado tiempo (por ejemplo por un error de programación) el navegador informa al usuario de que un script está consumiendo demasiados recursos y le da la posibilidad de detener su ejecución. A pesar de todo, existen alternativas para poder saltarse algunas de las limitaciones anteriores. La alternativa más utilizada y conocida consiste en firmar digitalmente el script y solicitar al usuario el permiso para realizar esas acciones.

Es necesario conocer los elementos básicos con los que se construyen las aplicaciones con JavaScript, y por tal motivo a continuación se hace un breve resumen de la forma en que se hace la programación en este lenguaje.

En JavaScript los textos se denotan por comillas dobles o comillas simples, pero no ambas a la vez. Podemos guardar indistintamente en una variable un número entero, decimal, positivo o negativo. Por defecto, el sistema de numeración es el decimal. Cuando una variable no contiene ningún valor, su contenido es nulo.

En JavaScript la conversión de datos puede ser explícita utilizando funciones que el mismo lenguaje proporciona, tales como: `parseFloat(valor)`, `parseInt(valor, base)`, `toString(valor)`.

Las variables de tipo *boolean* o booleano también se conocen con el nombre de variables de tipo lógico, su funcionamiento básico es muy sencillo. Una variable de tipo *boolean* almacena un tipo especial de valor que solamente puede tomar dos valores: `true` (verdadero) o `false` (falso).

Los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables. De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

El operador de asignación es el más utilizado y el más sencillo. Este operador se utiliza para guardar un valor específico en una variable. El símbolo utilizado es “=”.

Los operadores Incremento y decremento solamente son válidos para las variables numéricas y se utilizan para incrementar o decrementar en una unidad el valor de una variable.

Los operadores lógicos son imprescindibles para realizar aplicaciones complejas, ya que se utilizan para tomar decisiones sobre las instrucciones que debería ejecutar el programa en función de ciertas condiciones. El resultado de cualquier operación que utilice operadores lógicos siempre es un valor lógico o booleano.

La negación lógica se obtiene prefijando el símbolo `!` al identificador de la variable. Los operadores lógicos en JavaScript no difieren en relación a otros lenguajes de programación, utilizan una gramática similar: `AND = &&`, `OR = ||`.

JavaScript permite realizar manipulaciones matemáticas sobre el valor de las variables numéricas. Los operadores definidos son: suma (+), resta (-), multiplicación (*) y división (/).

Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=).

Los operadores que relacionan variables son imprescindibles para realizar cualquier aplicación compleja. El resultado de todos estos operadores siempre es un valor booleano.

Los programas que se pueden realizar utilizando solamente variables y operadores son una simple sucesión lineal de instrucciones básicas. Sin embargo, no se pueden realizar programas que muestren un mensaje si el valor de una variable es igual a un valor determinado y no muestren el mensaje en el resto de casos. Para realizar este tipo de programas son necesarias las estructuras de control de flujo, que con instrucciones del tipo “*si se cumple esta condición, hazlo; si no se cumple, haz esto otro*”. También existen instrucciones del tipo “*repite esto mientras se cumpla esta condición*”.

Si se utilizan estructuras de control de flujo, los programas dejan de ser una sucesión lineal de instrucciones para convertirse en programas *inteligentes* que pueden tomar

decisiones en función del valor de las variables. De entre las estructuras de control de flujo se encuentran: if, if...else, for, for...in, while, do...while, switch, que tienen una gramática similar a otros lenguajes de programación.

JavaScript incorpora una serie de herramientas y utilidades para el manejo de las variables. De esta forma, muchas de las operaciones básicas con las variables, se pueden realizar directamente con las utilidades que ofrece JavaScript. Las estructuras de control, los operadores y todas las utilidades propias de JavaScript, permiten crear scripts sencillos y de mediana complejidad. Sin embargo, para las aplicaciones más complejas son necesarios otros elementos como las funciones y otras estructuras de control más avanzadas. Cuando se desarrolla una aplicación compleja, es muy habitual utilizar una y otra vez las mismas instrucciones. Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que el código de la aplicación es mucho más largo porque muchas instrucciones están repetidas y si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

Las funciones son la solución a todos estos problemas, tanto en JavaScript como en el resto de lenguajes de programación. Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente. Las funciones en JavaScript se definen mediante la palabra reservada `function`, seguida del nombre de la función. Su definición formal es la siguiente:

```
function nombre_funcion() {...}
```

Las funciones más sencillas no necesitan ninguna información para producir sus resultados. Sin embargo, la mayoría de funciones de las aplicaciones reales deben acceder al valor de algunas variables para producir sus resultados. Las variables que necesitan las funciones se llaman *argumentos*. Antes de que pueda utilizarlos, la función debe indicar cuántos argumentos necesita y cuál es el nombre de cada argumento. Además, al invocar la función, se deben incluir los valores que se le van a pasar a la función. Los argumentos se indican dentro de los paréntesis que van detrás del nombre de la función y se separan con una coma (,).

Las funciones no solamente pueden recibir variables y datos, sino que también pueden devolver los valores que han calculado. Para devolver valores dentro de una función, se utiliza la palabra reservada `return`. Aunque las funciones pueden devolver valores de cualquier tipo, solamente pueden devolver un valor cada vez que se ejecutan. Si la función llega a una instrucción de tipo `return`, se devuelve el valor indicado y finaliza la ejecución de la función. Por tanto, todas las instrucciones que se incluyen después de un `return` se ignoran y por ese motivo la instrucción `return` suele ser la última de la mayoría de funciones.

2.4.3. DOM

DOM es el acrónimo de Document Object Model [Eguiluz 2009], [Hall 2010], [San Martin 2007], es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas. DOM permite acceder y

manipular las páginas XHTML como si fueran documentos XML. De hecho, DOM se diseñó originalmente para manipular de forma sencilla los documentos XML.

A pesar de sus orígenes, DOM se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (Java, PHP, JavaScript) y cuyas únicas diferencias se encuentran en la forma de implementarlo.

Una de las claves para crear páginas web dinámicas es el uso de manejadores de eventos. Esto permite ejecutar el código específico de un script como respuesta a las acciones del usuario o del sistema.

La mayor parte de los eventos se refieren a la Interfaz General de Usuario (GUI) del Navegador, como los movimientos del mouse, clics de los botones del mouse o pulsaciones de teclas y actualizaciones de campos input de formularios. Éstos son agregados usualmente a un elemento específico de la página. Otros se refieren a las acciones del navegador, tales como cuando un documento o imagen se ha cargado completamente.

En cualquier caso, todos los eventos siguen el mismo modelo. El DOM proporciona métodos para capturar eventos por lo que se pueden personalizar estas acciones en respuesta a ellos. También proporciona un objeto "Event" que contiene información específica para un evento dado que puede ser utilizada por el código para procesar los eventos.

Una de las tareas habituales en la programación de aplicaciones web con JavaScript consiste en la manipulación de las páginas web. De esta forma, es habitual obtener el valor almacenado por algunos elementos (por ejemplo los elementos de un formulario), crear un elemento (párrafos, <div>, etc.) de forma dinámica y añadirlo a la página, aplicar una animación a un elemento (que aparezca/desaparezca, que se desplace, etc.). Todas estas tareas habituales son muy sencillas de realizar gracias a DOM. Sin embargo, para poder utilizar las utilidades de DOM, es necesario "*transformar*" la página original.

Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular. Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular. Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla. El motivo por el que se muestra el funcionamiento de esta transformación interna es que condiciona el comportamiento de DOM y por tanto, la forma en la que se manipulan las páginas.

DOM transforma todos los documentos XHTML en un conjunto de elementos llamados *nodos*, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama "*árbol de nodos*". La siguiente página XHTML sencilla:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body></html>
```

Se transforma en el siguiente árbol de nodos (ver figura 2.8):

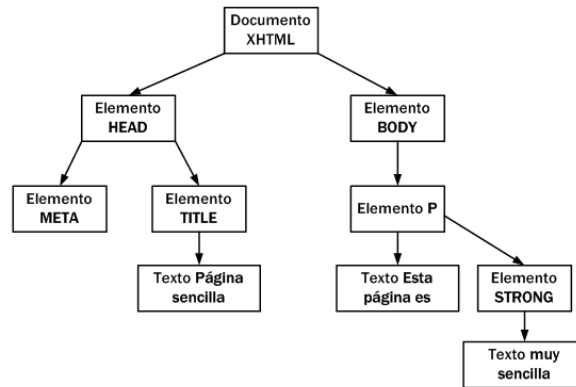


Figura 2.8. Árbol de nodos generado automáticamente por DOM a partir del código XHTML.

En la figura anterior, cada rectángulo representa un nodo DOM y las flechas indican las relaciones entre nodos. Dentro de cada nodo, se ha incluido su tipo y su contenido. La raíz del árbol de nodos de cualquier página XHTML siempre es la misma: un nodo de tipo especial denominado “*Documento*”. A partir de ese nodo raíz, cada etiqueta XHTML se transforma en un nodo de tipo “*Elemento*”. La conversión de etiquetas en nodos se realiza de forma jerárquica. De esta forma, del nodo raíz solamente pueden derivar los nodos HEAD y BODY. A partir de esta derivación inicial, cada etiqueta XHTML se transforma en un nodo que deriva del nodo correspondiente a su “*etiqueta padre*”. La transformación de las etiquetas XHTML habituales genera dos nodos: el primero es el nodo de tipo “*Elemento*” (correspondiente a la propia etiqueta XHTML) y el segundo es un nodo de tipo “*Texto*” que contiene el texto encerrado por esa etiqueta XHTML.

Así, la siguiente etiqueta XHTML:

`<title>Página sencilla</title>`

Genera los siguientes dos nodos (ver Figura 2.9):



Figura 2.9. Nodos generados automáticamente por DOM para una etiqueta XHTML sencilla.

De la misma forma, la siguiente etiqueta XHTML:

`<p>Esta página es <strong>muy sencilla</strong></p>`

Genera los siguientes nodos (ver Figura 2.10):

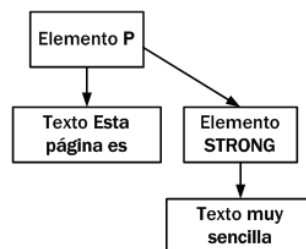


Figura 2.10. Nodos generados automáticamente por DOM para una etiqueta XHTML con otras etiquetas XHTML en su interior.

- Nodo de tipo “*Elemento*” correspondiente a la etiqueta `<p>`.

- Nodo de tipo "*Texto*" con el contenido textual de la etiqueta <p>.
- Como el contenido de <p> incluye en su interior otra etiqueta XHTML, la etiqueta interior se transforma en un nodo de tipo "*Elemento*" que representa la etiqueta y que deriva del nodo anterior. El contenido de la etiqueta genera a su vez otro nodo de tipo "*Texto*" que deriva del nodo generado por .

La transformación automática de la página en un árbol de nodos siempre sigue las mismas reglas:

- Las etiquetas XHTML se transforman en dos nodos: el primero es la propia etiqueta y el segundo nodo es hijo del primero y consiste en el contenido textual de la etiqueta.
- Si una etiqueta XHTML se encuentra dentro de otra, se sigue el mismo procedimiento anterior, pero los nodos generados serán nodos hijo de su etiqueta padre.

Como se puede suponer, las páginas XHTML habituales producen árboles con miles de nodos. Aun así, el proceso de transformación es rápido y automático, siendo las funciones proporcionadas por DOM las únicas que permiten acceder a cualquier nodo de la página de forma sencilla e inmediata.

La especificación completa de DOM define 12 tipos de nodos [DOM 2000], aunque las páginas XHTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- Document. Nodo raíz del que derivan todos los demás nodos del árbol.
- Element. Representa cada una de las etiquetas XHTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- Attr. Se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas XHTML, es decir, uno por cada par atributo=valor.
- Text. Nodo que contiene el texto encerrado por una etiqueta XHTML.
- Comment. Representa los comentarios incluidos en la página XHTML.

Los otros tipos de nodos existentes que no son frecuentemente utilizados y comentados son DocumentType, CDataSection, DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.

Al no apoyarse en un lenguaje de programación en particular, DOM facilita el diseño de páginas web activas, proporcionando una interfaz estándar para que otro software manipule los documentos.

2.5. Ingeniería de Software

La ingeniería de software [Pressman 2002] es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de software. La ingeniería de software incluye el análisis previo de la situación, el diseño del proyecto, el desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema. Cabe destacar que el proceso de desarrollo de software

implica lo que se conoce como ciclo de vida del software. Una vez que se completa este ciclo, se inicia con el mantenimiento del software, el cual se trata de una fase de la ingeniería de software donde se solucionan los errores descubiertos y se incorporan actualizaciones para hacer frente a los nuevos requisitos. El proceso de mantenimiento incorpora además nuevos desarrollos, para permitir que el software pueda cumplir con una mayor cantidad de tareas.

2.5.1. Etapas del ciclo de vida en la Ingeniería de Software

El fundamento de la ingeniería del software es la capa de proceso. El proceso de la ingeniería del software es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la ingeniería del software.

Los métodos de la ingeniería del software indican «cómo» construir técnicamente el software. Estos métodos incluyen: análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

El ciclo de vida del software es una sucesión de etapas o fases por las cuales pasa un software a lo largo de su vida. Crear un ciclo de vida permite detectar errores más rápido, mejorar la calidad del software, estimar los plazos de implementación y sus costos, etc.

El trabajo que se realiza en la ingeniería del software se puede dividir en tres etapas o fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto.

- La fase de definición se centra sobre el qué. Es decir, durante la definición, el que desarrolla el software intenta identificar qué información debe ser procesada, qué función y rendimiento se desea, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto. Por tanto, han de identificarse los requisitos clave del sistema y del software. Aunque los métodos aplicados durante la fase de definición variarán dependiendo del paradigma de ingeniería del software (o combinación de paradigmas) que se aplique, de alguna manera tendrán lugar tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto del software y análisis de los requisitos.
- La fase de desarrollo se centra en el cómo. Es decir, durante el desarrollo un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación (o lenguaje no procedimental) y cómo ha de realizarse la prueba. Los métodos aplicados durante la fase de desarrollo variarán, aunque las tres tareas específicas técnicas deberían ocurrir siempre: diseño del software, generación de código y prueba del software.
- La fase de mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes

del cliente. Durante la fase de mantenimiento se encuentran cuatro tipos de cambios:

- a) Corrección. Cuando ocurre o existen errores en el software los cuales no fueron encontrados en una fase de prueba, se lleva a cabo el mantenimiento correctivo, el cual cambia el software para corregir los defectos.
- b) Adaptación. Con el paso del tiempo, el entorno original para el que se desarrolló el software tiende a cambiar. El mantenimiento adaptativo produce modificaciones en el software para ajustarlo a dichos cambios.
- c) Mejora. Conforme se utilice el software, el cliente/usuario puede descubrir funciones adicionales que van a producir beneficios y las cuales no fueron contempladas inicialmente. El mantenimiento perfectivo lleva al software más allá de sus requisitos funcionales originales.
- d) Prevención. El software también se deteriora debido a los cambios ocurridos en alguno de los tipos de mantenimiento, es por esto que se lleva a cabo el mantenimiento preventivo también llamado reingeniería del software. En esencia, el mantenimiento preventivo hace cambios en programas de computadora a fin de que se puedan corregir, adaptar y mejorar más fácilmente.

Esta clasificación no es estricta, puede haber más o menos etapas de acuerdo al nivel de detalle conceptual que se desee. Además, las etapas no necesariamente se realizan en ese orden ni de forma serial.

La clasificación, el orden y otros aspectos del ciclo de vida dependen del modelo de ciclo de vida que se esté utilizando.

Las etapas anteriores descritas de forma genérica, se complementan con un número de actividades protectoras. Entre las actividades típicas de esta categoría se incluyen:

- Seguimiento y control del proyecto de software
- Revisiones técnicas formales
- Garantía de calidad del software
- Gestión de configuración del software
- Preparación y producción de documentos
- Gestión de reutilización
- Mediciones
- Gestión de riesgos

Estas actividades de protección se aplican a lo largo de todo el proceso del software, sin importar el modelo que se esté siguiendo.

2.5.2. Modelos de la Ingeniería de Software.

En la ingeniería del software, el software sigue un ciclo de vida para que pueda decirse que el software fabricado es de calidad. A continuación se describirán los ciclos de vida más importantes en la ingeniería del software.

2.5.2.1. Modelo lineal secuencial.

Llamado también ciclo de vida básico o modelo en cascada, el modelo lineal secuencial sugiere un enfoque sistemático, secuencial, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y

mantenimiento. La figura 2.11 muestra el modelo lineal secuencial para la ingeniería de software.

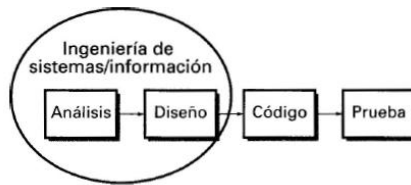


Figura 2.11. Modelo lineal secuencial.

El modelo lineal secuencial comprende las siguientes actividades:

- Ingeniería y modelado de Sistemas/Información. El trabajo comienza estableciendo requisitos de todos los elementos del sistema y asignando al software algún subgrupo de estos requisitos. Esta actividad es importante cuando el software debe interactuar con otros elementos como hardware, personas y bases de datos.
- Análisis: Es el proceso de reunión de requisitos se intensifica y se centra especialmente en el software. Toma como entrada una descripción en lenguaje natural de lo que quiere el cliente. Produce el SRD (*Software Requirements Document*).
- Diseño: El diseño del software es un proceso de muchos pasos que se centra en cuatro atributos distintos de programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo). Su entrada es el SRD y produce el SDD (*Software Design Document*).
- Código: Es la actividad en la cual se traduce el diseño a una forma legible por la máquina. A partir del SDD se producen módulos.
- Pruebas: El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos. A partir de los módulos probados se realiza la integración y pruebas de todo el sistema. El resultado de las pruebas es el producto final listo para entregar.

El modelo lineal secuencial es el paradigma más antiguo y más extensamente utilizado en la ingeniería del software. Entre los problemas que se encuentran algunas veces en el modelo lineal secuencial están:

- Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo.
- A menudo es difícil que el cliente exponga explícitamente todos los requisitos.
- El cliente debe tener paciencia, pues solo vera resultados hasta cuando el ciclo llegue a su fin.

El ciclo de vida clásico sigue siendo el modelo de proceso más extensamente utilizado por la ingeniería del software. Pese a tener debilidades, es significativamente mejor que un enfoque hecho al azar para el desarrollo del software.

2.5.2.2. Modelo de construcción de prototipos.

El modelo de construcción de prototipos (ver Figura 2.12) comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para

el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un «diseño rápido». El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer. La construcción de prototipos también puede ser problemática por las siguientes razones:

- El cliente ve lo que parece ser una versión de trabajo del software, sin saber que con la prisa de hacer que funcione no se ha tenido en cuenta la calidad del software o la facilidad de mantenimiento a largo plazo.
- El desarrollador, a menudo, hace compromisos de implementación para hacer que el prototipo funcione rápidamente.

Aunque pueden surgir problemas, la construcción de prototipos puede ser un paradigma efectivo para la ingeniería del software.

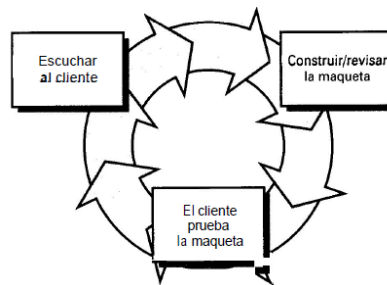


Figura 2.12. Modelo de construcción por prototipos.

2.5.2.3. Modelo espiral

El modelo en espiral, propuesto originalmente por Boehm en 1988. Es un proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Consiste en una serie de ciclos que se repiten. Cada uno tiene las mismas fases y cuando termina da un producto ampliado con respecto al ciclo anterior. En este sentido es parecido al modelo incremental, la diferencia importante es que tiene en cuenta el concepto de riesgo. Un riesgo puede ser muchas cosas: requisitos no comprendidos, mal diseño, errores en la implementación, etc., una representación típica de esta estructura se muestra en la figura (ver Figura 2.13):

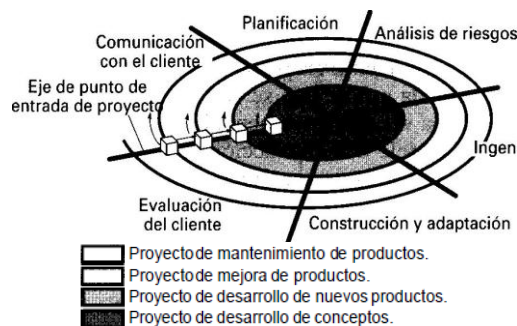


Figura 2.13. Modelo espiral típico

En cada iteración Boehm recomienda recopilar la siguiente lista de informaciones:

- Objetivos: Se hacen entrevistas a los clientes, se les hace rellenar cuestionarios, etc.
- Alternativas: Son las diferentes formas posibles de conseguir los objetivos.
- Riesgos: Lista de riesgos identificados.
- Resolución de riesgos: La técnica más usada es la construcción de prototipos.
- Resultados: Son lo que realmente ha ocurrido después de la resolución de riesgos.
- Planes: Lo que se va a hacer en la siguiente fase.
- Compromiso: Decisiones de gestión sobre como continuar.

Al terminar una iteración se comprueba si lo que se ha hecho efectivamente cumple con los requisitos establecidos, también se verifica que funciona correctamente. El propio cliente evalúa el producto. No existe una diferencia muy clara entre cuando termina el proyecto y cuando empieza la fase de mantenimiento. Cuando hay que hacer un cambio, este puede consistir en un nuevo ciclo.

2.5.2.4. Técnicas de cuarta generación.

Las técnicas de cuarta generación (T4G) se componen de muchas herramientas de software que tienen algo en común: todas facilitan al ingeniero del software la especificación de algunas características del software a alto nivel. Estas herramientas generan automáticamente el código fuente basándose en las especificaciones dadas. El paradigma T4G para la ingeniería del software se orienta hacia la posibilidad de especificar el software usando formas de lenguaje especializado o notaciones gráficas que describa el problema que hay que resolver en términos que los entienda el cliente. El paradigma T4G puede incluir todas o algunas de las siguientes herramientas: lenguajes no procedimentales de consulta a bases de datos, generación de informes, manejo de datos, interacción y definición de pantallas, generación de códigos, capacidades gráficas de alto nivel y capacidades de hoja de cálculo, y generación automatizada de HTML y lenguajes similares utilizados para la creación de sitios web usando herramientas de software avanzado. De la misma forma que otros modelos, T4G comienza con la reunión de requisitos, el cliente describe los requisitos, que son, a continuación, traducidos directamente a un prototipo operativo. Sin embargo, en la práctica no se puede hacer eso. El cliente puede que no esté seguro de lo que necesita; puede ser ambiguo en la especificación de hechos que le son conocidos, y puede que no sea capaz o no esté dispuesto a especificar la información en la forma en que puede aceptar una herramienta T4G. Por esta razón, el diálogo del cliente con el desarrollador descrito por los otros paradigmas sigue siendo una parte esencial del enfoque T4G. Para aplicaciones pequeñas, se puede ir directamente desde el paso de recolección de requisitos al paso de implementación, usando un lenguaje de cuarta generación no procedimental L4G (Lenguaje de Cuarta Generación).

2.5.2.5. Modelo fuente

Los tipos de ciclos de vida descritos hasta el momento son relativos al análisis y diseño estructurados. Pero los objetos tienen una particularidad, y es que están basados en componentes que se relacionan entre ellos a través de interfaces, o lo que es lo mismo, son mas modulares y por lo tanto el trabajo se puede dividir en un conjunto de mini proyectos. Además, hoy en día la tendencia es a reducir los riesgos, y en este sentido, el ciclo de vida

en cascada no proporciona muchas facilidades. Debido a todo esto, el ciclo de vida típico en una metodología de diseño orientado a objetos es iterativo e incremental.

El modelo fuente fue creado por Henderson-Sellers y Edwards en 1990. Es un tipo de ciclo de vida pensado para la orientación a objetos y posiblemente el más seguido. Un proyecto basado en un modelo orientado a objetos se divide en las siguientes fases:

1. Planificación del negocio
2. Construcción: Es la más importante y se divide a su vez en otras cinco actividades
 - Planificación.
 - Investigación.
 - Especificación.
 - Implementación.
 - Revisión.
3. Entrega.

La primera y la tercera fase son independientes de la metodología de desarrollo orientado a objetos. Además de las tres fases, existen dos periodos:

1. Crecimiento: Es el tiempo durante el cual se construye el sistema.
2. Madurez: Es el periodo de mantenimiento del producto. Cada mejora se planifica igual que el periodo anterior, es decir, con las fases de Planificación del negocio, Construcción y Entrega.

Cada clase puede tener un ciclo de vida sólo para ella debido a que cada una puede estar en una fase diferente en un momento cualquiera. La ventaja es que permite un desarrollo solapado e iterativo. En la figura 2.14, se muestra un esquema de este tipo de ciclo de vida.

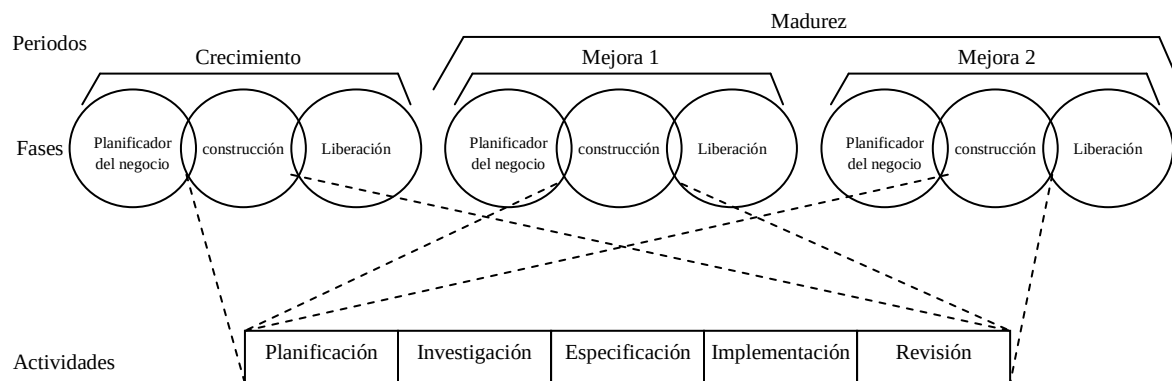


Figura 2.14. Modelo fuente.

2.5.3. Análisis y diseño orientado a objetos

La Ingeniería del Software y la Orientación a Objetos son dos áreas que al unirse producen una gran cantidad y variedad de técnicas y metodologías que pretenden facilitar la construcción de software. Las tecnologías basadas en objetos nos llevan a reutilizar componente de software, y la reutilización de estos componentes nos lleva a un desarrollo de software más rápido y a programas de mejor calidad.

El software orientado a objetos es más fácil de mantener debido a que su estructura es inherentemente poco acoplada. Esto lleva a que solo aquel componente o componentes que deban ser modificados en un momento dado, no afecten a otros. Además, los sistemas orientados a objetos son más fáciles de adaptar y más fácilmente escalables.

2.5.3.1. Definiciones

Un modelo evolutivo de proceso acoplado con un enfoque que fomente la reutilización de componentes de software es el mejor paradigma para la ingeniería del software Orientada a Objetos (OO). El proceso OO se mueve a través de una espiral evolutiva que comienza con la comunicación con el usuario. Es aquí donde se define el dominio del problema y se identifican las clases básicas del problema. La planificación y el análisis de riesgos establecen una base para el plan del proyecto OO. El trabajo técnico asociado con la ingeniería del software OO sigue el camino iterativo que maneja el modelo evolutivo de espiral. La ingeniería del software OO hace énfasis en la reutilización. Por lo tanto, en primera instancia las clases se buscan en una biblioteca (de clases OO si es que existente) antes de construirse. Cuando una clase no puede encontrarse en la biblioteca, esta debe construirse aplicando el análisis orientado a objetos (AOO), diseño orientado a objetos (DOO), programación orientada a objetos (POO) y pruebas orientadas a objetos (PROO). La nueva clase se pone en la biblioteca de tal manera que pueda ser reutilizada en el futuro.

A continuación se da una definición de los conceptos utilizados en la ingeniería de software orientada a objetos:

- Un objeto es una unidad atómica que encapsula un estado y un comportamiento, el objeto puede representar una entidad física o una entidad abstracta encapsulando datos (atributos) y los métodos (operaciones) que lo manipulan.
- Una clase es una descripción generalizada para una colección de objetos similares, que por definición, todos los objetos que existen dentro de una clase heredan sus atributos y las operaciones disponibles para manipularlos.
- Los atributos son aquellos valores que describen las características de una clase u objeto. Un atributo puede tomar un valor definido por un dominio enumerado, aunque en la mayoría de los casos, un dominio es simplemente un conjunto de valores específicos.
- Un objeto encapsula datos (representados como una colección de atributos) y los algoritmos que procesan estos datos. Estos algoritmos son llamados operaciones, métodos o servicios. Siempre que un objeto es estimulado por un mensaje, inicia algún comportamiento ejecutando una operación. Cada una de las operaciones encapsuladas por un objeto proporciona una representación de uno de los comportamientos del objeto.
- Los mensajes son el medio a través del cual interactúan los objetos. Un mensaje estimula la ocurrencia de cierto comportamiento en el objeto receptor. El comportamiento se realiza cuando se ejecuta una operación.
- El encapsulamiento es la característica que hace que los objetos contengan los datos y las operaciones que operan sobre ellos. Las principales características del encapsulamiento son:
 1. Ocultación de la información, reduce la propagación de efectos colaterales cuando ocurren cambios.
 2. Los objetos son auto contenidos, lo cual favorece la reutilización.
 3. Favorece el bajo acoplamiento y la alta cohesión.
- La herencia es una de las diferencias clave entre los sistemas convencionales y los sistemas OO. Una subclase hereda todos los atributos y operaciones asociadas con su superclase. Esto significa que todas las estructuras de datos y algoritmos

originalmente diseñados e implementados para la superclase están inmediatamente disponibles para la subclase.

- El polimorfismo es una característica que reduce en gran medida el esfuerzo necesario para extender un sistema OO. Y al igual que el encapsulamiento y la herencia, el polimorfismo es pieza fundamental en la reutilización.

En el mundo real los objetos se manifiestan de alguna de las siguientes formas:

- entidades externas (por ejemplo: otros sistemas, dispositivos, personas) que producen o consumen información a usar por un sistema.
- cosas (por ejemplo: informes, presentaciones, cartas, señales) que son parte del dominio de información del problema;
- ocurrencias o sucesos (por ejemplo: una transferencia de propiedad o la terminación de una serie de movimientos en un robot) que ocurren dentro del contexto de una operación del sistema;
- papeles o roles (por ejemplo: director, ingeniero, vendedor) desempeñados por personas que interactúan con el sistema;
- unidades organizacionales (por ejemplo: división, grupo, equipo) que son relevantes en una aplicación.
- lugares (por ejemplo: planta de producción o muelle de carga) que establecen el contexto del problema y la función general del sistema;
- estructuras (por ejemplo: sensores, vehículos de cuatro ruedas o computadoras) que definen una clase de objetos o, en casos extremos, clases relacionadas de objetos.

Coad y Yourdon [Coad 1991] sugieren seis características de selección a usar cada vez que un analista considera si incluye o no un objeto potencial en el modelo de análisis:

1. Información retenida: el objeto potencial será de utilidad durante el análisis solamente si la información acerca de él debe recordarse para que el sistema funcione.
2. Servicios necesarios: el objeto potencial debe poseer un conjunto de operaciones identificables que pueden cambiar de alguna manera el valor de **sus** atributos.
3. Atributos múltiples: durante el análisis de requisitos, se debe centrar la atención en la información principal (un objeto con un solo atributo puede, en efecto, ser Útil durante el diseño, pero seguramente será mejor presentado como un atributo de otro objeto durante la actividad de análisis).
4. Atributos comunes: puede definirse un conjunto de atributos para el objeto potencial, los cuales son aplicables a todas las ocurrencias del objeto.
5. Operaciones comunes: puede definirse un conjunto de operaciones para el objeto potencial, las cuales son aplicables a todas las ocurrencias del objeto.
6. Requisitos esenciales: entidades externas que aparecen en el espacio del problema y producen o consumen información esencial para la producción de cualquier solución para el sistema, serán casi siempre definidas como objetos en el modelo de requisitos.

Para ser considerado un objeto válido a incluir en el modelo de requisitos, un objeto potencial debe satisfacer todas (o casi todas) las características anteriores. La decisión de incluir objetos potenciales en el modelo de análisis es algo subjetivo, y una evaluación

posterior puede llegar a descartar un objeto o reincluirlo. Sin embargo, el primer paso del AOO debe ser la definición de objetos, y la consiguiente toma de decisiones.

2.5.3.2. Análisis orientado a objetos

El objetivo del Análisis Orientado a Objetos (AOO), es desarrollar una serie de modelos que describan el software de computadora al trabajar para satisfacer un conjunto de requisitos. Entre los modelos más utilizados se encuentran:

El método de Booch, este método abarca un micro proceso de desarrollo y un macro proceso de desarrollo. El nivel micro define un conjunto de tareas de análisis que se reaplican en cada etapa en el macro proceso. Por esto se mantienen un enfoque evolutivo. El micro proceso de desarrollo identifica clases y objetos y la semántica de dichas clases y objetos, define las relaciones entre clases y objetos y realiza una serie de refinamientos para elaborar el modelo del análisis.

El método de Rumbaugh, desarrolla una Técnica de Modelado de Objetos (OMT) para el análisis, diseño del sistema y diseño a nivel de objetos. La actividad de análisis crea tres modelos:

- El modelo de objetos (una representación de objetos, clases, jerarquías y relaciones)
- El modelo dinámico (una representación del comportamiento del sistema y los objetos)
- El modelo funcional (una representación a alto nivel del flujo de información a través del sistema similar al DFD).

El método de Jacobson, es una versión simplificada de *objectory*, un método patentado, también desarrollado por Jacobson. Este método se diferencia de los otros por la importancia que da al caso de uso como una descripción o escenario que describe cómo el usuario interactúa con el producto o sistema.

El método de Coad y Yourdon, se considera con frecuencia como uno de los métodos del AOO más sencillos de aprender. La notación del modelado es relativamente simple y las reglas para desarrollar el modelo de análisis son evidentes, a continuación se da una descripción resumida de este proceso:

- Identificar objetos, usando el criterio de «qué buscar».
- Definir una estructura de generalización-especificación.
- Definir una estructura de todo-parte.
- Identificar temas (representaciones de componentes de subsistemas).
- Definir atributos.
- Definir servicios.

El método de Wirfs-Brock, no hace una distinción clara entre las tareas de análisis y diseño. En su lugar, propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño, a continuación se describen brevemente las tareas relacionadas con este modelo:

- Evaluar la especificación del cliente.

- Usar un análisis gramatical para extraer clases candidatas de la especificación.
- Agrupar las clases en un intento de determinar superclases.
- Definir responsabilidades para cada clase.
- Asignar responsabilidades a cada clase.
- Identificar relaciones entre clases.
- Definir colaboraciones entre clases basándose en sus responsabilidades.
- Construir representaciones jerárquicas de clases para mostrar relaciones de herencia.
- Construir un gafo de colaboraciones para el sistema.

UML (*Unified Modeling Language*) es el resultado de la combinación de las mejores características de los métodos de diseño y análisis orientado a objetos de Grady Booch, James Rumbaugh e Ivar Jacobson. UML permite expresar un modelo de análisis utilizando una notación de modelado con unas reglas sintácticas, semánticas y prácticas. En UML, un sistema viene representado por cinco vistas diferentes que lo describen desde diferentes perspectivas. Cada vista se representa mediante un conjunto de diagramas. En UML están presentes las siguientes vistas:

- Vista del usuario: Representa el sistema (producto) desde la perspectiva de los usuarios (llamados actores en UML). El caso de uso es el enfoque elegido para modelar esta vista.
- Vista estructural: Los datos y la funcionalidad se muestran desde dentro del sistema, es decir, modela la estructura estática (clases, objetos y relaciones).
- Vista del comportamiento: Esta parte del modelo del análisis representa los aspectos dinámicos o de comportamiento del sistema. También muestra las interacciones o colaboraciones entre los diversos elementos estructurales descritos en las vistas anteriores.
- Vista de implementación: Los aspectos estructurales y de comportamiento se representan aquí tal y como van a ser implementados.
- Vista del entorno: Aspectos estructurales y de comportamiento en el que el sistema a implementar se representa.

En general, el modelo de análisis de UML se centra en las vistas del usuario y estructural.

Aunque la terminología y etapas del proceso para cada uno de estos métodos de AOO difieren, los procesos generales de AOO son en realidad muy similares. Para realizar un análisis orientado a objetos, se deben ejecutar las siguientes etapas genéricas:

1. Obtener los requisitos del cliente para el sistema.
2. Identificar escenarios o casos de uso.
3. Seleccionar clases y objetos usando los requisitos básicos como guías.
4. Identificar atributos y operaciones para cada objeto del sistema.
5. Definir estructuras y jerarquías que organicen las clases.
6. Construir un modelo objeto-relación.
7. Construir un modelo objeto-comportamiento.
8. Revisar el modelo de AOO con relación a los casos de uso/escenarios.

Para construir un sistema orientado a objetos se deben definir las clases que representan el problema a resolver, la forma en que las clases se relacionan e interactúan unas con otras, el funcionamiento interno de los objetos y la forma de comunicarse para que puedan trabajar juntos. El modelo de análisis conlleva una descripción de las características de las

clases que describen un sistema, ya que no se puede construir software hasta que se tenga un buen conocimiento del sistema, y es aquí en donde el AOO nos proporciona una forma de representar el conocimiento de los requisitos. El AOO comienza con una descripción de casos de uso, que es una descripción de escenarios sobre cómo interactúan los actores (personas, maquinas u otros sistemas) con el sistema a construir. El modelo de clases-responsabilidad-colaboración (CRC) traslada la información de los casos de uso a una representación de las clases y su colaboración con las otras clases. Las características estáticas y dinámicas se modelan entonces utilizando un lenguaje de modelado unificado o cualquier otro método.

2.5.3.3. Diseño orientado a objetos

El Diseño Orientado a Objetos (DOO) transforma el modelo de análisis creado usando AOO en un modelo de diseño que sirve como anteproyecto para la construcción de software. El DOO proporciona un diseño que alcanza diferentes niveles de modularidad. La mayoría de los componentes de un sistema, están organizados en subsistemas (módulos). Los datos y las operaciones que manipulan los datos se encapsulan en objetos, además, el DOO debe describir la organización específica de los datos, atributos y el detalle procedural de cada operación. Estos representan datos y piezas algorítmicas de un sistema OO y son los que contribuyen a la modularidad global. El DOO, reside en su capacidad para construir cuatro conceptos importantes de diseño de software: la abstracción, el ocultamiento de información, la independencia funcional y la modularidad. Todos los métodos de diseño procuran software que exhiba estas características fundamentales, pero sólo el DOO provee un mecanismo que permite al diseñador alcanzar las cuatro, sin complejidad ni compromiso.

Las cuatro capas de la pirámide (ver figura 2.15) de diseño OO son:

1. La capa de subsistema. Contiene una representación de cada uno de los subsistemas, para permitir al software conseguir sus requisitos definidos por el cliente e implementar la infraestructura que soporte los requerimientos del cliente.
2. La capa de clases y objetos. Contiene la jerarquía de clases, que permiten al sistema ser creado usando generalizaciones y cada vez especializaciones más acertadas. Esta capa también contiene representaciones.
3. La capa de mensajes. Contiene detalles de diseño, que permite a cada objeto comunicarse con sus colaboradores. Esta capa establece interfaces externas e internas para el sistema.
4. La capa de responsabilidades. Contiene estructuras de datos y diseños algorítmicos, para todos los atributos y operaciones de cada objeto.

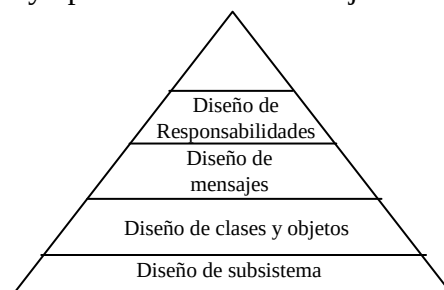


Figura 2.15. Pirámide del diseño OO.

La Figura 2.16 muestra la relación entre el modelo de análisis OO y el modelo de diseño OO que se derivará de él.

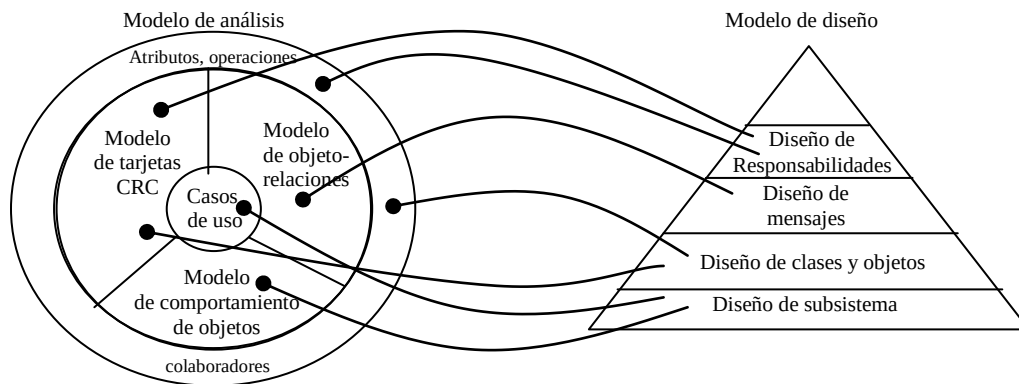


Figura 2.16. Transformación de un modelo de análisis OO a un modelo de diseño OO.

Anteriormente se mencionó como Grady Booch, James Rumbaugh e Ivar Jacobson, combinaron las mejores cualidades de sus métodos personales de análisis y diseño orientado a objetos, en un método unificado llamado UML, el cual se organiza en dos actividades mayores: diseño del sistema y diseño de objetos.

El diseño de sistema se centra en arquitectura de software y definición de subsistemas.

El diseño de objetos describe objetos, hasta un nivel en el cual puedan ser implementados, en un lenguaje de programación.

El diseño de sistemas y objetos en UML se extiende para considerar el diseño de interfaces, administración de datos con el sistema que se va a construir y administración de tareas para los subsistemas que se han especificado.

2.5.3.4. UML

UML (*Unified Modeling Language -Lenguaje de Modelado Unificado-*) [Rumbaugh 2000], [Schmuller 2002] permite modelar sistemas de información y su objetivo es lograr modelos que además de describir con cierto grado de formalismo tales sistemas, puedan ser entendidos por los clientes o usuarios de aquello que se modela. UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar un sistema de software.

UML no es un método de desarrollo, pues no dice cómo pasar del análisis al diseño y de este al código. No son una serie de pasos que lleven a producir código a partir de unas especificaciones. UML al no ser un método de desarrollo es independiente del ciclo de desarrollo que se siga, puede encajar en un tradicional ciclo en cascada, o en un evolutivo ciclo en espiral.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. UML describe lo que supuestamente hará un sistema, pero no dice como implementar dicho sistema. UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema.

La estructura estática o estructural define los tipos de objetos importantes para un sistema y para su implementación así como las relaciones entre los objetos, dentro de esta clasificación se encuentran: diagramas de clase, diagramas de objetos, diagramas de componentes y diagramas de implementación.

El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos, los diagramas que se encuentran en esta clasificación son: diagramas de colaboración, diagramas de casos de uso, diagramas de secuencia, diagramas de actividad y diagramas de estado.

Diagrama de casos de uso: Los casos de uso no son parte del diseño, sino parte del análisis. De forma que al ser parte del análisis nos ayudan a describir qué es lo que el sistema debe hacer. Los casos de uso son qué hace el sistema desde el punto de vista del usuario. Es decir, describen un uso del sistema y cómo este interactúa con el usuario. Lo primordial de los casos de uso no son los diagramas (ver Figura 2.18), sino el documento (ver Figura 2.17) que describe el caso de uso, en este documento se explica la forma de interactuar entre el sistema y el usuario. Omitiendo los campos evidentes como nombre, autor, fecha y descripción; los actores son aquellos que interactúan con el sistema. Las precondiciones son los hechos que se han de cumplir para que el flujo de evento se pueda llevar a cabo. Luego tenemos el flujo de eventos, que corresponde a la ejecución normal y exitosa del caso de uso. Los flujos alternativos son los que nos permiten indicar qué es lo que hace el sistema en los casos menos frecuentes e inesperados. Por último, las poscondiciones son los hechos que se ha de cumplir si el flujo de eventos normal se ha ejecutado correctamente. Los casos de uso se pueden detallar más o menos dependiendo de la necesidad del problema.

Nombre:	Crear mensaje foro
Autor:	Joaquin Gracia
Fecha:	22/09/2010
Descripción: Permite crear un mensaje en el foro de discusión.	
Actores: Usuario de internet logeado.	
Precondiciones: El usuario debe haberse logeado en el sistema.	
Flujo Normal: 1. El actor pulsa sobre el botón para crear un nuevo mensaje. 2. El sistema muestra una caja de texto para introducir el título del mensaje y una zona de mayor tamaño para introducir el cuerpo del mensaje. 3. El actor introduce el título del mensaje y el cuerpo del mismo. 4. El sistema comprueba la validez de los datos y los almacena.	
Flujo Alternativo: 1 El sistema comprueba la validez de los datos, si los datos no son correctos, se avisa al actor de ello permitiéndole que los corrija	
Poscondiciones: El mensaje ha sido almacenado en el sistema.	

Figura 2.17. Ejemplo de un documento de casos de uso.

Cuando se comienza a tener un número considerable de casos de uso, no resulta nada fácil situarlos y relacionarlos. Entonces se tiene la necesidad de una visión general del asunto, y en estos momentos es cuando los diagramas de casos de uso son de utilidad. En los diagramas de casos de uso los muñecos son los actores y los círculos son los casos de uso. Así que se dibuja un muñeco por actor y un círculo por cada caso de uso y se enlazan con líneas cuando haya una relación entre ellos. Con esto se consigue una visión general de cómo los diferentes actores interactúan con los distintos casos de uso.

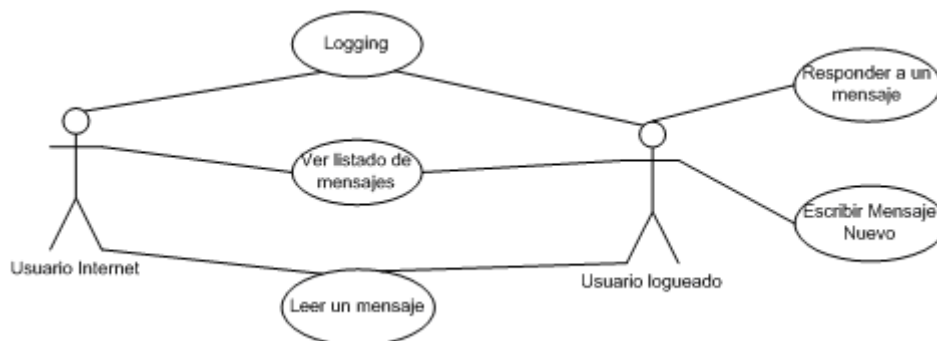


Figura 2.18. Ejemplo de un diagrama de caso de uso correspondiente al documento de caso de uso de la Figura 2.17.

Diagrama de clases: Muestra las clases (descripción de objetos que comparten características comunes) que componen el sistema y como se relacionan entre sí. En UML, un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. El área superior contiene el nombre de la clase, el área central contiene los atributos o propiedades, y el área inferior, las acciones, procedimientos, métodos o funciones. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí. La figura 2.19 muestra la forma general el diagrama de clases.

Cuando las clases se conectan entre sí de forma conceptual, esta conexión se conoce como asociación (ver figura 2.19a).

Se le llama “multiplicidad” a la cantidad de objetos de una clase que se relacionan con un objeto de la clase asociada, esta puede ser de uno a uno, de uno a muchos o de muchos a uno. Para representar los números en el diagrama, se colocan sobre la línea de asociación junto a la clase como se muestra en la figura 2.19b.

Cuando un objeto de una clase tiene que seleccionar un objeto particular de otro tipo para cumplir con un papel en la asociación, la primera clase deberá atenerse a un atributo en particular para localizar al objeto adecuado. Normalmente este atributo es un identificador que puede ser un número de identidad. Cuando sucede esto, se dice que es una “asociación calificada” (ver Figura 2.19c).

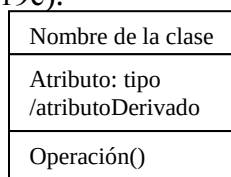


Figura 2.19. Forma general de un diagrama de clases.



Figura 2.19a. Representación general de la asociación entre dos clases.



Figura 2.19b. Representación general de la multiplicidad entre dos clases.

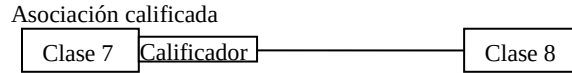


Figura 2.19c. Representación general de una asociación calificada.

En UML a la herencia también se reconoce como “generalización”. Una clase (la clase secundaria o subclase) puede heredar los atributos y operaciones de otra (la clase principal o superclase). La clase principal (o madre) es más genérica que la secundaria (o hija). La jerarquía de la herencia no tiene que finalizar en dos niveles: una clase secundaria puede ser la principal para otra clase secundaria. En UML se representa la herencia con una línea que conecte a la clase principal con la secundaria. En la parte de la línea que se conecta con la clase principal, se coloca un triángulo sin rellenar que apunte a la clase principal, así como se muestra en la Figura 2.19d.

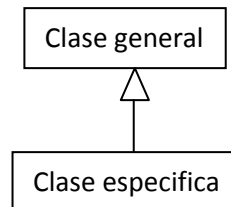


Figura 2.19d. Forma general de la “generalización”.

En otro tipo de relación, una clase utiliza a otra. A esto se le llama “dependencia” (ver Figura 2.19e). El uso más común de una dependencia es mostrar que la firma de la operación de una clase utiliza a otra clase.

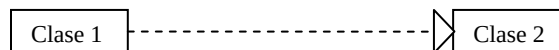


Figura 2.19e Forma general de la dependencia entre clases.

Diagrama de objetos: Muestra una serie de objetos (instancias de clases) y sus relaciones. Estos diagramas se enfocan en la perspectiva de casos reales. Por lo tanto, partiendo del hecho que un objeto es una instancia de clase, tal como se define en la conceptualización básica de la programación orientada a objetos, en UML la representación de un diagrama de objetos se hace de tal forma que teniendo ya una clase, el símbolo del objeto es un rectángulo, pero con el nombre subrayado. El nombre de la instancia específica se encuentra a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha. La figura 2.20 muestra gráficamente un diagrama de objeto.

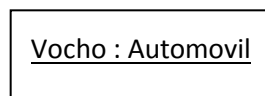


Figura 2.20. Ejemplo de un diagrama de objetos.

Diagrama de secuencia: Hace énfasis en la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado, aunque por orden lo usual es colocar los objetos de izquierda a derecha y en la parte superior. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba hacia abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren. Cada línea de vida de un objeto es una línea discontinua que se desplaza hacia abajo del objeto. Una línea continua con una punta de flecha conecta a una línea de vida con otra, y representa un mensaje de un objeto a otro. El tiempo se inicia en la parte superior y continúa hacia abajo. Aunque un actor es el que normalmente inicia la secuencia, su símbolo no es parte del conjunto de símbolos del diagrama de secuencias. No necesariamente se debe especificar el tiempo de manera explícita (como en segundos, minutos, horas, días, etc.), aunque se puede hacer si resulta necesario o conveniente. En todo caso lo que siempre se tiene que hacer es ubicar correctamente en el eje vertical la secuencia correcta de eventos que se deben ir dando de forma cronológica o en una línea de tiempo. La figura 2.21 muestra gráficamente un diagrama de secuencia.

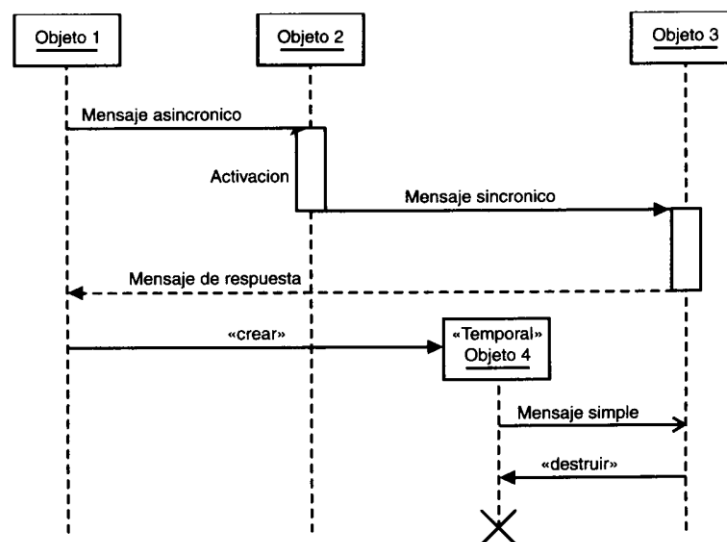


Figura 2.21. Ejemplo de diagrama de secuencia.

Diagrama de colaboración: En este tipo de diagramas se muestra una interacción organizada, basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos. A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia. Aunque se muestran los mensajes que se envían los objetos entre sí, por lo general se evita la multiplicidad de mensajes dado que podría ser fuente de confusión. En la representación de los mensajes, la flecha que se dibuja cerca de la línea de asociación entre dos objetos, apunta al objeto receptor. El mensaje finaliza con un par de paréntesis, dentro de los cuales se colocan los parámetros (en caso de haber alguno) con los que funcionará la operación. La figura 2.22 muestra un ejemplo general de este tipo de diagramas. Se puede convertir cualquier diagrama de secuencias en diagrama de colaboraciones y viceversa.

Para ello se agrega una cifra a la etiqueta de un mensaje, la cual corresponde a la secuencia propia del mensaje. La cifra y el mensaje se separan mediante dos puntos (:).

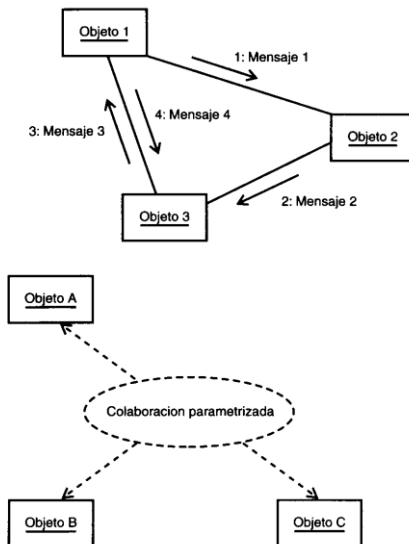


Figura 2.22. Ejemplo de diagrama de colaboración.

Diagrama de estados: Este diagrama sirve para analizar los cambios de estado de los objetos. Muestra los estados, eventos, transiciones y actividades de los diferentes objetos siendo útiles en sistemas que reaccionan a eventos. Conforme un sistema interactúa con los usuarios y (posiblemente) con otros sistemas, los objetos que lo conforman pasan por cambios necesarios para ajustar las interacciones. Un cambio en un sistema se da debido a que los objetos que componen dicho sistema modificaron su estado como respuesta a los sucesos y al tiempo. Un diagrama de estados también se conoce como un "motor de estado". El ícono para el estado, como se aprecia en la Figura 2.x, es un rectángulo de vértices redondeados, y el símbolo de una transición es una línea continua y una punta de flecha. El círculo relleno se interpreta como el punto inicial de una secuencia de estados, y la diana representa al punto final.

Se puede subdividir el símbolo de la Figura 2.23 en áreas que muestren el nombre, variables y actividades del estado tal y como lo muestra la Figura 2.24.

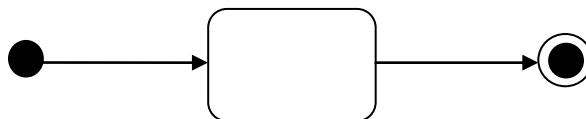


Figura 2.23. Ejemplo de diagrama de estado.

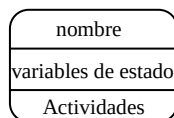


Figura 2.24. Adición de detalles al icono de estado.

Diagrama de actividades: Este diagrama es muy parecido a los viejos diagramas de flujo, pues muestra los pasos (conocidos como actividades) así como puntos de decisión y bifurcaciones. Lo que hacen es mostrar una visión simplificada de lo que ocurre durante una operación o proceso. Se puede decir también que es una extensión del diagrama de

estados. El diagrama de estados (ver Figura 2.25) muestra los estados de un objeto y representa las actividades como flechas que conectan a los estados. Por su parte, el diagrama de actividades resalta, precisamente, las actividades. Cada actividad se representa por un rectángulo con las esquinas redondeadas. El procesamiento dentro de una actividad se lleva a cabo y, al realizarse, se continúa con la siguiente actividad. Una flecha representa la transición de una a otra actividad. Al igual que el diagrama de estados, el de actividad cuenta con un punto inicial (representado por un círculo relleno) y uno final (representado por una diana). Es posible también modelar actividades que serán ejecutadas al mismo tiempo (es decir, de forma concurrente) y que luego se reúnan. Para representar esto, se utiliza una línea gruesa perpendicular a la transición y las rutas parten de ella. Para representar la reincorporación, ambas rutas apuntan a otra línea gruesa.

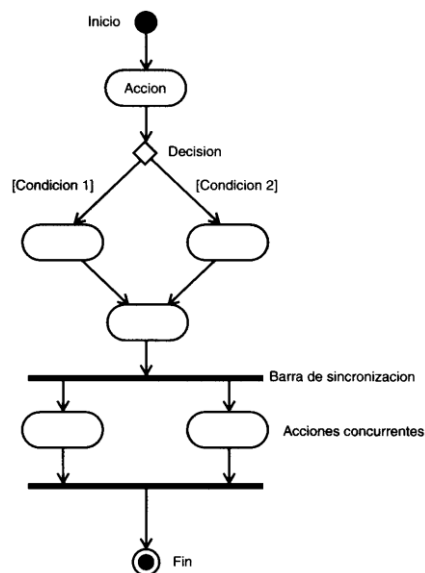


Figura 2.25. Ejemplo de diagrama de actividades.

Diagrama de componentes: Un componente de software es una parte física de un sistema, y se encuentra en la computadora, no en la mente del analista. Ejemplos de componentes son tablas, archivos de datos, ejecutables, bibliotecas de vínculos dinámicos, documentos, etc.

Lo que contiene un diagrama de componentes es lógicamente componentes, interfaces y relaciones, aunque también pueden aparecer otros tipos de símbolos. El símbolo principal de un diagrama de componentes es un rectángulo que tiene otros dos sobrepuestos en su lado izquierdo, con el nombre del componente dentro del rectángulo más grande, como se muestra en la Figura 2.26. Las flechas discontinuas que como se explicó anteriormente en los diagramas de casos de uso, representan relaciones de dependencia, donde la dirección de la flecha apunta a la clase dependiente, que en este caso sería el componente dependiente, y por consecuencia en el extremo donde no hay flecha se encuentra el componente independiente.

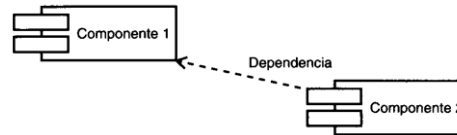


Figura 2.26. Ejemplo de diagrama de componentes.

Diagrama de implementación: Muestra los dispositivos que se encuentran en un sistema y su distribución en el mismo.

Existen herramientas que facilitan el proceso de creación de software, a dichas herramientas se les denomina “CASE” (*Computer Assisted Software Engineering - Ingeniería de software Asistida por Computadora-*). La tecnología CASE mantiene una disciplina de la ingeniería automatizada para el desarrollo de software, con metodologías estructuradas y herramientas automatizadas.

CASE se puede definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de desarrollo de un software. CASE es también definido como el conjunto de métodos, utilidades y técnicas que facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

La mejor razón para la creación de estas herramientas fue el incremento en la velocidad de desarrollo de los sistemas. Las herramientas CASE también permiten a los analistas tener más tiempo para el análisis y diseño y minimizar el tiempo para codificar y probar.

Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software, algunas de ellas son:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la Base de Datos.
- Generar estructuras de código.

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad.

La mejora de calidad se consigue reduciendo sustancialmente muchos de los problemas de análisis y diseño, inherentes a los proyectos de mediano y gran tamaño.

La mejora de productividad se consigue a través de la automatización de determinadas tareas, como la generación de código y la reutilización de objetos o módulos.

Las herramientas CASE, en función de las fases del ciclo de vida abarcadas, se pueden agrupar de la forma siguiente:

- Herramientas integradas, I-CASE (Integrated CASE, CASE integrado): Abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.
- Herramientas de alto nivel, U-CASE (Upper CASE - CASE superior) o front-end, orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.

- Herramientas de bajo nivel, L-CASE (Lower CASE - CASE inferior) o back-end, dirigidas a las últimas fases del desarrollo: construcción e implantación.
- Juegos de herramientas o Tools-Case, son el tipo más simple de herramientas CASE, pues automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

Seleccionar una Herramienta CASE no es una tarea simple. No existe una “mejor” herramienta respecto de otra. Las fallas o las respuestas satisfactorias están en relación con las expectativas. Si el proceso de evaluación y selección de las Herramientas CASE falla, entonces la herramienta no cumplirá con las especificaciones o expectativas del negocio, esto puede ocurrir durante el proceso de implementación o ejecución del producto.

Capítulo 3. Análisis y Diseño de las herramientas colaborativas de SisAAC

Dentro de la creación de software, las fases de análisis y diseño son de suma importancia ya que a partir de ellos se define la forma en que se modelara el sistema a desarrollar.

Para llevar a cabo la implementación tanto de la plataforma de SisAAC como de las herramientas colaborativas, se utilizó el modelo en espiral, ya que este cubre las mejores características tanto del modelo clásico como el de creación de prototipos. Además se hizo uso de UML, ya que este lenguaje de modelado se adapta a cualquier modelo y sus representaciones gráficas hacen más sencillo de comprender y realizar un buen análisis y diseño de las herramientas en especial porque es orientado a objeto, lo cual hace más sencillo el desarrollo del sistema ya que se utilizó un lenguaje de programación orientado a objetos. Y si tomamos en cuenta que el modelo en espiral nos permite regresar a la fase del análisis y diseño tantas veces como se requiera, entonces obtenemos un sólido y cómodo marco de trabajo.

3.1. Análisis

El análisis que se presenta, se aplica a cuatro herramientas colaborativas que fueron desarrolladas para la plataforma de SisAAC desarrollada en el trabajo de tesis denominado “Una Plataforma de Apoyo al Aprendizaje basada en Servicios Web”.

Las herramientas colaborativas propuestas para desarrollar son:

- Foro: Esta herramienta brindara al usuario un entorno en el cual pueda discutir con sus condiscípulos acerca de temas de interés común.
- Blog: Herramienta con la cual el usuario pueda mostrar a sus contactos una serie de imágenes y comentarios relevantes para él.
- Buzón de mensajes: Esta herramienta tiene la finalidad de permitir la comunicación directa y personal entre un usuario y sus contactos.
- Espacio para compartir archivos: Esta herramienta proporcionara al usuario la facilidad para compartir información con todos los usuarios registrados en el sistema de SisAAC.

Para el análisis de requisitos se utilizaron casos de uso de UML para tener un mejor panorama de cómo un usuario tendrá interacción con las herramientas propuestas para que el diseño sea lo más apegado a los requerimientos. Además del documento de caso de uso también se presenta su diagrama correspondiente lo cual facilitara la comprensión de los actores con las herramientas.

A continuación se describen los casos de uso más relevantes en el análisis de las herramientas colaborativas.

3.1.1. Casos de uso de las herramientas colaborativas de SisAAC

El caso de uso “**Esquema de herramientas colaborativas de SisAAC**” (ver Figura 3.1), presenta de forma general las opciones de herramientas a utilizar por un usuario una vez que este registrado y haya ingresado al sistema SisAAC. A si mismo se muestra su

diagrama correspondiente en el cual se tiene una mejor apreciación del usuario y las herramientas (ver Figura 3.1a).

Nombre:	Esquema de las herramientas colaborativas de SisAAC.
Fecha:	22/09/2010
Descripción:	Uso de las herramientas colaborativas para SisAAC.
Actores:	Usuario de SisAAC.
Precondiciones:	Ser usuario registrado y haber ingresado a SisAAC.
Flujo Normal:	1. Elegir una herramienta colaborativa (Foro, Archivos, Blog, Buzón de mensajes). 2. Terminar.
Flujo Alternativo:	1. Si no se elige una herramienta colaborativa, el usuario solo tendrá la opción de salir de la aplicación o cambiar su información personal.
Poscondiciones:	El usuario tiene acceso al uso de la herramienta colaborativa elegida.

Figura 3.1 Documento de caso de uso del esquema de las herramientas colaborativas.

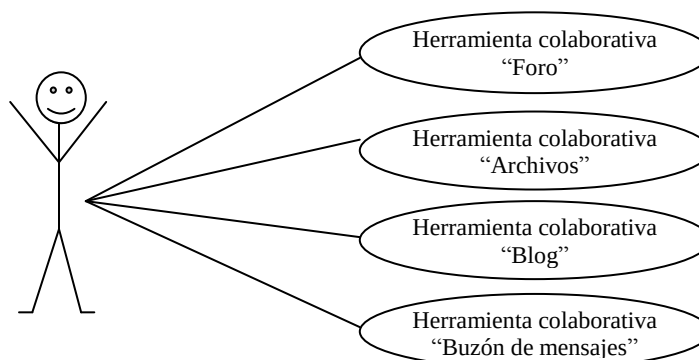


Figura 3.1a Diagrama de caso de uso de las “Herramientas Colaborativas de SisAAC”.

El caso de uso “**Foro**” (ver Figura 3.2) muestra las opciones que un usuario tiene al ingresar a esta herramienta tales como: crear o ingresar a un foro, y hacer o consultar las preguntas o respuestas de un foro. Su diagrama (ver Figura 3.2a) muestra claramente como el usuario puede realizar estas acciones.

Nombre:	Herramienta colaborativa “Foro” de SisAAC.
Fecha:	22/09/2010
Descripción:	Uso de la herramienta colaborativa “Foro” para SisAAC.
Actores:	Usuario de SisAAC.
Precondiciones:	Pertenecer a un grupo de SisAAC.
Flujo Normal:	1. Crear o Ingresar a un foro. 2. Hacer una pregunta dentro del foro. 3. Responder a una pregunta hecha en el foro. 4. Consultar las preguntas y respuestas del foro.
Flujo Alternativo:	1. Si no se rellenan los campos necesarios se envía un mensaje solicitándolos.
Poscondiciones:	El usuario tiene acceso al uso de la herramienta “Foro” de SisAAC.

Figura 3.2 Documento de caso de uso del foro.

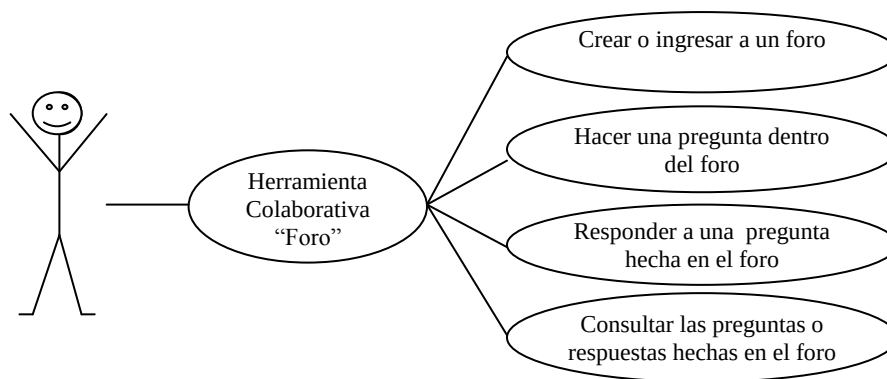


Figura 3.2a Diagrama de caso de uso “Foro”.

El caso de uso “**Espacio para compartir archivos**” (ver Figura 3.3), describe que un usuario puede subir, bajar o eliminar un archivo que este siendo compartido con otros usuarios. Al igual que los anteriores diagramas de casos de uso, este también representa gráficamente la interacción del usuario con esta herramienta (ver Figura 3.3a).

Nombre:	Herramienta colaborativa “Espacio para compartir Archivos” de SisAAC.
Fecha:	22/09/2010
Descripción:	Uso de la herramienta colaborativa “Espacio para compartir Archivos” de SisAAC.
Actores:	Usuario de SisAAC.
Precondiciones:	Pertenecer a un grupo de SisAAC y haber ingresado a SisAAC.
Flujo Normal:	1. Subir un archivo al grupo. 2. Bajar un archivo del grupo. 3. Eliminar un archivo.
Flujo Alternativo:	1. Si no se rellenan los campos necesarios se envía un mensaje solicitándolos. 2. Si se excede el tamaño permitido se envía un mensaje de error. 3. Si el archivo seleccionado es de un formato no permitido se envía un mensaje de error.
Poscondiciones:	El usuario tiene acceso al uso de la herramienta “Archivos” de SisAAC.

Figura 3.3 Documento de caso de uso del espacio para compartir archivos.

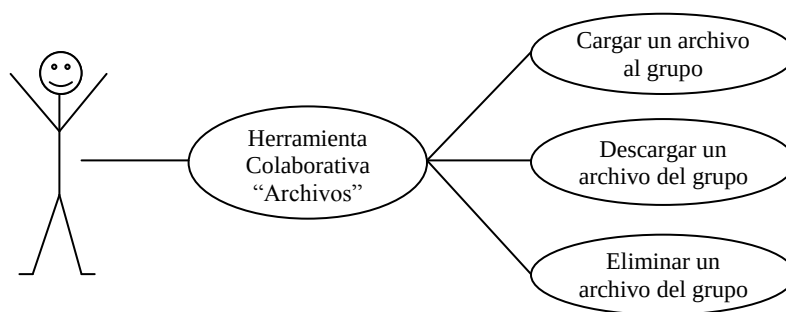


Figura 3.3a Diagrama de caso de uso “Espacio para compartir Archivos”.

El caso de uso de la herramienta colaborativa “**Blog**” (ver Figura 3.4) muestra que un usuario puede agregar imágenes y comentarios, así como ingresar al blog de otro usuario. El diagrama asociado a este caso de uso (ver Figura 3.4a) muestra de forma más sencilla lo ya descrito.

Nombre:	Herramienta colaborativa “Blog” de SisAAC.
Fecha:	22/09/2010
Descripción: Uso de la herramienta colaborativa “Blog” para SisAAC.	
Actores: Usuario de SisAAC.	
Precondiciones: Ser usuario de SisAAC, haber ingresado a SisAAC y tener al menos un contacto.	
Flujo Normal: 1. Ingresar al blog. 2. Agregar una imagen y comentario al blog. 3. Agregar un comentario al blog. 4. Consultar comentarios de contactos en el blog. 5. Ingresar al blog de un contacto. 6. Agregar un comentario en el blog de un contacto.	
Flujo Alternativo: 1. Si no se cuenta con contactos, estos no aparecerán en el blog. 2. Si un usuario no es contacto de otro, ninguno tendrá acceso al blog del otro. 3. Si al agregar una imagen no se agrega su comentario correspondiente, se enviara un mensaje solicitándolo. 4. Si al agregar un comentario el campo esta vacio, se enviara un mensaje de error. 5. Si al agregar un mensaje en el blog de un contacto no se rellena el campo necesario, se enviara un mensaje de error.	
Poscondiciones: El usuario tiene acceso al uso de la herramienta “Blog” de SisAAC.	

Figura 3.4 Documento de caso de uso del blog.

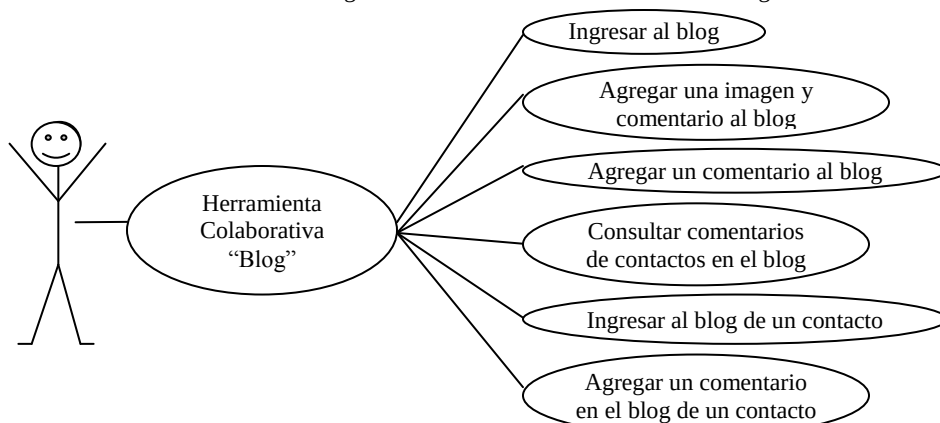


Figura 3.4a Diagrama de caso de uso “Blog”.

El caso de uso de la herramienta colaborativa “**Buzón de mensajes**” (ver Figura 3.5), nos dice que un usuario del sistema de SisAAC puede comunicarse con otro siempre y cuando el primero sea contacto del segundo o viceversa, además de las diferentes opciones que el usuario tiene para administrar tantos sus mensajes como sus contactos. El diagrama correspondiente a este caso de uso (ver Figura 3.5a), a lo mismo que los anteriores diagramas tiene como finalidad ser mas descriptivo de la interacción del usuario con la herramienta.

Nombre:	Herramienta colaborativa “Buzón de mensajes” de SisAAC.
Fecha:	22/09/2010
Descripción: Uso de la herramienta colaborativa “Buzón de mensajes” para SisAAC.	
Actores: Usuario de SisAAC.	
Precondiciones: Ser usuario de SisAAC, haber ingresado a SisAAC y tener al menos un contacto.	
Flujo Normal: 1. Agregar un contacto. 2. Eliminar un contacto. 3. Crear un mensaje. 4. Eliminar un mensaje. 5. Consultar los mensajes.	
Flujo Alternativo: 1. Si no se rellenan los campos necesarios para enviar un mensaje a un contacto, se envía un mensaje solicitando todos los datos necesarios.	
Poscondiciones: El usuario tiene acceso al uso de la herramienta “Buzón de mensajes” de SisAAC.	

Figura 3.5 Documento de caso de uso del buzón de mensajes.

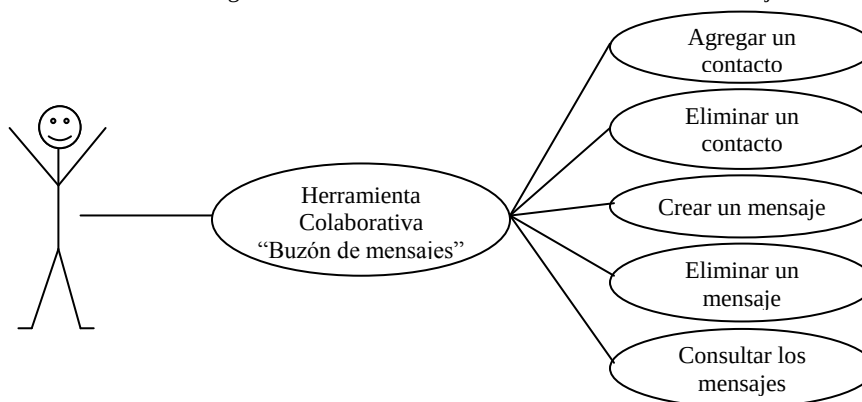


Figura 3.5a Diagrama de caso de uso “Buzón de mensajes”.

3.2. Diseño

En esta sección, se describe el diseño de los principales elementos que permiten la implementación de las distintas herramientas colaborativas para el sistema de SisAAC, por tanto, se presenta el diagrama de clases, los diagramas de secuencia más representativos y el diagrama entidad-relación, así como se explica brevemente la normalización de la base de datos presentada en el diagrama entidad-relación.

3.2.1. Diagrama de Clases

Un diagrama de clases como ya se menciono anteriormente en la sección 2.5.3.4., sirve para visualizar las relaciones entre las clases que involucran el sistema.

En esta sección, se describen las clases que están involucradas en el diseño de las herramientas colaborativas, tomando en cuenta que cada clase está representando realmente a cada uno de los Servicios Web que serán implementados. Por tanto, al tratarse de Servicios Web la relación que guarda cada clase con otra se lleva a cabo en el momento en que un servicio web invoca a otro a través de alguno de sus métodos.

Los conjuntos de clases que intervienen en el funcionamiento de cada una de las herramientas colaborativas utilizan un color en específico para diferenciarse de las otras herramientas.

El diagrama de clases que se presenta a continuación (ver Figura 3.6), contiene algunas clases que no forman parte de una herramienta en específico, sino que algunas herramientas hacen uso de las mismas clases para su buen funcionamiento individual.

A continuación se describen las cuatro herramientas colaborativas con respecto a las clases relacionadas con estas y también se hace mención de las clases que están relacionadas con varias herramientas.

Herramienta colaborativa “Foro”. Las clases relacionadas con esta herramienta colaborativa y que se pueden apreciar en la Figura 3.6 distinguiéndose en el diseño por el color naranja son:

- Perfil: Es la clase con la cual están relacionadas las clases principales de cada herramienta pues a través de sus métodos se obtienen de la base de datos alguna información requerida por la herramienta que la invoca.
- Grupos: Esta clase que no solo da soporte a la clase foro permite la administración de un sin número de instancias de la clase foro. Esta clase depende de la clase “Perfil”
- Foro: Las instancias de esta clase representan un foro de discusión en específico. Esta clase depende de las clases “Perfil” y “Grupos”.
- Preguntas: Los métodos de esta clase permiten que un usuario pueda generar y almacenar una pregunta en la base de datos. Esta clase depende de las clases “Perfil” y “Foro”.
- Respuestas: Esta clase depende directamente de la clase “Preguntas” pues a partir de esta, el usuario puede dar una respuesta a una pregunta en específico la cual será almacenada en la base de datos. Esta clase depende de las clases “Perfil” y “Preguntas”.

Herramienta colaborativa “Espacio para compartir archivos”. Las clases relacionadas con esta herramienta colaborativa y que se pueden apreciar en la Figura 3.6 distinguiéndose en el diseño por el color azul son:

- Perfil: Tiene la misma función que en las otras herramientas y que se describió anteriormente.
- Grupos: Al igual que la clase foro, esta herramienta también depende de los métodos de esta clase. Esta clase depende de la clase “Perfil”.
- Archivos: A través de los métodos de esta clase el usuario tendrá acceso para almacenar información en la base de datos. Esta clase depende de las clases “Perfil” y “Grupos”.

Herramienta colaborativa “Buzón de mensajes”. Las clases relacionadas con esta herramienta colaborativa y que se pueden apreciar en la Figura 3.6 distinguiéndose en el diseño por el color verde son:

- Perfil: Tiene la misma función que en las otras herramientas y que se describió anteriormente.
- Contactos: Los métodos de esta clase permiten al usuario agregar o eliminar un contacto de su lista de contactos. Esta clase depende de la clase “Perfil”. Esta clase depende de la clase “Perfil”.
- MensajesUsuarios: A través de los métodos de esta clase un usuario podrá enviar o recibir mensajes personales de sus contactos. Esta clase depende de las clases “Perfil” y “Contactos”.

Herramienta colaborativa “Blog”: Las clases relacionadas con esta herramienta colaborativa y que se pueden apreciar en la Figura 3.6 distinguiéndose en el diseño por el color rojo son:

- Perfil: Tiene la misma función que en las otras herramientas y que se describió anteriormente.
- Contactos: Para el caso de esta herramienta, esta clase proporciona el acceso a la lista de contactos de cada usuario. Esta clase depende de la clase “Perfil”.
- Blog: Los métodos de esta clase permiten al usuario administrar su información personal a través de las otras clases (MBinvitado, Imagen y Mensajes Blog). Esta clase depende de las clases “Perfil” y “Contactos.”
- MBinvitado: Con los métodos de esta clase el usuario podrá dejar un mensaje en el blog de uno de sus contactos. Esta clase depende de las clases “Blog” y “Perfil”.
- Imagen: Los métodos de esta clase le dan al usuario la opción de colocar una imagen y un comentario adjunto a esta, en su blog. Esta clase depende de las clases “Blog” y “Perfil”.
- MensajesBlog: Los métodos de esta clase permiten al usuario agregar un comentario personal a su blog. Esta clase depende de la clase “Blog”.

Es importante señalar que los atributos y métodos utilizados en la representación del diagrama de clases están en el lenguaje orientado a objetos C#.NET.

Las clases “Perfil” y “Grupos”, que son presentadas en el presente diagrama de clases y que forman parte indispensable de las herramientas colaborativas, son tratadas con más

detalle en el trabajo de tesis denominado “Una Plataforma de Apoyo al Aprendizaje basada en Servicios Web”.

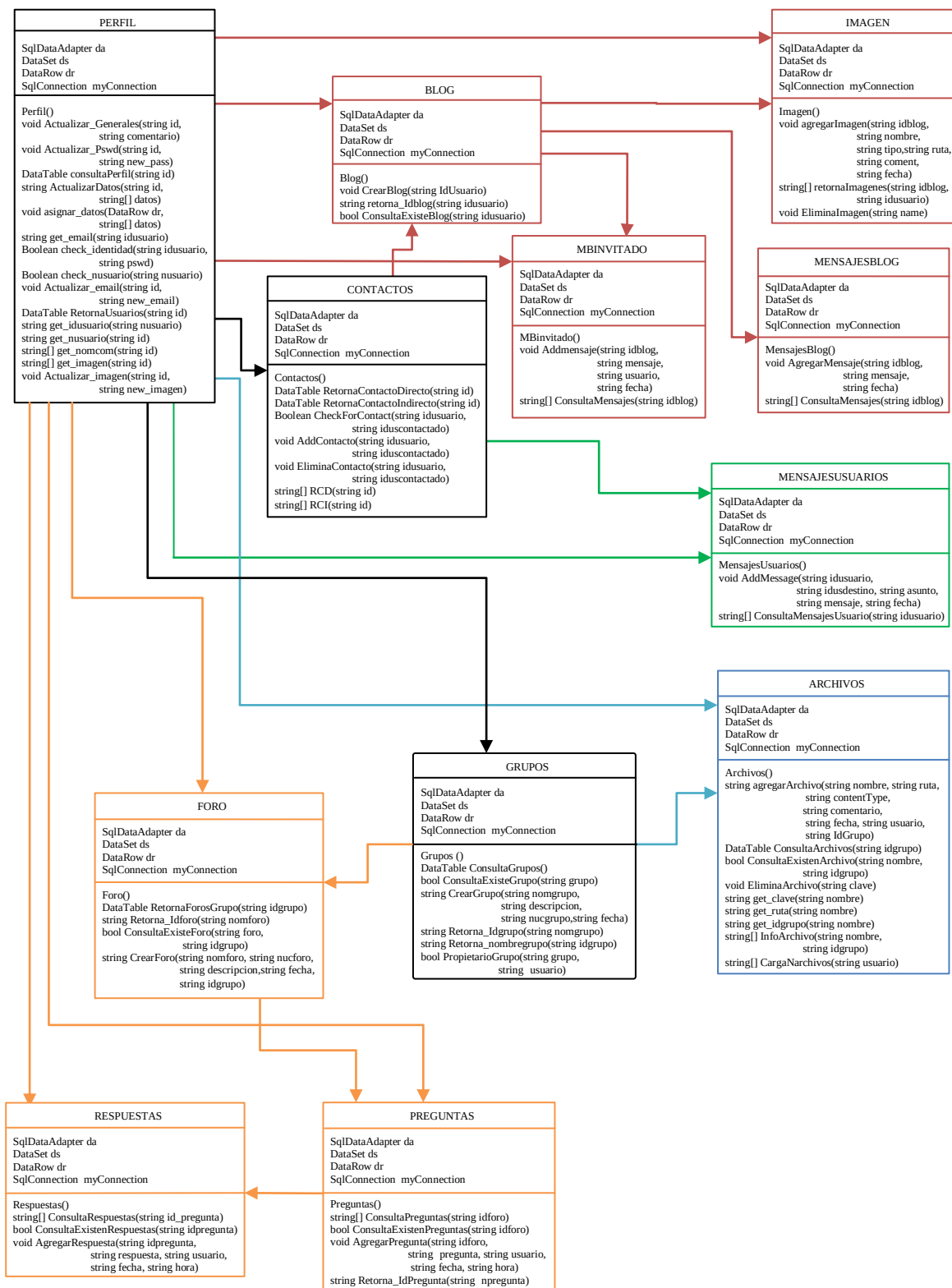


Figura 3.6 Diagrama de clases de las herramientas colaborativas de SisAAC.

3.2.2. Diagramas de Secuencia

Los casos de uso descritos durante el análisis, muestran gráficamente la interacción que un usuario tiene con el sistema, por tanto los diagramas de secuencia delinearán esos casos de uso dentro del sistema. Es decir, se verá con más claridad la secuencia de acciones que un usuario deberá realizar al interactuar con alguna de las herramientas colaborativas.

A continuación se muestran y describen los diagramas de secuencia de las cuatro herramientas colaborativas desarrolladas para el sistema de SisAAC.

El diagrama de secuencia de la herramienta colaborativa **“Blog”** (ver Figura 3.7), describe gráficamente el orden en que un usuario interactúa con esta herramienta, desde el momento en que ingresa y las distintas opciones que tiene tales como: agregar una imagen y comentario o ingresar al blog de algún contacto y dejarle un comentario en su blog.

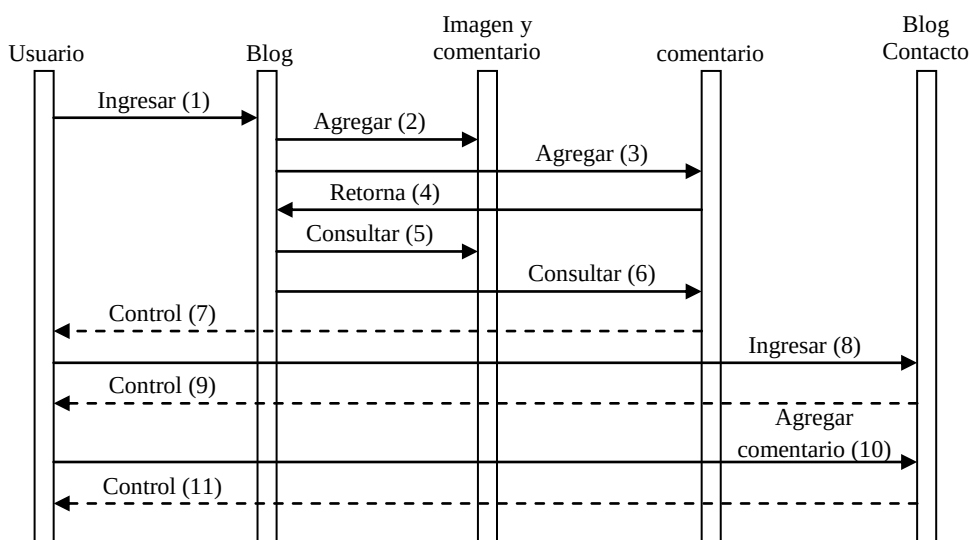


Figura 3.7 Diagrama de secuencia del blog.

El diagrama de secuencia de la herramienta colaborativa **“Foro”** (ver Figura 3.8), muestra como un usuario puede crear o ingresar un foro de discusión y a continuación la secuencia para hacer o consultar las preguntas de dicho foro y por último que hacer para dar respuesta a alguna pregunta o consultar las respuestas dadas por otros usuarios.

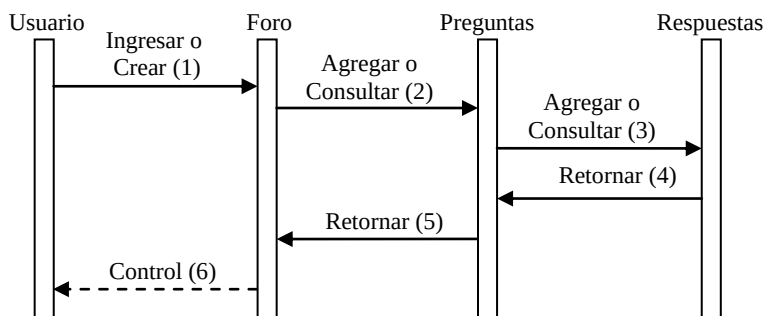


Figura 3.8 Diagrama de secuencia del foro.

El diagrama de secuencia de la herramienta colaborativa **“Buzón de Mensajes”** (ver Figura 3.9), muestra el orden de las acciones que un usuario debe seguir para enviar un mensaje a uno de sus contactos, consultar los mensajes recibidos o eliminar algún mensaje de su buzón personal.

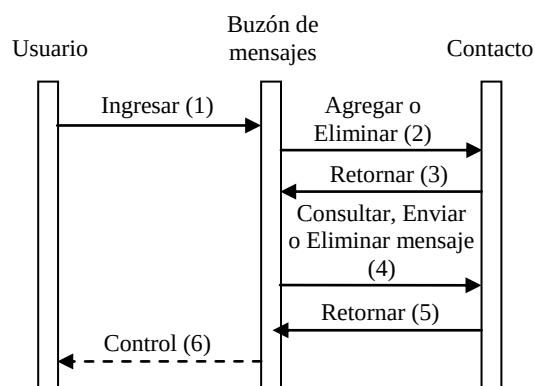


Figura 3.9 Diagrama de secuencia del buzón de mensajes.

El diagrama de secuencia de la herramienta colaborativa **“Espacio para compartir archivos”** (ver Figura 3.10), describe la forma en que un usuario puede subir, bajar o eliminar un archivo, ya que estas tres acciones pueden suceder eligiendo una a la vez.

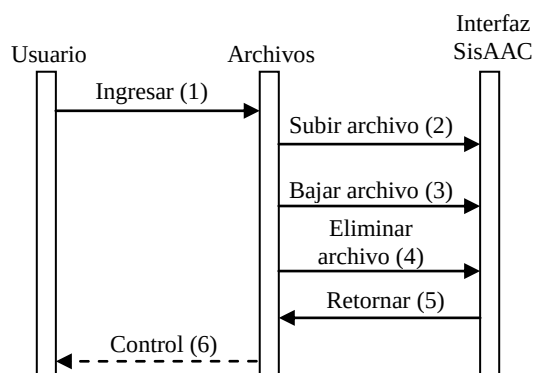


Figura 3.10 Diagrama de secuencia del espacio para compartir archivos.

3.2.3. Diagrama Entidad-Relación

Hasta este momento se ha descrito parte de lo que se necesita para poder hacer la implementación de las herramientas colaborativas, sin embargo aun se necesita dar el planteamiento de la base de datos en la cual se almacenara toda la información para que dichas herramientas puedan funcionar de manera óptima. La información necesaria para poder hacer este planteamiento la obtenemos a partir del análisis de requerimientos dados por los casos de uso de los cuales se pueden identificar las entidades necesarias. Tomando en cuenta que se trabajo con prototipos, entonces el refinamiento en la selección de los atributos correctos se hizo de forma gradual.

También es necesario hacer énfasis que como las herramientas colaborativas deberían trabajar sobre una plataforma, entonces la base de datos fue diseñada conjuntamente con el trabajo de tesis denominado “Una Plataforma de Apoyo al Aprendizaje basada en Servicios Web”, pues no se podía hacer un diseño independiente a la plataforma.

A continuación se describen las entidades y sus atributos correspondientes, agrupándolas por herramienta.

Entidades de la herramienta colaborativa “Foro”.

Entidad “Foro”: Herramienta colaborativa que permite crear foros de discusión.

- Idforo: Identificador del foro, este es único y auto incrementable.
- IdGrupo: Identificador del grupo al que pertenece el foro.
- nomforo: Nombre del foro.
- nucforo: Alias del usuario que creó el foro.
- descripción: Descripción del foro.
- fechacreacion: Fecha en la cual fue creado el foro.

Entidad “Preguntas”: Describe la información que identificara a una pregunta en un foro.

- Idpregunta: Identificador de una pregunta. Este es único y auto incrementable.
- nupregunta: Alias del usuario que hace la pregunta.
- pregunta: Texto correspondiente a la pregunta.
- fecha: Fecha en que fue hecha la pregunta.
- hora: Hora en que fue hecha la pregunta.
- Idforo: Identificador del foro al que pertenece la pregunta.

Entidad “Respuestas”: Describe la información que identificara a una respuesta con respecto a una pregunta.

- Idrespuesta: Identificador de una pregunta. Este es único y auto incrementable.
- nurespuesta: Alias del usuario que responde a una pregunta.
- Respuesta: Texto correspondiente a la respuesta.
- Fecha: Fecha en que fue respondida la pregunta.
- hora: Hora en que fue respondida la pregunta.
- Idpregunta: Identificador de la pregunta a la que corresponde la respuesta.

Entidades de la herramienta colaborativa “Blog”.

Entidad “Blog”: Herramienta colaborativa que permite la administración de la información que se describe en el resto de las entidades de esta misma herramienta.

- Idblog: Identificador del blog para cada usuario, este es único y auto incrementable.
- IdUsuario: Identificador del usuario al que está asociado el blog.

Entidad “ImagenesBlog”:

- idImagen: Identificador de la imagen que un usuario coloca en su blog personal, este es único y auto incrementable.
- MimeType: Tipo de imagen que el usuario coloca en su blog personal.
- nombre: Nombre de la imagen.
- comentario: descripción de la imagen, este comentario es obligatorio.
- fecha: Fecha en la cual la imagen es colocada en el blog.
- Idblog: Identificador del blog al cual estará asociada la imagen.

- ruta: Descripción del espacio físico en el cual será almacenada la imagen.

Entidad “MBInvitado”:

- Idmensaje: Identificador del mensaje que el usuario ha dejado en el blog de uno de sus contactos, este es único y auto incrementable.
- mensaje: Texto correspondiente al mensaje.
- fecha: Fecha en la cual fue escrito el mensaje.
- Idblog: Identificador del blog al cual está asociado el mensaje.
- nusuario: Alias del usuario que escribe el mensaje.

Entidad MensajesBlog:

- Idmensaje: Identificador del comentario que un usuario hacen su blog personal.
- mensaje: Texto correspondiente al comentario.
- fecha: fecha en la cual fue escrito el comentario.
- Idblog: Identificador del blog al cual está asociado el comentario.

Entidades de la herramienta colaborativa “Buzón de mensajes”.

Entidad MensajesUsuarios: Herramienta colaborativa que permite enviar y recibir mensajes personales entre usuarios.

- clave: Identificador del mensaje que un usuario envía al buzón de un contacto, este es único y auto incrementable.
- Idusdestino: Identificador del usuario al cual se le envía el mensaje.
- mensaje: Texto correspondiente al mensaje.
- fecha: Fecha en la cual se envía el mensaje.
- asunto: Comentario no obligatorio que da una descripción del mensaje a enviar.
- IdUsuario: Identificador del usuario que envía el mensaje.

Entidades de la herramienta colaborativa “Espacio para compartir archivos”.

Entidad UploadFiles: Herramienta colaborativa que permite a un usuario compartir archivos con toda la comunidad de usuarios registrados.

- clave: Identificador del archivo, este es único y auto incrementable.
- nombre: Nombre del archivo.
- ruta: Descripción del espacio físico en el cual será almacenada la imagen.
- MimeType: Tipo de archivo que se está compartiendo.
- comentario: descripción del contenido del archivo.
- fecha: Fecha en la que el archivo se sube al sistema.
- usuario: Alias del usuario que comparte el archivo.
- IdGrupo: Identificador del grupo al cual estará asociado el archivo.

La entidad “Contactos” (a continuación descrita), no pertenece exclusivamente a una herramienta pues tanto la herramienta “blog” como la herramienta “Buzón de mensajes” hacen referencia en algún momento a esta entidad.

Entidad Contactos: Describe a los usuarios que están relacionados entre sí, diciendo cual de los dos es el que contacto al otro inicialmente.

- Iduscontactado: Identificador del usuario solicitado como contacto.
- IdUsuario: Identificador del usuario que contacta a otro usuario.
- clave: Identificador de la relación existente entre dos usuarios, este es único y auto incrementable.

Las entidades “Perfil” y “Grupos” al igual que la entidad “Contactos” no pertenecen a alguna herramienta en especial, pues varias entidades hacen referencia a ellas, pero a diferencia de la entidad “contactos”, ambas se encuentran más claramente inmersas en el diseño correspondiente a la plataforma. Por tal razón en este trabajo de tesis no serán descritas pues se hará en el trabajo de tesis previamente mencionado al inicio de esta sección.

Dados los tipos de entidades anteriormente descritas, se procede a describir las relaciones existentes entre ellas, así como su cardinalidad agrupándolas nuevamente de acuerdo a la herramienta que correspondan.

- Relaciones entre las entidades de la herramienta colaborativa “Foro”.
 1. Relación “Tiene”: Es la relación existente entre la entidad “Grupo” y la entidad “Foro”, la cual describe que un grupo puede tener muchos foros pero un foro solo puede pertenecer a un grupo. Por lo tanto la cardinalidad existente entre estas dos relaciones es de 1 a muchos (1 : m).
 2. Relación “Administra”: Es la relación existente entre la entidad “Foro” y la entidad “Preguntas”, la cual describe que en un foro se pueden hacer muchas preguntas pero una pregunta en específico solo puede albergarse en un foro en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (1 : m).
 3. Relación “Tiene”: Es la relación existente entre la entidad “Preguntas” y la entidad “Respuestas”, la cual describe que una pregunta puede tener múltiples respuestas pero una respuesta solo puede corresponder a una pregunta en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (1 : m).
- Relaciones entre las entidades de la herramienta colaborativa “Blog”.
 1. Relación “Tiene”: Es la relación existente entre la entidad “Perfil” y la entidad “Blog”, la cual describe que un usuario puede tener un blog y que un blog específico solo puede pertenecer a un usuario. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a uno (1 : 1).
 2. Relación “Administra”: Es la relación existente entre la entidad “Blog” y la entidad “MensajesBlog”, la cual describe que en un usuario puede colocar

muchos comentarios en su blog, sin embargo un comentario solo puede corresponder a un blog en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (1 : m).

3. Relación “Administra”: Es la relación existente entre la entidad “Blog” y la entidad “MBInvitado”, la cual describe que en un blog un contacto puede dejar muchos mensajes pero sin duda cada mensaje corresponderá a un usuario en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (m : 1).
 4. Relación “Administra”: Es la relación existente entre la entidad “Blog” y la entidad “ImágenesBlog”, la cual describe que un usuario puede colocaren su blog todas las imágenes que desee pero cada imagen solo podrá pertenecer a ese blog en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (1 : m).
- Relaciones entre las entidades de la herramienta colaborativa “Buzón de mensajes”.
 1. Relación “Envía”: Es la relación existente entre la entidad “Usuarios” y la entidad “Mensajes”, la cual describe que en un usuario puede enviar muchos mensajes a un único usuario y cada mensaje que ese usuario reciba corresponderá a un usuario en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a uno (1 : 1).
 2. Relación “Recibe”: Es la relación existente entre la entidad “Usuarios” y la entidad “Mensajes”, la cual describe que un usuario puede recibir muchos mensajes de otro usuario pero solo uno a la vez. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (1 : 1).
 - Relaciones entre las entidades de la herramienta colaborativa “Espacio para compartir archivos”.
 1. Relación “Administra”: Es la relación existente entre la entidad “Grupo” y la entidad “Archivos”, la cual describe que en un usuario puede subir muchos archivos a un grupo, pero cada archivo pertenecerá a un grupo en específico. Por lo tanto la cardinalidad existente entre estas dos entidades es de uno a muchos (1 : m).
 2. La relación “Tiene” correspondiente a las entidades “Usuarios” y “Contactos” que, como se comento antes está ligada tanto a la herramienta “Blog” como a la herramienta “Buzón de mensajes”, describe que un usuario puede tener muchos contactos y este a su vez puede ser contacto de el usuario que lo contacto y de otros más. Por lo tanto la cardinalidad existente entre estas dos entidades es de muchos a muchos (m : n).

De acuerdo a las entidades y sus relaciones que se describieron obtenemos el diagrama Entidad-Relación de la Figura 4.1, enfatizando en que tal diagrama muestra el esquema de

la base de datos completa que se diseño para SisAAC, resaltando en color verde exclusivamente aquellas entidades y relaciones concernientes a las herramientas colaborativas descritas en este trabajo de tesis.

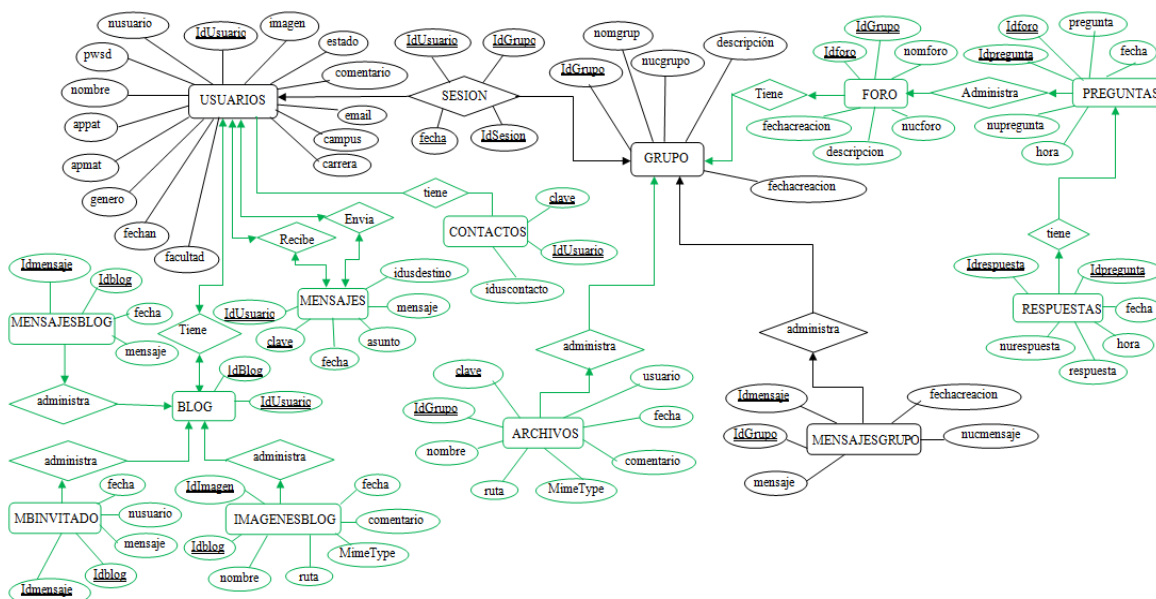


Figura 3.11. Diagrama Entidad-Relación de la base de datos SisAAC.

La conversión del diagrama Entidad-Relación a un Modelo Relacional (ver Figura 4.2), se realizo utilizando la herramienta case “Erwin Data Modeler v7.0”, la cual nos brinda la facilidad de detallar cada una de las entidades así como sus atributos y relaciones. Inherente a esto la definición de las llaves primarias y las llaves foráneas se realiza de forma casi automática lo cual nos ayuda a minimizar los riesgos en la integridad relacional.

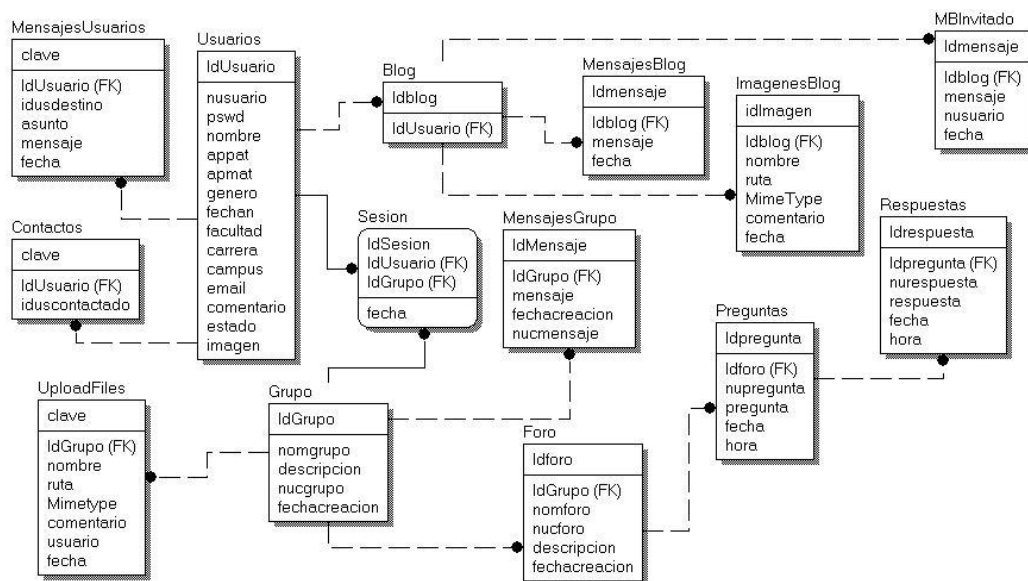


Figura 3.12. Modelo Relacional.

El proceso de normalización de una base de datos es indispensable ya que de no hacerlo, no se puede garantizar tanto su integridad como su buen funcionamiento.

Erwin Data Modeler v7.0 no garantiza que una base de datos este normalizada al realizar el diseño con sus herramientas, pero si ofrece analizar la congruencia entre las relaciones con respecto a las llaves primarias y foráneas lo cual permite al desarrollador enfocarse en el buen diseño de la base de datos.

Para que una base de datos sea confiable se recomienda que por lo menos cumpla con las tres primeras formas normales (ver sección 2.3), las cuales garantizaran que la base de datos funcionara correctamente.

La base de datos propuesta para el sistema SisAAC cumple con las tres primeras formas normales ya que:

1. Todos y cada uno sus atributos almacenan valores atómicos, es decir, no existen atributos que contengan valores múltiples.
2. No existen redundancias en ninguna de las entidades, en consecuencia las actualizaciones o eliminación de datos se realizan de una forma tal que no pone en riesgo la integridad de la base de datos.
3. Todas las entidades poseen una única clave con lo cual se garantiza que todos los atributos de cada entidad son dependientes de su clave primaria.

Algunas entidades hacen uso del atributo “nusuario” (alias de un usuario), el cual aunque parezca ser una redundancia no lo es, pues para fines de mantener la consistencia de datos en la base de datos se requiere que, aunque un usuario sea eliminado de la base de datos, esta información prevalezca.

Capítulo 4. Implementación de las herramientas colaborativas de SisAAC

Las herramientas colaborativas que han sido descritas hasta el momento, forman parte y complementan la plataforma de trabajo SisAAC que se desarrollo conjuntamente en el trabajo de tesis denominado “Una Plataforma de Apoyo al Aprendizaje basada en Servicios Web”. La cual tiene como función proporcionar un ambiente de trabajo el cual disponga de todas las herramientas necesarias e integradas en dicha plataforma para que un alumno de la FCC pueda:

- Compartir información a través de archivos ya sea subiendo o bajándolos de la plataforma SisAAC.
- Enviar y recibir mensajes usando un buzón de mensajes personal integrado en la plataforma SisAAC.
- Participar en temas afines a su carrera creando o accediendo a foros de discusión disponibles en la plataforma SisAAC.
- También podrá contar con un blog personal integrado en la plataforma SisAAC mediante el cual pueda llevar un historial de su vida académica en la FCC.

Todas estas herramientas han sido implementadas utilizando la programación orientada a objetos tomando como lenguaje a C#.NET, y puesto que se integro en un entorno de trabajo colaborativo basado en Servicios Web se utilizo a ASP.NET para desarrollar tanto la plataforma como las herramientas, haciendo uso también de DOM, Hojas de estilo (CSS) y JavaScript para darle dinamismo a la plataforma y a las herramientas.

La base datos se implemento en SQL Server 2008 utilizando las librerías proporcionadas por ADO.NET para acceder a la base de datos y las tablas generadas para la plataforma y herramientas.

Se utilizo IIS (Internet Information Server) v7.0 como servidor web ya que ASP.NET trabaja de forma nativa sobre este servidor que explota todas las capacidades y características de la programación orientada a objetos en .NET, así como también los Servicios Web.

El entorno de desarrollo de ASP.NET se encarga de los diferentes procesos que se llevan a cabo para generar y utilizar los Servicios Web lo cual permite que nos enfoquemos en la programación de los Servicios Web y nos liberemos del trabajo que implica la generación de los documentos XML ya que ASP.NET lo hace por nosotros.

A continuación se describe el proceso de generación de un servicio web en ASP.NET, en el cual se encuentra el documento WSDL, el repositorio de Servicios Web (UDDI) y el mensaje SOAP de petición y respuesta.

```
1. using System;
   using System.Web.Services;
   using System.Web;
2. [WebService(Namespace = "http://tempuri.org/")]
3. [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
4. [System.Web.Script.Services.ScriptService]:
5. public class Service : System.Web.Services.WebService{
   public Service () { }
6. [WebMethod(Description = "")]
   public string HelloWorld() { return "Hello World";}
}
```

1. Librerías que utiliza C#.NET u otro lenguaje de programación contenido en Visual Studio.
2. WebService establece el espacio de nombres XML predeterminado para el servicio Web (UDDI).
3. WebServiceBinding hace posible la interoperabilidad entre servicios, ya que de no contener o anular esta línea de código, un servicio web de ASP.NET no podría comunicarse con otro servicio desarrollado en otra plataforma.
4. Esta línea de código permite que el servicio web sea invocado por JavaScript o ASP.NET AJAX.
5. Define la clase base opcional de los Servicios Web XML, que proporciona acceso directo a los objetos comunes de ASP.NET.
6. WebMethod proporciona la capacidad a los métodos de un servicio Web de poder comunicarse a través de la Web.

A continuación se muestra el documento WSDL del servicio que se ha tomado como ejemplo el cual es generado por ASP.NET.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsc:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://tempuri.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://tempuri.org/"
xmlns:wsc="http://schemas.xmlsoap.org/wsdl/">
<wsc:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="HelloWorld">
<s:complexType />
</s:element>
<s:element name="HelloWorldResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
</s:schema>
</wsc:types>
<wsc:message name="HelloWorldSoapIn">
<wsc:part name="parameters" element="tns:HelloWorld" />
</wsc:message>
<wsc:message name="HelloWorldSoapOut">
<wsc:part name="parameters" element="tns:HelloWorldResponse" />
</wsc:message>
<wsc:portType name="ServiceSoap">
<wsc:operation name="HelloWorld">
<wsc:input message="tns:HelloWorldSoapIn" />
<wsc:output message="tns:HelloWorldSoapOut" />
</wsc:operation>
</wsc:portType>
<wsc:binding name="ServiceSoap" type="tns:ServiceSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
<wsc:operation name="HelloWorld">
<soap:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
<wsc:input>
<soap:body use="literal" />
</wsc:input>
</wsc:binding>
</wsc:definitions>
```

```

</wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
- <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="HelloWorld">
  <soap12:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
</wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
- <wsdl:service name="Service">
- <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
  <soap:address location="http://localhost/WebSite/Service.asmx" />
</wsdl:port>
- <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
  <soap12:address location="http://localhost/WebSite/Service.asmx" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

A continuación se muestra el mensaje SOAP 1.2 para la solicitud del servicio web.

POST /WebSite/Service.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: **length**

```

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
<soap12:Body>
<HelloWorld xmlns="http://tempuri.org/" />
</soap12:Body>
</soap12:Envelope>

```

A continuación se muestra el mensaje SOAP 1.2 que da como respuesta el servicio web.

HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: **length**

```

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
<soap12:Body>
<HelloWorldResponse xmlns="http://tempuri.org/">
<HelloWorldResult>string</HelloWorldResult>
</HelloWorldResponse>
</soap12:Body>
</soap12:Envelope>

```

Puesto que el ambiente de desarrollo implica que SisAAC trabaje en una arquitectura cliente-servidor los requerimientos de hardware y software son diferentes para cada caso. A continuación se dan los requerimientos mínimos recomendados para el cliente y el servidor.

- Cliente:
 1. Procesador a 2.0 Ghz.
 2. 512 Mb en RAM.
 3. 120 Gb en HD.
 4. Monitor 17"
 5. Resolución de video de 1152 por 864 pixeles.
 6. Windows XP SP1.
 7. Internet Explorer 6.0

- Servidor:
 1. Procesador Quad-Core a 1.8 Ghz.
 2. 4 Gb en RAM.
 3. 600 Gb en HD.
 4. Monitor 17"
 5. Resolución de video de 1152 por 864 pixeles.
 6. Windows Web Server 2008.
 7. Internet Explorer 8.0.
 8. IIS 7.0.
 9. Visual Studio 2008.

El Diagrama de Clases y el Modelo Relacional ambos diseñados en la sección 3.2 concerniente al diseño del sistema son retomados ahora para poder hacer la implementación de las herramientas colaborativas.

Los documentos que se generan para cada una de las herramientas colaborativas son similares a los presentados por el servicio web que ha sido tomado como ejemplo.

A continuación se describe la implementación de cada una de las herramientas colaborativas desarrolladas para SisAAC, mostrando solo el servicio web más representativo de cada herramienta así como el método principal de cada una de ellas.

4.1. Herramienta colaborativa “Foro”

El servicio web llamado “Foro.aspx”, consta inicialmente de la definición de las bibliotecas del sistema las cuales se identifican con la palabra reservada “using”, seguida de la biblioteca.

```
using System.Web.Services;  
using System.Configuration;  
using System.Data.SqlClient;  
using System.Data;  
using System;
```

Cuando un servicio web será invocado desde un script de JavaScript o por ASP.NET AJAX se debe definir un espacio de nombres identificado por la palabra reservada “namespace”, seguida del espacio de nombres.

```
namespace Samples.AspNet
```

Todo servicio web debe guardar una referencia en un directorio UDDI y para el caso de ASP.NET la forma de definirlo es a través de “WebService”.

```
[WebService(Namespace = "http://tempuri.org/")]
```

Este servicio web cumple con los requisitos de interoperabilidad pues es por medio de “WebServiceBinding” que se le proporciona esta característica.

```
[WebServiceBinding (ConformsTo = WsiProfiles.BasicProfile1_1)]
```

Este servicio web puede ser invocado por JavaScript o ASP.NET AJAX incluyendo la siguiente línea de código.

```
[System.Web.Script.Services.ScriptService]
```

A pesar de haber definido las librerías que utilizara el servicio web, se debe hacer una definición explícita del uso de las librerías más comunes al momento de declarar la clase principal del servicio web.

```
public class Foro : System.Web.Services.WebService
```

Este servicio web hace uso de variables de clase que se utilizan en la declaración de objetos utilizados para hacer la conexión a la base de datos utilizando ADO.NET.

```
private SqlDataAdapter da;  
private DataSet ds;  
private DataRow dr;  
private SqlConnection myConnection = new SqlConnection();
```

Como en todo lenguaje orientado a objetos, C#.NET necesita la definición de un constructor para inicializar datos que sean requeridos o de lo contrario de un constructor por default el cual será invocado al crear una instancia del servicio web.

```
public Foro() { }
```

El método que se muestra de este servicio web utiliza “WebMethod” para indicar que podrá ser accedido desde la web.

```
[WebMethod(Description = " Metodo que Crea un Foro")]
```

El método “CrearForo” es el más importante de este servicio web pues sirve para crear un foro, accediendo a la base de datos para generar un nuevo registro en el cual se almacenara la información correspondiente al nuevo foro.

```
public string CrearForo(string nomforo, string nucforo, string descripcion, string fecha, string idgrupo){  
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;  
    SqlConnection myConnection = new SqlConnection(strConnectionString);  
    string strCommandText = "SELECT * FROM Foro ";  
    da = new SqlDataAdapter(strCommandText, myConnection);  
    SqlCommandBuilder cb = new SqlCommandBuilder(da);  
    cb.QuotePrefix = "[";  
    cb.QuoteSuffix = "]";  
    da.UpdateCommand = cb.GetUpdateCommand();  
    da.InsertCommand = cb.GetInsertCommand();  
    da.DeleteCommand = cb.GetDeleteCommand();  
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
```

```

DataTable dt = new DataTable();
try {
    da.Fill(dt);
    dr = dt.NewRow();
    dr["nomforo"] = nomforo;
    dr["nucforo"] = nucforo;
    dr["descripcion"] = descripcion;
    dr["fechacreacion"] = fecha;
    dr["idGrupo"] = idgrupo;
    dt.Rows.Add(dr);
    da.Update(dt);
    dt.AcceptChanges();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }
finally {
    if (myConnection != null)
        myConnection.Close();
}
return "Foro creado exitosamente!!!";
}

```

El código completo del servicio web descrito y otros Servicios Web utilizados en la implementación de esta herramienta colaborativa es mostrado en el Apéndice B.

Las hojas de estilo fueron utilizadas para dar formato a la interfaz (colores, distribución de todos los elementos, fuentes, etc.).

JavaScript y DOM fueron utilizados para invocar los Servicios Web, validación de formularios y para mostrar en la interfaz algunos de los elementos de esta tales como:

- Formato de preguntas ya hechas
- Formulario para enviar una pregunta
- Etc.

La Figura 4.1 muestra el resultado obtenido del uso de los Servicios Web, hojas de estilo, JavaScript y DOM.

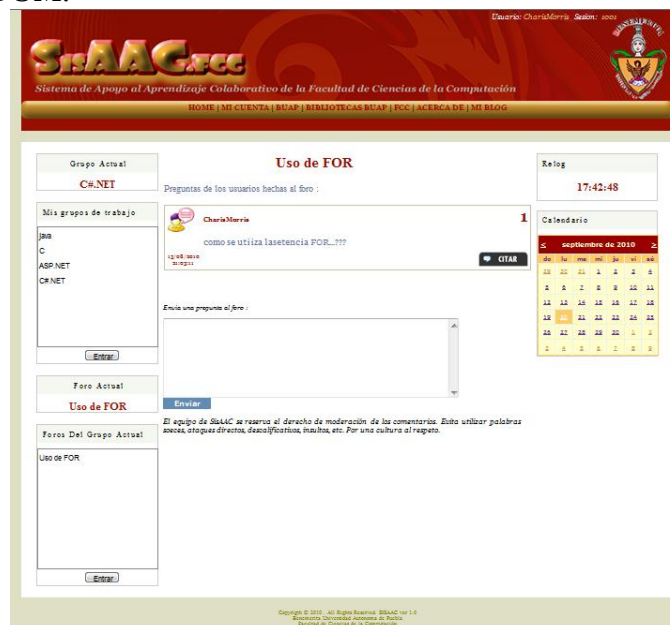


Figura 4.1 Interfaz de un foro de discusión.

4.2. Herramienta colaborativa “Buzón de mensajes”

El servicio web llamado “MensajesUsuarios.aspx” que corresponde a la herramienta colaborativa en cuestión, consta de los mismos elementos para la comunicación, publicación y descripción de los Servicios Web necesarios para su implementación, ya que como se menciono todos los Servicios Web se desarrollaron en ASP.NET.

A continuación se presenta la descripción de este servicio web y el método utilizado para enviar un mensaje a otro usuario.

Declaración de librerías.

```
using System.Web.Services;  
using System.Configuration;  
using System.Data.SqlClient;  
using System.Data;  
using System;
```

Declaración del espacio de nombres.

```
namespace Samples.AspNet
```

Declaración del directorio UDDI.

```
[WebService(Namespace = "http://tempuri.org/")]
```

Declaración de la sentencia para habilitar la interoperabilidad.

```
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
```

Declaración de la sentencia para que el servicio web pueda ser invocado por JavaScript o ASP.NET AJAX.

```
[System.Web.Script.Services.ScriptService]
```

Declaración de la clase principal del servicio web para habilitar el uso de las librerías comunes.

```
public class MensajesUsuarios : System.Web.Services.WebService
```

Declaración del constructor por default.

```
public MensajesUsuarios() { }
```

Declaración de las variables de instancia correspondientes a objetos utilizados para la conexión a la base de datos.

```
protected SqlDataAdapter da;  
protected DataSet ds;  
private DataRow dr;  
protected SqlConnection myConnection = new SqlConnection();
```

Declaración de la sentencia que permite que el servicio web sea accedido desde la web.

```
[WebMethod(Description = " Metodo que Crea un Mensaje")]
```

Método que permite enviar un mensaje a otro usuario, el cual es almacenado en la base de datos para estar disponible en línea en todo momento.

```
public void AddMessage(string idusuario, string idusdestino, string asunto, string mensaje, string fecha) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT * FROM mensajesusuarios ";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    cb.QuotePrefix = "[";
    cb.QuoteSuffix = "]";
    da.UpdateCommand = cb.GetUpdateCommand();
    da.InsertCommand = cb.GetInsertCommand();
    da.DeleteCommand = cb.GetDeleteCommand();
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    try {
        da.Fill(dt);
        dr = dt.NewRow();
        dr["idusuario"] = idusuario;
        dr["idusdestino"] = idusdestino;
        dr["asunto"] = asunto;
        dr["mensaje"] = mensaje;
        dr["fecha"] = fecha;
        dt.Rows.Add(dr);
        da.Update(dt);
        dt.AcceptChanges();
    }
    catch (Exception ex) { Console.WriteLine(ex.Message); }
    finally{
        if (myConnection != null)
            myConnection.Close();
    }
}
```

El código completo del servicio web descrito y otros Servicios Web utilizados en la implementación de esta herramienta colaborativa es mostrado en el Apéndice B.

Esta interfaz fue generada utilizando hojas de estilo para darle un formato amigable al usuario. JavaScript fue utilizado en la validación de la información requerida en los formularios, así como también en la invocación de los Servicios Web y DOM que trabaja incrustado en los script de JavaScript, a lo mismo que las hojas de estilo contribuyo en el maquetación de la interfaz.

La Figura 4.2 muestra el resultado obtenido del uso de los Servicios Web, hojas de estilo, JavaScript y DOM.

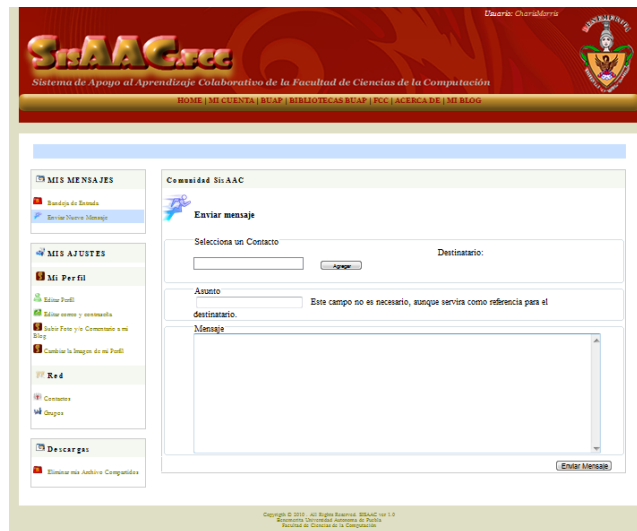


Figura 4.2 Interfaz para enviar un mensaje.

4.3. Herramienta colaborativa “Espacio para compartir archivos”

El servicio web llamado “Archivos.asmx” que será descrito a continuación, está conformado bajo la misma estructura básica con la que cuentan los servicios anteriores, con una diferencia que radica en que el método que se muestra no guarda el archivo directamente en la base de datos sino que solo guarda la ruta relativa al directorio en el sistema de archivos del servidor web en donde será almacenado físicamente. Al utilizar este método se tuvo en consideración la rapidez con la que se accede al sistema de archivos que aunque es inferior a la rapidez que se tiene al acceder a la base de datos, este presenta la ventaja de que su almacenamiento no representa el crecimiento excesivo que se podría tener al almacenar los archivos en la base de datos.

A continuación se presenta la descripción de este servicio web y el método utilizado para guardar las referencias del archivo en la base de datos.

Declaración de librerías.

```
using System.Web.Services;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System;
```

La declaración del espacio de nombres que en particular para este servicio no es indispensable, se agrega pues si en algún momento, algún cliente desea consumir el servicio web, pueda utilizar el lenguaje que prefiera.

```
namespace Samples.AspNet
```

Declaración del directorio UDDI.

```
[WebService(Namespace = "http://tempuri.org/")]
```

Declaración de la sentencia para habilitar la interoperabilidad.

```
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
```

Declaración de la sentencia para que el servicio web pueda ser invocado por JavaScript o ASP.NET AJAX. Al igual que la declaración del espacio de nombres, tampoco es necesario agregar esta línea de código pues se accederá por medio de ASP.NET a los métodos de este servicio web.

```
[System.Web.Script.Services.ScriptService]
```

Declaración de la clase principal del servicio web para habilitar el uso de las librerías comunes.

```
public class MensajesUsuarios : System.Web.Services.WebService
```

Declaración del constructor por default.

```
public Archivos(){ }
```

Declaración de las variables de instancia correspondientes a objetos utilizados para la conexión a la base de datos.

```
protected SqlDataAdapter da;  
protected DataSet ds;  
private DataRow dr;  
protected SqlConnection myConnection = new SqlConnection();
```

Declaración de la sentencia que permite que el servicio web sea accedido desde la web.

```
[WebMethod(Description = " Metodo que Guarda la referencia de un archivo en la Base de Datos")]
```

Método que permite guardar la información concerniente a un archivo en la base de datos, archivo que será almacenado en un directorio del sistema de archivos del servidor web.

```
public string agregarArchivo(string nombre, string ruta, string contentType, string comentario, string fecha, string  
                             usuario, string IdGrupo) {  
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;  
    SqlConnection myConnection = new SqlConnection(strConnectionString);  
    string strCommandText = "SELECT * FROM UploadFiles";  
    da = new SqlDataAdapter(strCommandText, myConnection);  
    SqlCommandBuilder cb = new SqlCommandBuilder(da);  
    DataSet ds = new DataSet();  
    cb.QuotePrefix = "[";  
    cb.QuoteSuffix = "]";  
    da.UpdateCommand = cb.GetUpdateCommand();  
    da.InsertCommand = cb.GetInsertCommand();  
    da.DeleteCommand = cb.GetDeleteCommand();  
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;  
    DataTable dt = new DataTable();  
    try {  
        da.Fill(dt);  
        dr = dt.NewRow();  
        dr["nombre"] = nombre;  
        dr["ruta"] = ruta;  
        dr["Mimetype"] = contentType;  
        dr["comentario"] = comentario;
```

```

dr["fecha"] = fecha;
dr["usuario"] = usuario;
dr["IdGrupo"] = IdGrupo;
dt.Rows.Add(dr);
da.Update(dt);
dt.AcceptChanges();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }
finally{
    if (myConnection != null)
        myConnection.Close();
}
return "archivo cargado exitosamente ";
}

```

El siguiente fragmento de código es el que se utiliza para guardar un archivo en el directorio del sistema de archivos del servidor web.

```

string fn = System.IO.Path.GetFileName(flarchivo.PostedFile.FileName);
string SaveLocation = Server.MapPath(@"~\profile\UploadFiles\" + Session["id_grupo"]) + "\\" + fn;
string contentType = flarchivo.PostedFile.ContentType;
flarchivo.PostedFile.SaveAs(SaveLocation);

```

El código completo del servicio web descrito y otros Servicios Web utilizados en la implementación de esta herramienta colaborativa es mostrado en el Apéndice B.

A diferencia de las otras interfaces, esta interfaz (ver Figura 4.3), no utiliza DOM ni JavaScript para la maquetación y la invocación del servicio web. Solo se basa en el uso de ASP.NET para la invocación de los Servicios Web y CSS para dar el formato que presenta.



Figura 4.3 Interfaz para subir o bajar un archivo.

4.4. Herramienta colaborativa “Blog”

Por último se describirá el servicio web llamado “Blog.asmx” y su método para crear un registro en la base de datos el cual contiene solo dos atributos: el identificador del blog que es un dato que se genera automáticamente y el identificador del usuario al cual pertenece el blog. Por lo tanto solo se ingresa explícitamente un solo dato a la base de datos, el cual es el identificador del usuario. La estructura básica del servicio web es la misma que la de los Servicios Web descritos anteriormente.

Declaración de librerías.

```
using System.Web.Services;  
using System.Configuration;  
using System.Data.SqlClient;  
using System.Data;  
using System;
```

La declaración del espacio de nombres que en particular para este servicio es indispensable pues toda la maquetación y la invocación del servicio web se realiza con DOM y JavaScript respectivamente.

```
namespace Samples.AspNet  
Declaración del directorio UDDI.
```

```
[WebService(Namespace = "http://tempuri.org/")]
```

Declaración de la sentencia para habilitar la interoperabilidad.

```
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
```

Declaración de la sentencia para que el servicio web pueda ser invocado por JavaScript o ASP.NET AJAX.

```
[System.Web.Script.Services.ScriptService]
```

Declaración de la clase principal del servicio web para habilitar el uso de las librerías comunes.

```
public class Blog : System.Web.Services.WebService
```

Declaración del constructor por default.

```
public Blog(){ }
```

Declaración de las variables de instancia correspondientes a objetos utilizados para la conexión a la base de datos.

```
protected SqlDataAdapter da;  
protected DataSet ds;  
private DataRow dr;  
protected SqlConnection myConnection = new SqlConnection();
```

Declaración de la sentencia que permite que el servicio web sea accedido desde la web.

```
[WebMethod(Description = " Metodo que Crea el Blog de un Usuario")]
```

Método que se invoca para crear un blog personal.

```
public void CrearBlog(string IdUsuario) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT * FROM Blog";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    cb.QuotePrefix = "[";
    cb.QuoteSuffix = "]";
    da.UpdateCommand = cb.GetUpdateCommand();
    da.InsertCommand = cb.GetInsertCommand();
    da.DeleteCommand = cb.GetDeleteCommand();
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    try {
        da.Fill(dt);
        dr = dt.NewRow();
        dr["IdUsuario"] = IdUsuario;
        dt.Rows.Add(dr);
        da.Update(dt);
        dt.AcceptChanges();
    }
    catch (Exception ex) { Console.WriteLine(ex.Message); }
    finally {
        if (myConnection != null)
            myConnection.Close();
    }
}
```

El código completo del servicio web descrito y otros Servicios Web utilizados en la implementación de esta herramienta colaborativa es mostrado en el Apéndice B.

La interfaz que se obtiene como resultado de la utilización de Servicios Web, el uso de JavaScript y las hojas de estilo se puede apreciar en la Figura 4.4.



Figura 4.4 Interfaz del Blog personal.

Capítulo 5. Conclusiones

La implementación de las herramientas colaborativas asíncronas centradas en el aprendizaje en grupo reducirá el tiempo y las dificultades que puede tener un alumno al tratar de estudiar un tema relacionado con la currícula. Además, estas herramientas servirán de medio de comunicación entre alumnos que no sólo reforzarán el conocimiento entre grupos de alumnos, sino también las relaciones sociales entre estos. De tal manera, que los alumnos adquieren conocimiento y aprenden a manejarse en diversos grupos de trabajo.

Las aportaciones del presente trabajo de tesis son:

- Desarrollo de cuatro herramientas colaborativas asíncronas que contribuyen a reforzar el aprendizaje colaborativo.
- La integración de estas herramientas en una plataforma de apoyo al aprendizaje colaborativo.
- Creación de relaciones académicas entre alumnos de la FCC, que con el tiempo se puedan convertir en relaciones sociales lo cual ayudará a su formación integral.
- Fomento de la reutilización y adaptación de Servicios Web en la comunidad académica de la FCC.

Apéndice A Manual de Usuario

El siguiente manual de usuario tiene como objetivo mostrar el uso adecuado de cada una de las herramientas colaborativas implementadas para la plataforma SisAAC, tomando en cuenta que un usuario puede ser tanto alumno como maestro.

Las pantallas mostradas en este manual se refieren exclusivamente al uso de las herramientas, dado que la forma de cómo interactuar con la plataforma se explica en el trabajo de tesis “Una Plataforma de Apoyo al Aprendizaje basada en Servicios Web”.

Herramienta colaborativa “Foro”

Estando dentro de SisAAC y después de haberse unido o creado algún grupo de trabajo se ingresa a una pantalla que muestra una lista de los foros de discusión existentes que pertenecen a ese grupo, pero también esta pantalla muestra formulario en donde el usuario tiene la posibilidad de crear un nuevo foro de discusión. Una vez ingresando a algún foro, el usuario puede hacer una pregunta o responder a alguna ya existente. En esta misma pantalla se encuentra un hipervínculo al espacio para compartir archivos, por tanto la Figura A.1 muestra la pantalla con todos estos elementos algunos de los cuales serán descritos más adelante en este mismo manual.



Figura A.1 Pantalla que muestra el formulario para crear un nuevo foro, un ListBox de los foros existentes y el hipervínculo del espacio para compartir archivos.

A continuación se muestra el procedimiento para llevar a cabo las acciones antes mencionadas referentes a los foros de discusión.

Creación de un nuevo foro de discusión:

1. La figura A.2 muestra el formulario que permite la creación de un nuevo foro en donde el usuario debe llenar los campos correspondientes al nombre del nuevo foro y su descripción.

Figura A.2 formulario para crear un nuevo foro.

2. Cuando se ha llenado el formulario correctamente, el usuario debe dar clic en el botón “Crear Foro” para crear el nuevo foro. De forma automática el sistema lo redirigirá hacia el foro de discusión recientemente creado (ver Figura A.3) que mostrara elementos tales como: el nombre del grupo actual, el foro en el que se encuentra, los grupos de trabajo a los que pertenece, los foros de discusión que pertenecen al grupo actual, las preguntas (si existen) hechas en el foro y un cuadro de texto para hacer una pregunta en el foro, estos dos últimos elementos serán tratados más adelante.

Figura A.3. Elementos de la pantalla de un foro creado por un usuario.

3. Para cambiar de grupo de trabajo o foro de discusión basta con seleccionar el grupo o foro y dar clic en el botón “Entrar” correspondiente a cada opción.

Envío de preguntas y respuestas al foro:

1. Para enviar una pregunta a un foro, se debe dar clic en el cuadro de texto que muestra la Figura A.3, a continuación se agrega el texto correspondiente y se da clic en el botón “Enviar”, el sistema enviara un mensaje de confirmación (ver Figura A.4), e inmediatamente después se mostrara la pregunta en la misma pantalla mostrando una referencia de el usuario que envió la pregunta (ver Figura A.5) y asociado a esta, se encontrara un botón “Citar” que al dar clic en él se podrán consultar sus respuestas o incluso dar una respuesta nueva (ver Figura A.5).

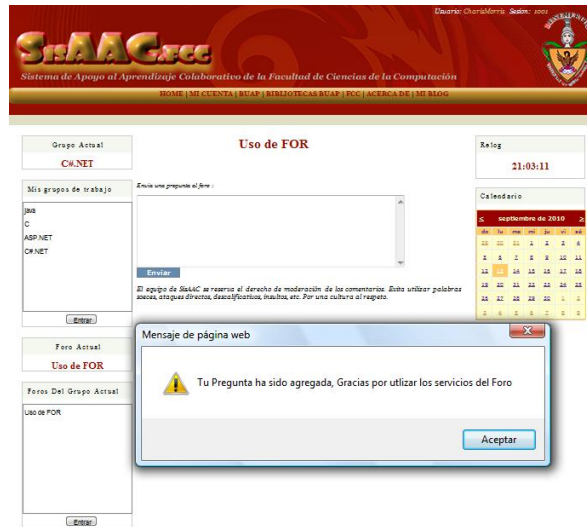


Figura A.4. Pantalla que muestra el mensaje de confirmación de una pregunta enviada al foro.



Figura A.5. Formato de una pregunta enviada a un foro.

2. Para consultar o dar respuesta a una pregunta como ya se mencionó se debe dar clic en el botón citar e inmediatamente aparecerá una nueva pantalla que contendrá los mismos elementos que la anterior, además mostrara (si existen) las respuestas dadas por otros usuarios a esa pregunta, o si se desea se puede dar una respuesta escribiendo en el cuadro de texto predefinido para ello y después dar clic en el botón “Enviar” (ver Figura A.6). Antes y después de las respuestas se encuentra un botón en color naranja con la leyenda “Retorna” (ver Figura A.7), el cual tiene la función de regresar al usuario a la pantalla de las preguntas.

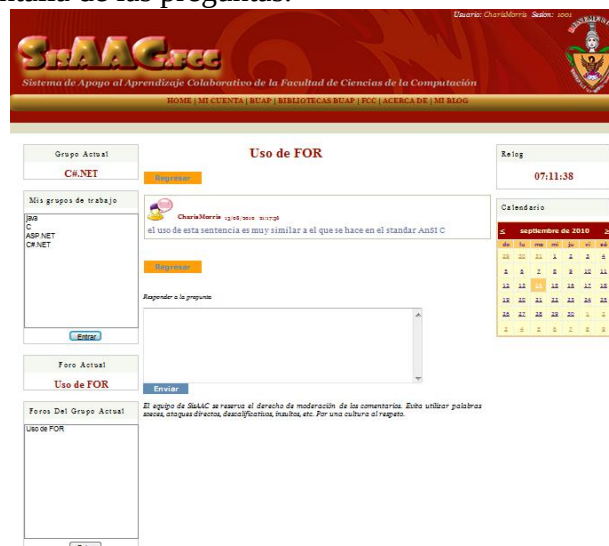


Figura A.6. Pantalla que muestra el cuadro de texto para enviar un mensaje y un mensaje enviado.

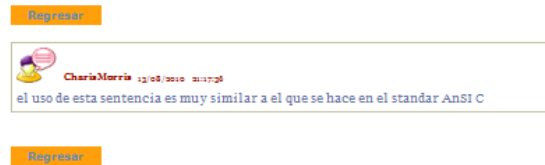


Figura A.7 Formato de una respuesta y los botones de “Regresar”.

Herramienta colaborativa “Espacio para Compartir Archivos”

Como se mencionó anteriormente, el hipervínculo hacia el espacio para compartir archivos (ver Figura A.1), se encuentra en la misma pantalla de acceso y creación de foros este tiene la leyenda “Subir y bajar archivos”. Al dar clic en este hipervínculo se mostrara una pantalla que permitirá al usuario subir o bajar un archivo del grupo en el que se encuentre. La Figura A.8 muestra los elementos con los que cuenta esta pantalla.



Figura A.8 Pantalla del espacio para compartir archivos.

A continuación se describe la funcionalidad de cada elemento de esta pantalla.

Subir un archivo al grupo de trabajo:

1. Para subir un archivo al grupo de trabajo se utiliza el formulario que tiene como leyenda “sube tu archivo” (ver Figura A.9) en el cual aparece un botón “Examinar” el cual permite al usuario hacer una búsqueda en cualquier dispositivo instalado en su computadora para seleccionar el archivo a compartir (ver Figura A.10), tomando en cuenta que debe ser de un formato y tamaño permitidos.



Figura A.9 Formulario para subir un archivo.

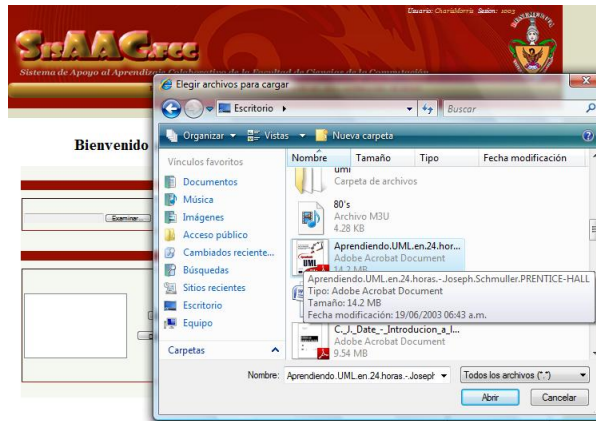


Figura A.10 Búsqueda de un archivo en la computadora.

- Una vez seleccionado dicho archivo se da clic en el botón “subir” y si el archivo fue aceptado inmediatamente aparecerá un mensaje de éxito y el archivo podrá ser visto en el “Área de descargas” a si como lo muestra la Figura A.11.



Figura A.11 Mensaje de éxito y archivo mostrado en el área de descargas.

Bajar un archivo del grupo de trabajo:

- Para bajar un archivo se dispone dentro de la misma pantalla de un formulario “área de descargas” (ver Figura A.12), el cual muestra una lista de todos los archivos disponibles en el grupo, un botón de “Detalles” que al dar clic en el proporciona una descripción del archivo seleccionado y un botón “Descargar” que como se indica baja el archivo seleccionado a la maquina del usuario.



Figura A.12 Formulario “Área de descargas” que muestra un archivo seleccionado y sus detalles.

2. Cuando se ha seleccionado el archivo a bajar y se da clic en el botón “Descargar” aparecerá un cuadro de dialogo de confirmación (ver Figura A.13) y después aparecerá una ventana en la cual se podrá elegir la una ruta en donde el archivo será guardado así como lo muestra la Figura A.14.

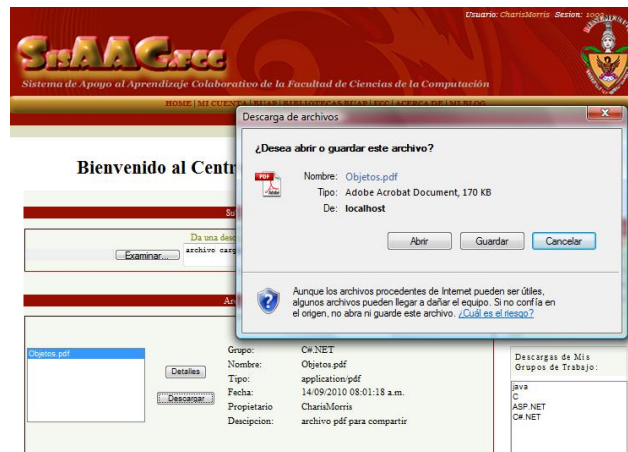


Figura A.13 Mensaje de confirmación para bajar un archivo.

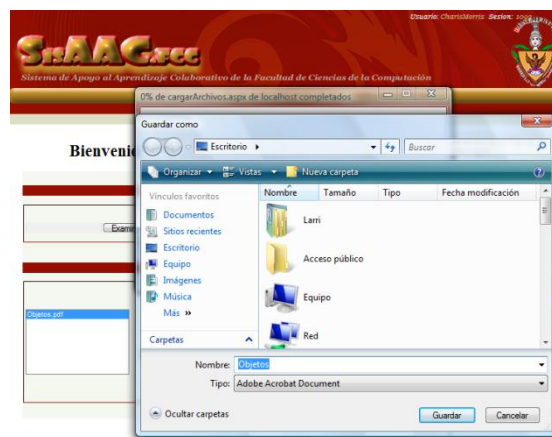


Figura A.14 Ventana para dar una ruta donde se guarde el archivo.

Eliminar un archivo:

1. Dar clic en el vínculo “Mi cuenta” (ver Figura A.15), que se localiza en la barra de navegación, a la cual se tiene acceso en cualquier momento.



Figura A.15. Barra de navegación.

2. Dar clic en el vínculo llamado “Eliminar mis archivos compartidos” (ver Figura A.16), que abre una ventana en la cual el usuario podrá elegir un archivo para eliminarlo.



Figura A.16. Vinculo para ingresar a la interfaz de eliminación de archivos.

3. La figura A.17 muestra un cuadro de texto en el cual se muestra una lista de los archivos que el usuario ha compartido. También muestra un botón “Ver Detalles”, el cual al presionarlo muestra la información concerniente a un archivo previamente seleccionado. También tiene un botón “Eliminar Archivo”, el cual al presionarlo, elimina el archivo seleccionado permanentemente sistema.

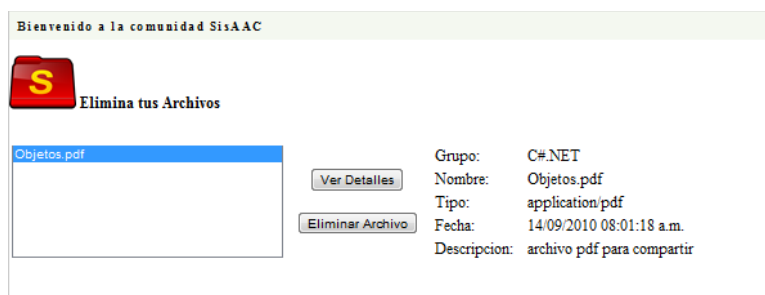


Figura A.17. Interfaz para eliminar un archivo.

Herramienta colaborativa “Buzón de mensajes”

Para que un usuario tenga acceso a su buzón de mensajes es necesario que ingrese al panel de control, dando clic en “Mi cuenta” (ver Figura A.15), lo cual fue descrito anteriormente en este mismo manual.

Una vez que el usuario se encuentra en el panel de control, hallara un recuadro con el título de “Mis mensajes” (ver figura A.18), el cual le proporciona dos opciones: consultar sus mensajes y enviar mensajes.



Figura A.18. Vínculos del recuadro “Mis mensajes”.

Consultar mensajes:

1. Al dar clic en el vínculo “bandeja de entrada” de la figura A.17, se abre una nueva ventana en la cual, si no existen mensajes dirigidos al usuario, el sistema mostrara una notificación (ver Figura A.18). En otro caso se mostrara una lista de los mensajes existentes (ver Figura A.20), dando los detalles de este tales como: Remitente, Asunto y Fecha. Cada mensaje tendrá anexado un botón “Eliminar” que al presionarlo Eliminara el mensaje de forma inmediata.

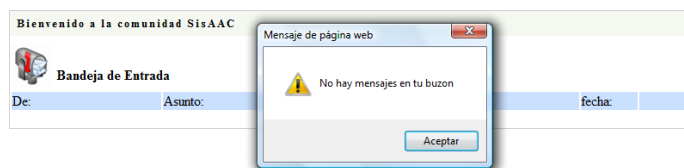


Figura A.19. Notificación del sistema cuando no existen mensajes.

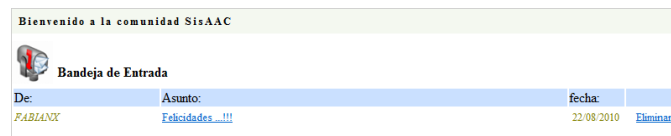


Figura A.20. Lista de los mensajes existentes en la bandeja de entrada.

2. Para ver el contenido de un mensaje se debe dar clic en el vínculo que hace referencia al asunto que de manera inmediata abrirá una nueva ventana mostrando el contenido de este (ver Figura A.21).

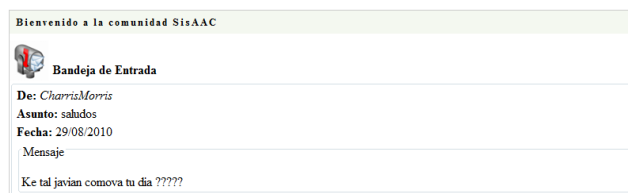


Figura A.21. Contenido de un mensaje.

Enviar mensajes:

En la figura A.17 se muestra el vínculo llamando “Enviar nuevo mensaje” por medio del cual se ingresa a la interfaz que le permitirá a un usuario enviar mensajes a sus contactos.

La Figura A.22 muestra dicha interfaz con todos sus elementos los cuales serán descritos a continuación.

Figura A.22. Interfaz para enviar un mensaje.

1. Formulario “Selecciona un contacto”. Este formulario consta de los siguientes elementos:
 - Lista con los contactos del usuario: Es una lista desplegable la cual contiene todos los nombres de los contactos de un usuario.
 - Botón “Agregar”: Después de seleccionar un contacto de la lista de contactos, al presionar este botón el nombre del contacto es cargado a un cuadro de texto llamado “Destinatario”.
 - Cuadro de texto “Destinatario”: Almacena el nombre del contacto al cual le será enviado el mensaje.
2. Asunto: Es un cuadro de texto en el cual se escribe el motivo por el cual se envía ese mensaje.
3. Mensaje: Es un cuadro de Texto en el cual el usuario puede escribir el contenido del mensaje que enviara.
4. Botón “Enviar mensaje”: Como su nombre lo indica, al presionar este botón el mensaje será enviado.

La figura A.23 muestra la forma correcta en que debe ser llenado el formato para enviar un mensaje.

Figura A.23. Formato debidamente llenado.

Herramienta colaborativa “Blog”

Un usuario puede acceder a su blog personal inmediatamente después de haber ingresado a SisAAC pues el vínculo que lo relaciona se encuentra en la barra de navegación. Este vínculo se puede apreciar en la Figura A.15.

La interfaz del blog que se muestra en la figura A.24, se divide en cuatro partes principales, las cuales serán descritas a continuación.



Figura A.24. Interfaz del blog de un usuario.

- Encabezado. Es el área en donde se localiza el logo de SisAAC y la barra de navegación (ver Figura A.25).



Figura A.25. Encabezado.

- Barra lateral izquierda. En esta área se encuentra la imagen que el usuario eligió para su perfil, información general del usuario, una lista de todos sus contactos y los mensajes que sus contactos dejan en el blog. La Figura A.26 muestra esta barra.



Figura A.26. Barra lateral izquierda.

- Centro. En esta área se encuentra el nombre del usuario, su mensaje personal, un formulario par que sus contractos dejen un mensaje en su blog y una lista en orden

cronológico de todas las imágenes que el usuario a colocado en su blog. La Figura A.27 muestra esta barra.



Figura A.27. Barra central.

- Barra lateral derecha. Esta barra contiene un solo elemento, el cual es una lista de todos los mensajes que el usuario ha dejado a sus contactos en su blog. La Figura A.28 muestra esta barra.

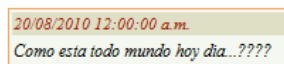


Figura A.28. Barra lateral derecha.

Hasta aquí se ha descrito la distribución de los elementos en la interfaz, y a continuación se describe como un usuario debe hacer para que su blog contenga dicha información.

En el panel de control se encuentra un vínculo llamado “Subir foto y/o comentario a mi blog” (ver Figura A.29), el cual abre una ventana que le permite al usuario colocar una foto o un comentario en su blog.



Figura A.29. Vínculo para agregar una foto o un comentario al blog.

La Figura A.30 muestra los elementos que contiene de esta interfaz y proporciona al usuario las opciones ya mencionadas.

Figura A.30. Interfaz para agregar una foto o un comentario al blog.

Como se muestra en la figura A.30, esta interfaz contiene dos formularios uno de los cuales llamado “Subir foto y comentario a mi blog” permite al usuario colocar tanto una imagen como un comentario acorde a esta. Dicho formulario contiene los siguientes elementos:

- Un botón “Examinar”. Al presionar este botón, aparece una ventana que permite al usuario hacer una búsqueda en cualquier dispositivo conectado a su computadora que contenga imágenes con formato gif, jpg o png.
- Un cuadro de texto en el cual el usuario debe colocar necesariamente un comentario acorde a la imagen seleccionada.
- Un botón “Subir foto y comentario” que al presionarlo almacena automáticamente toda la información de este formulario en la base de datos.

El otro formulario llamado “Subir mensaje a mi blog” permite al usuario colocar un mensaje en su blog teniendo los siguientes elementos:

- Un cuadro de texto donde el usuario debe escribir el mensaje.
- Un botón “Subir comentario” que agrega la información de este formulario a la base de datos.

En el panel de control también se encuentra un vínculo llamado “Contactos” (ver Figura A.31), el cual abre una ventana en donde el usuario tiene la opción de agregar o eliminar un contacto a su lista de contactos.

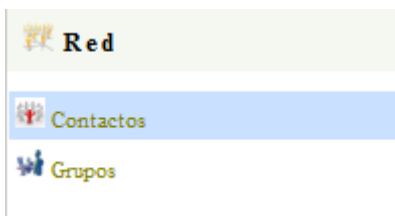


Figura A.31. Vínculo “Contactos”.

La Figura A.32 muestra los elementos que contiene esta interfaz y que a continuación serán descritos.



Figura A.32. Interfaz para agregar o eliminar un contacto.

- Una lista llamada “Usuarios Registrados” la cual muestra a todos los usuarios registrados en SisAAC.
- Un botón “Agrega” que al presionarlo después de haber seleccionado un contacto, este aparece inmediatamente en la lista llamada “Mis Contactos”.
- Una lista llamada “Mis Contactos” que contiene a todos los contactos del usuario.
- Un botón “Elimina” el cual al presionarlo después de haber seleccionado un usuario de la lista de contactos, elimina a este de esa lista automáticamente.

Apéndice B Manual Técnico

Para el desarrollo de las herramientas colaborativas se utilizaron: Microsoft Framework .NET, JavaScript, CSS, y DOM. El Framework .NET al ser un entorno completo de desarrollo para el diseño de aplicaciones web, proporciona todas las herramientas necesarias para construir una aplicación web desde cero, sin embargo existen ciertas herramientas que no son exclusivas de Microsoft y que tienen un rol muy importante en cuanto al diseño de la interfaz y la forma en que un usuario interactúa con la aplicación. Tal es el caso de JavaScript, CSS, y DOM que trabajan junto con el Framework .NET para darle una apariencia amigable y dinámica a las aplicaciones web, debido a esto fueron utilizadas en la implementación de las herramientas colaborativas propuestas en esta tesis, en tanto que el desarrollo de los Servicios Web fue hecho completamente con ASP.NET utilizando el lenguaje C#.NET para la programación de estos.

El código que se generó a partir de los Servicios Web, y las GUI (interfaces gráficas de usuario), y que se presenta en este manual, está agrupado en cada una de las secciones referentes a cada herramienta colaborativa implementada.

La base de datos fue construida con SQL Server 2008 utilizando el código generado automáticamente por “Erwin Data Modeler v7.0”, esto a partir del Modelo Relacional construido bajo este mismo entorno. Dicho código se mostrará más adelante en este mismo manual.

Herramienta colaborativa “Foro”

Servicio Web “Foro.asmx”. Este Servicio Web tiene la finalidad de proporcionar los métodos: RetornaForosGrupo, Retorna_Idforo, ConsultaExisteForo y CrearForo, necesarios para la creación de un nuevo foro en la base de datos y obtención de información básica del al respecto del foro.

```
using System.Web.Services;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System;

namespace Samples.Aspnet {
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.Web.Script.Services.ScriptService]

    public class Foro : System.Web.Services.WebService {
        private SqlDataAdapter da;
        private DataSet ds;
        private DataRow dr;
        private SqlConnection myConnection = new SqlConnection();

        public Foro() { }

        [WebMethod(Description = " Metodo que retorna un datatable con todos los Foros pertenecientes a un grupo")]
        public DataTable RetornaForosGrupo(string idgrupo) {
            string strConnectionString =
                ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
            SqlConnection myConnection = new SqlConnection(strConnectionString);
            string strCommandText = "SELECT idforo,nomforo FROM Foro where idgrupo='"+idgrupo+"'";
```

```

da = new SqlDataAdapter(strCommandText, myConnection);
SqlCommandBuilder cb = new SqlCommandBuilder(da);
da.UpdateCommand = cb.GetUpdateCommand();
da.InsertCommand = cb.GetInsertCommand();
da.DeleteCommand = cb.GetDeleteCommand();
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
DataTable dt = new DataTable();
try{ da.Fill(dt); }
catch (Exception ex) {
    Console.WriteLine(ex.Message);
    return null;
}
finally{
    if (myConnection != null)
        myConnection.Close();
}
return dt;
}

[WebMethod(Description = " Metodo que retorna el Id de un foro a partir del nombre del foro")]
public string Retorna_Idforo(string nomforo) {
    string strConnectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
SqlConnection myConnection = new SqlConnection(strConnectionString);
string strCommandText = "SELECT idforo FROM foro where nomforo='" + nomforo + "'";
da = new SqlDataAdapter(strCommandText, myConnection);
SqlCommandBuilder cb = new SqlCommandBuilder(da);
da.UpdateCommand = cb.GetUpdateCommand();
da.InsertCommand = cb.GetInsertCommand();
da.DeleteCommand = cb.GetDeleteCommand();
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
DataTable dt = new DataTable();
string id;
try{ da.Fill(dt); }
catch (Exception ex) {
    Console.WriteLine(ex.Message);
    return null;
}
finally{
    if (myConnection != null)
        myConnection.Close();
}
if (dt.Rows.Count > 0)
    id = dt.Rows[0][0].ToString();
else
    id = "false";
return id;
}

[WebMethod(Description = " Metodo que checa si existe un Foro Determinado")]
public bool ConsultaExisteForo(string foro, string idgrupo) {
    string strConnectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
SqlConnection myConnection = new SqlConnection(strConnectionString);
string strCommandText = "SELECT * FROM Foro WHERE nomforo='" + foro + "'and idgrupo='" + idgrupo + "'";
da = new SqlDataAdapter(strCommandText, myConnection);
ds = new DataSet();
try{ da.Fill(ds, "Foro"); }
catch (Exception ex) { Console.WriteLine(ex.Message); }
finally{
    if (myConnection != null) myConnection.Close();
}
}

```

```

        if (ds.Tables["Foro"].Rows.Count > 0)
            return true;
        else
            return false;
    }

    [WebMethod(Description = " Metodo que Crea un Foro")]
    public string CrearForo(string nomforo, string nucforo, string descripcion, string fecha, string idgrupo) {
        string strConnectionString =
        ConfigurationManager.ConnectionStrings["SqlConnection"].ConnectionString;
        SqlConnection myConnection = new SqlConnection(strConnectionString);
        string strCommandText = "SELECT * FROM Foro ";
        da = new SqlDataAdapter(strCommandText, myConnection);
        SqlCommandBuilder cb = new SqlCommandBuilder(da);
        da.UpdateCommand = cb.GetUpdateCommand();
        da.InsertCommand = cb.GetInsertCommand();
        da.DeleteCommand = cb.GetDeleteCommand();
        da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
        DataTable dt = new DataTable();
        try{
            da.Fill(dt);
            dr = dt.NewRow();
            dr["nomforo"] = nomforo;
            dr["nucforo"] = nucforo;
            dr["descripcion"] = descripcion;
            dr["fechacreacion"] = fecha;
            dr["idGrupo"] = idgrupo;
            dt.Rows.Add(dr);
            da.Update(dt);
            dt.AcceptChanges();
        }
        catch (Exception ex) { Console.WriteLine(ex.Message); }
        finally{
            if (myConnection != null)
                myConnection.Close();
        }
        return "Foro creado exitosamente!!!";
    }
}
}

```

El archivo “Preguntas.js” es un script de JavaScript por medio del cual se hace la invocación a los métodos del Servicio Web “Preguntas.asmx” y los valores que retorna este son manipulados por DOM para ser presentados posteriormente en la GUI.

```

var idglobal = "";
var estado_boton = true;

function mostrar_preguntas() {
    var id_foro = document.getElementById('lbl_idforo').innerHTML;
    Samples.Aspnet.Preguntas.ConsultaPreguntas(id_foro, success_preguntas);
}

function success_preguntas(resultado) {
    document.getElementById("Preguntas").innerHTML = "";
    document.getElementById("Preguntas").innerHTML += "Preguntas de los usuarios hechas al foro :<br><br>";
    for (var i = 0; i < resultado.length / 5; i++) {
        var cont = i + 1;
        var citar = "<img src=images/citar.png style=width:82px;height:25px onclick='" + responder_pregunta
        ("'+resultado[i*5+4]+'>";
        var sScript = "<SCRIPT DEFER>";
    }
}

```

```

sScript = sScript + "function responder_pregunta(id_pregunta)";
sScript = sScript + "</SCRIPT" + ">";
citar+=sScript;
var rp_img3 = '<img alt="" src=';
rp_img3 += '"images/rp_img3.png" style="width: 46px; height: 40px">';
var tabla = '<div id=div_Respuestas ><table CELLPADDING=3 CELLSPACING=1><TBODY>';
tabla += '<tr><td style="width: 46px;" align="left" VALIGN = "bottom">' + rp_img3 + '</td>';
tabla += '<td id="us">' + resultado[i * 5 + 1] + '</td>';
tabla += '<td id="contador" valign="top">' + cont + '</td></tr><tr><td></td>';
tabla += '<td id="question" style="width:510px;" text-align="justify" VALIGN="TOP">' + (resultado[i*5]); + '</td>';
tabla += '<td></td></tr>';
tabla += '<tr><td class="date">' + resultado[i * 5 + 2] + ' ' + resultado[i * 5 + 3] + '</td><td></td>';
tabla += '<td>' + citar + '</td></tr></TBODY></table></div><br/>';
document.getElementById("Preguntas").innerHTML += tabla + sScript;
}
}

function busqueda() {
    var id_foro = document.getElementById('lbl_idforo').innerHTML;
    Samples.Aspnet.Preguntas.ConsultaExistenPreguntas(id_foro, success_busqueda);
}

function success_busqueda(resultado) {
    if (resultado)
        mostrar_preguntas();
}

function agrega_pregunta() {
    var p = document.getElementById('pregyresp').innerHTML;
    var id_foro = document.getElementById('lbl_idforo').innerHTML;
    var usuario = document.getElementById('lbl_nusuario').innerHTML;
    var h = document.getElementById('lbl_relog').innerHTML;
    Samples.Aspnet.Preguntas.AgregarPregunta(id_foro, p, usuario, fecha(),h);
    document.getElementById('pregyresp').innerHTML = "";
    estado_boton = false;
    alert("Tu Pregunta ha sido agregada, Gracias por utilizar los servicios del Foro");
    mostrar_preguntas();
}

function fecha() {
    var f = new Date();
    var d = f.getDate();
    if (d < 10) d = "0" + d + "/";
    else d = d + "/";
    var m = f.getMonth();
    if (m < 10) m = "0" + m + "/";
    else m = m + "/";
    var y = f.getFullYear();
    return d + m + y;
}

function _en_blanco() {
    if (document.getElementById('pregyresp').innerHTML == "")
        alert("Debes agregar texto...!!!");
    else {
        if (estado_boton)
            agrega_pregunta();
        else
            agrega_respuesta();
    }
}
}

```

```

function responder_pregunta(id_pregunta) {
    idglobal = id_pregunta;
    mostrar_respuestas();
    document.getElementById("sugerencia").innerHTML = "Responder a la pregunta";
    estado_boton = false;
}

function mostrar_respuestas() {
    Samples.Aspnet.Respuestas.ConsultaRespuestas(idglobal, success_respuestas);
}

function success_respuestas(arreglo) {
    document.getElementById("Preguntas").innerHTML = "";
    var retornar = '<input type="button" id="boton_retorna" value="Regresar" class="boton_retorna" onclick="retornar()" />';
    document.getElementById("Preguntas").innerHTML = retornar + '<br/><br/>';
    for (var i = 0; i < arreglo.length / 4; i++) {
        var cont = i + 1;
        var rp_img3 = '<img alt="" src=';
        rp_img3 += '"images/rp_img3.png" style="width: 46px; height: 40px">';
        var tabla = '<div id=div_Respuestas ><table CELLPADDING=3 CELLSPACING=1 ><TBODY>';
        tabla += '<tr><td style="width: 46px" class="avatar" VALIGN = "bottom">' + rp_img3 + '</td>';
        tabla += '<td id="us" VALIGN = "bottom" style="width: 50px">' + arreglo[i * 4 + 1] + '</td>';
        tabla += '<td class="avatar" style="width: 25px" VALIGN = "bottom">' + arreglo[i * 4 + 2] + '</td><td class="avatar" style="width: 400px" VALIGN = "bottom">' + arreglo[i * 4 + 3] + '</td></tr>';
        tabla += '<tr><th colspan="4" id="question"text-align="justify"VALIGN="TOP">' + (arreglo[i * 4]) + '</th></tr>';
        tabla += '</TBODY></table></div>';
        document.getElementById("Preguntas").innerHTML += tabla + '<br/>';
    }
    document.getElementById("Preguntas").innerHTML += '<br/>' + retornar;
}

function agrega_respuesta() {
    var r = document.getElementById('pregyresp').innerHTML;
    var usuario = document.getElementById('lbl_nusuario').innerHTML;
    var h = document.getElementById('lbl_relog').innerHTML;
    Samples.Aspnet.Respuestas.AgregarRespuesta(idglobal, r, usuario, fecha(), h);
    document.getElementById('pregyresp').innerHTML = "";
    alert("Tu Respuesta ha sido agregada, Gracias por contribuir a la comunidad...!!!");
    mostrar_respuestas();
}

function retornar() {
    estado_boton = true;
    document.getElementById("sugerencia").innerHTML = "Envia una pregunta al foro :";
    mostrar_preguntas()
}

```

El archivo “Respuestas.js” es un script de JavaScript por medio del cual se hace la invocación a los métodos del Servicio Web “Respuestas.asmx” y los valores que retorna este son manipulados por DOM para ser presentados posteriormente en la GUI.

```

function mostrar_preguntas() {
    var id_foro = document.getElementById('lbl_idforo').innerHTML;
    Samples.Aspnet.Preguntas.ConsultaPreguntas(id_foro, success_preguntas);
}

function success_preguntas(resultado) {
    for (var i = 0; i < resultado.length / 4; i++) {
        var cont = i + 1;
        var citar = '<a href="" + pag + ""><img alt="" src=';
    }
}

```

```

        citar += "images/citar.png" style="width: 82px; height: 25px"></a>;
        var rp_img3 = '<img alt="" src=';
        rp_img3 += "images/rp_img3.png" style="width: 46px; height: 40px">;
        var tabla = '<table BORDER=1 RULES=GROUPS CELLPADDING=3 CELLSPACING=1
FRAME=HSIDES><TBODY>';
        tabla+='\<tr><td style="width: 46px;" align="left" VALIGN = "bottom">' + rp_img3 + '</td>';
        tabla+='\<td id="us">' + resultado[i * 4 + 1] + '</td>';
        tabla+='\<td id="contador" valign="top">' + cont + '</td></tr>';
        tabla+='\<tr><td></td>';
        tabla+='\<td id="question" style="width: 368px;" text-align="justify" VALIGN="TOP">' + (resultado[i*4]) + '</td>';
        tabla+='\<td></td></tr>';
        tabla+='\<tr><td id="date">' + resultado[i * 4 + 2] + ' ' + resultado[i * 4 + 3] + '</td><td></td>';
        tabla+='\<td>' + citar + '</td></tr>';
        tabla+='\</TBODY></table><br>';
        document.getElementById("Respuestas").innerHTML += tabla;
    }
}

function busqueda() {
    var id_foro = document.getElementById('lbl_idforo').innerHTML;
    Samples.AspNet.Preguntas.ConsultaExistenPreguntas(id_foro, success_busqueda);
}

function success_busqueda(resultado) {
    if (resultado)
        mostrar_preguntas();
}

function agrega_pregunta() {
    var p = document.getElementById('text_pregunta').innerHTML;
    var id_foro = document.getElementById('lbl_idforo').innerHTML;
    var usuario = document.getElementById('lbl_nusuario').innerHTML;
    var h = document.getElementById('lbl_relog').innerHTML;
    Samples.AspNet.Preguntas.AgregarPregunta(id_foro, p, usuario, fecha(), h);
    document.getElementById('text_pregunta').innerHTML = "";
    alert("Tu Pregunta ha sido agregada, y la podras ver la proxima vez que visites el Foro");
}

function fecha() {
    var f = new Date();
    var d = f.getDate();
    if (d < 10)
        d = "0" + d + "/";
    else
        d = d + "/";
    var m = f.getMonth();
    if (m < 10)
        m = "0" + m + "/";
    else
        m = m + "/";
    var y = f.getFullYear();
    return d + m + y;
}

function _en_blanco() {
    if (document.getElementById('text_pregunta').innerHTML == "")
        alert("Debes agregar texto...!!!");
    else
        agrega_pregunta();
}

```

El archivo “Foro.aspx” es el documento generado con ASP.NET el cual proporciona la GUI para esta herramienta y contiene todas las especificaciones necesarias para que puedan ejecutarse correctamente los archivos de JavaScript, CSS y los Servicios Web.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Foro.aspx.cs" Inherits="Foro" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"> <title>SisAAC -Foros-</title>
  <link href="App_Themes/perfil.css" rel="stylesheet" type="text/css" />
  <link href="App_Themes/foro.css" rel="stylesheet" type="text/css" />
  <link href="App_Themes/Utilerias.css" rel="stylesheet" type="text/css" />
</head>
<body onload="actualizaReloj() ; busqueda() ; menu_nav()">
  <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
      <Services>
        <asp:ServiceReference Path="http://localhost/PreguntasWS/Preguntas.asmx" />
        <asp:ServiceReference Path="http://localhost/RespuestasWS/Respuestas.asmx" />
      </Services>
      <Scripts>
        <asp:ScriptReference Path="~/App_Code_JS/Preguntas.js" />
        <asp:ScriptReference Path="~/App_Code_JS/Utilerias.js" />
      </Scripts>
    </asp:ScriptManager>
    <div id="encabezado">
      <div id="logo">
        <div id="login">Usuario:
          <asp:Label ID="lbl_nusuario" runat="server" ForeColor="#E1BA71"></asp:Label>&nbsp; Sesion:&nbsp;
          <asp:Label ID="lbl_idforo" runat="server" ForeColor="#E1BA71"></asp:Label>
        </div>
      </div>
      <div id="menu">
      </div>
    </div><br />
    <div id="contenedor">
      <div id="left"> <br />
        <div id="grupo" class="herramientas" align="center">
          <h2 class="titlat">Grupo Actual</h2>
          <asp:Label ID="Label6" runat="server" Font-Bold="True" Font-Size="Large" ForeColor="#960F00"></asp:Label>
        </div> <br />
        <div id="grupos" class="herramientas">
          <h2 class="titlat">Mis grupos de trabajo</h2>
          <asp:UpdatePanel ID="UpdatePanel3" runat="server">
            <ContentTemplate>
              <asp:ListBox ID="ListBox2" runat="server" Height="200px" Width="199px"></asp:ListBox><br />
              <table style="width: 89%;"><tr> <td>&nbsp;</td><td>
                <asp:Button ID="AbrirGrupo" runat="server" onclick="AbrirGrupo_Click"
                  style="margin-left: 66px; margin-right: 0px" Text="Entrar" Width="58px" /> </td><td>&nbsp;</td></tr>
              </table>
            </ContentTemplate>
          </asp:UpdatePanel>
        </div> <br />
        <div id="Div1" class="herramientas" align="center">
          <h2 class="titlat">Foro Actual</h2>
          <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Large"
            ForeColor="#960F00"></asp:Label>
        </div> <br />
        <div id="ForosCreados" class="herramientas">
          <h2 class="titlat">Foros Del Grupo Actual</h2>
          <asp:UpdatePanel ID="UpdatePanel2" runat="server">
```

```

<ContentTemplate>
    <asp:ListBox ID="ListBox1" runat="server" Height="200px" Width="199px">
    </asp:ListBox><br />
    <table style="width: 89%; "> <tr><td >&nbsp;</td><td>
        <asp:Button ID="AbrirForo" runat="server" onclick="AbrirForo_Click"
            style="margin-left: 66px; margin-right: 0px" Text="Entrar" Width="58px" /></td><td>&nbsp;</td></tr>
    </table>
</ContentTemplate>
</asp:UpdatePanel>
</div> <br />
</div>
<div id="centro">
    <div id="titulo">
        <h2> <asp:Label ID="Label4" runat="server" ForeColor="#960F00"></asp:Label></h2>
    </div>
    <div id="Preguntas" >
    </div>
    <div id="comentario"> <br /><br />
    <p id="sugerencia" >Envia una pregunta al foro :</p>
        <textarea id="pregyresp" name="comment" cols="60" rows="10" ></textarea><br />
    <input type="button" id="boton_pregunta" value="Enviar" class="boton" onclick="_en_blanco()" /> <br />
    <p id="recomendaciones" style="font-size: small; text-align: justify; ">El equipo de SisAAC se
        reserva el derecho de moderaci&oacute;n de los comentarios. Evita utilizar palabras soeces, ataques directos,
        descalificativos, insultos,
        etc. Por una cultura al respeto. </p>
    </div>
</div><br />
<div id="lateral">
    <div id="relog" class="herramientas">
        <h2 class="titlat">Relog</h2>
        <p align="center">
            <asp:Label ID="lbl_relog" runat="server" Font-Bold="True"
                Font-Size="Large" ForeColor="#960F00"></asp:Label></p>
    </div><br />
        <div id="calendario" class="herramientas">
            <h2 class="titlat">Calendario</h2>
            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>
                    <asp:Calendar ID="Calendar1" runat="server" BackColor="#FFFFCC"
                        BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest"
                        Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399" Height="200px"
                        ShowGridLines="True" Width="200px">
                        <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
                        <SelectorStyle BackColor="#FFCC66" />
                        <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
                        <OtherMonthDayStyle ForeColor="#CC9966" />
                        <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
                        <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
                        <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
                            ForeColor="#FFFFCC" />
                    </asp:Calendar>
                </ContentTemplate>
            </asp:UpdatePanel>
        </div><br />
    </div>
</div>
<div id="pie">
    <p>Copyright © 2010 . All Rights Reserved. SISAAC ver 1.0<br />
        Benemerita Universidad Autonoma de Puebla<br />
        Facultad de Ciencias de la Computaci&oacute;n</p>
    <p><asp:Label ID="lbl_idusuario" runat="server" ForeColor="#DFDFC5"></asp:Label></p>
</div>
</div></form></body></html>

```

El archivo “Foro.aspx.cs” contiene el código en C#.NET llamado “CodeBehind” (Código adjunto), a través del cual se invocan a los métodos de los Servicios Web dentro del entorno de ASP.NET y se codifica el funcionamiento de todos los objetos pertenecientes a este.

```
using System;
using System.Data;

public partial class Foro : System.Web.UI.Page{
    private DataRow dr;
    DataTable datos = new DataTable();
    localhost.Grupos groups = new localhost.Grupos();
    localhost.Sesion sesion = new localhost.Sesion();
    localhost.Foro forum = new localhost.Foro();
    System.DateTime fechaA = new System.DateTime();
    validated validar = new validated();
    string id, alias, idg, ng, idf, nf, fecha;

    protected void Page_Load(object sender, EventArgs e) {
        id = Request.QueryString["id"];
        alias = Request.QueryString["alias"];
        idg = Request.QueryString["idg"];
        ng = Request.QueryString["ng"];
        idf=Request.QueryString["idf"];
        nf=Request.QueryString["nf"];
        fechaA =DateTime.Now;
        fecha =fechaA.ToString("dd/MM/yyyy");
        Label4.Text = nf;
        lbl_nusuario.Text = alias;
        lbl_idforo.Text = idf;
        Label6.Text = ng;
        Label1.Text = nf;
        lbl_idusuario.Text = id;
        if (!Page.IsPostBack) {
            datos = forum.RetornaForosGrupo(idg);
            try{
                lbl_nusuario.Text = alias;
                int x = datos.Rows.Count;
                for (int i = 0; i < datos.Rows.Count; i++){
                    dr = datos.Rows[i];
                    ListBox1.Items.Add(dr["nomforo"].ToString());
                }
            }
            catch (Exception ex) { Console.Write(ex.Message); }
            datos = sesion.ConsultaGrupos(id);
            try{
                for (int i = 0; i < datos.Rows.Count; i++){
                    dr = datos.Rows[i];
                    ListBox2.Items.Add(groups.Retorna_nombregrupo((dr["idgrupo"].ToString())));
                }
            }
            catch (Exception ex) { Console.Write(ex.Message); }
        }
    }
}
```

```

protected void AbrirForo_Click(object sender, EventArgs e) {
    if (ListBox1.Text != ""){
        nf = ListBox1.Text;
        idf= forum.Retorna_Idforo(ListBox1.Text);
        Response.Redirect("Foro.aspx?id=" + id + "&alias=" + alias + "&idg=" + idg + "&ng=" + ng +
"&idf=" + idf + "&nf=" + nf);
    }
    else
        validated.MostrarMensaje("Debes seleccionar un Foro", this.Page);
}

protected void AbrirGrupo_Click(object sender, EventArgs e) {
    if (ListBox2.Text != ""){
        if (sesion.ConsultaExisteSesion(id, groups.Retorna_Idgrupo(ListBox2.Text))) {
            ng= ListBox2.Text;
            idg= groups.Retorna_Idgrupo(ListBox2.Text);
            Response.Redirect("Grupo.aspx?id=" + id + "&alias=" + alias + "&idg=" + idg + "&ng=" + ng);
        }
        else{
            sesion.CrearSesion(id, groups.Retorna_Idgrupo(ListBox2.Text), fecha);
            ng= ListBox2.Text;
            Label4.Text = "";
            idg = groups.Retorna_Idgrupo(ListBox2.Text);
            Response.Redirect("Grupo.aspx?id=" + id + "&alias=" + alias + "&idg=" + idg + "&ng=" + ng);
        }
    }
    else
        validated.MostrarMensaje("Debes seleccionar un Grupo", this.Page);
}
}

```

Herramienta colaborativa “Espacio para compartir archivos”

Servicio Web “Archivos.asmx”. Este Servicio Web proporciona los métodos: AgregarArchivo, ConsultaArchivos, ConsultaExistenArchivo, EliminaArchivo, get_clave, get_ruta, get_idgrupo, get_idgrupo, InfoArchivo y CargaNarchivos, por medio de los cuales se guardan y extraen las referencias de un archivo en de la base de datos.

```

using System.Web.Services;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System;

namespace Samples.AspNet {
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.Web.Script.Services.ScriptService]
    public class Archivos : System.Web.Services.WebService{
        private SqlDataAdapter da;
        private DataRow dr;
        private DataSet ds;

        public Archivos() { }

        [WebMethod(Description = " Metodo Guarda un archivo en una Base de Datos")]

```

```

public string agregarArchivo(string nombre, string ruta, string contentType, string comentario, string fecha, string
usuario, string IdGrupo) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT * FROM UploadFiles";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    DataSet ds = new DataSet();
    da.UpdateCommand = cb.GetUpdateCommand();
    da.InsertCommand = cb.GetInsertCommand();
    da.DeleteCommand = cb.GetDeleteCommand();
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    try {
        da.Fill(dt);
        dr = dt.NewRow();
        dr["nombre"] = nombre;
        dr["ruta"] = ruta;
        dr["Mimetype"] = contentType;
        dr["comentario"] = comentario;
        dr["fecha"] = fecha;
        dr["usuario"] = usuario;
        dr["IdGrupo"] = IdGrupo;
        dt.Rows.Add(dr);
        da.Update(dt);
        dt.AcceptChanges();
    }
    catch (Exception ex) { Console.WriteLine(ex.Message); }
    finally{
        if (myConnection != null)
            myConnection.Close();
    }
    return "archivo cargado exitosamente ";
}

```

```

[WebMethod(Description = " Metodo que retorna un datatable con todos los Archivos del grupo")]
public DataTable ConsultaArchivos(string idgrupo) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT * FROM UploadFiles WHERE IdGrupo= '" + idgrupo + "'";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    da.UpdateCommand = cb.GetUpdateCommand();
    da.InsertCommand = cb.GetInsertCommand();
    da.DeleteCommand = cb.GetDeleteCommand();
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    try{da.Fill(dt); }
    catch (Exception ex) {
        Console.WriteLine(ex.Message);
        return null;
    }
    finally{
        if (myConnection != null)
            myConnection.Close();
    }
    return dt;
}

```

```

[WebMethod(Description = " Metodo que checa si ya existen un archivo en un grupo")]
public bool ConsultaExistenArchivo(string nombre, string idgrupo) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;

```

```

        SqlConnection myConnection = new SqlConnection(strConnectionString);
        string strCommandText = "SELECT nombre FROM UploadFiles WHERE IdGrupo= '" + idgrupo + "' AND
nombre= '" + nombre + "'";
        da = new SqlDataAdapter(strCommandText, myConnection);
        ds = new DataSet();
        try{da.Fill(ds, "UploadFiles");}
        catch (Exception ex) { Console.WriteLine(ex.Message); }
        finally {
            if (myConnection != null)
                myConnection.Close();
        }
        if (ds.Tables["UploadFiles"].Rows.Count > 0)
            return true;
        else
            return false;
    }

    [WebMethod(Description = " Metodo Eliminar un Archivo")]
    public void EliminaArchivo(string clave) {
        string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
        SqlConnection myConnection = new SqlConnection(strConnectionString);
        SqlCommand cmd = new SqlCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "DELETE FROM UploadFiles WHERE clave= '" + clave + "'";
        cmd.Connection = myConnection;
        myConnection.Open();
        cmd.ExecuteNonQuery();
        myConnection.Close();
    }

    [WebMethod(Description = " Metodo que retorna la clave de un archivo")]
    public string get_clave(string nombre) {
        string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
        SqlConnection myConnection = new SqlConnection(strConnectionString);
        string strCommandText = "SELECT clave FROM UploadFiles WHERE nombre = '" + nombre + "'";
        da = new SqlDataAdapter(strCommandText, myConnection);
        SqlCommandBuilder cb = new SqlCommandBuilder(da);
        da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
        DataTable dt = new DataTable();
        try { da.Fill(dt); }
        catch (Exception ex) {
            Console.WriteLine(ex.Message);
            return null;
        }
        finally {
            if (myConnection != null)
                myConnection.Close();
        }
        string mensaje;
        if (dt.Rows.Count > 0)
            mensaje = dt.Rows[0][0].ToString();
        else
            mensaje = null;
        return mensaje;
    }

    [WebMethod(Description = " Metodo que retorna la ruta de un archivo de un usuario")]
    public string get_ruta(string nombre) {
        string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
        SqlConnection myConnection = new SqlConnection(strConnectionString);
        string strCommandText = "SELECT ruta FROM UploadFiles WHERE nombre = '" + nombre + "'";
        da = new SqlDataAdapter(strCommandText, myConnection);
    }

```

```

SqlCommandBuilder cb = new SqlCommandBuilder(da);
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
DataTable dt = new DataTable();
try { da.Fill(dt); }
catch (Exception ex) {
    Console.WriteLine(ex.Message);
    return null;
}
finally {
    if (myConnection != null)
        myConnection.Close();
}
string mensaje;
if (dt.Rows.Count > 0)
    mensaje = dt.Rows[0][0].ToString();
else
    mensaje = null;
return mensaje;
}

[WebMethod(Description = " Metodo que retorna id del grupo al que pertenece un archivo de un usuario")]
public string get_idgrupo(string nombre) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT idgrupo FROM UploadFiles WHERE nombre = '" + nombre + "'";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    try { da.Fill(dt); }
    catch (Exception ex) {
        Console.WriteLine(ex.Message);
        return null;
    }
    finally {
        if (myConnection != null)
            myConnection.Close();
    }
    string mensaje;
    if (dt.Rows.Count > 0)
        mensaje = dt.Rows[0][0].ToString();
    else
        mensaje = null;
    return mensaje;
}

[WebMethod(Description = " Metodo que retorna los datos de un archivo apartir del nombre usuario")]
public string[] InfoArchivo(string nombre, string idgrupo) {
    string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT grupo.nomgrupo, UploadFiles.nombre, UploadFiles.mimetype,
UploadFiles.fecha, UploadFiles.usuario, UploadFiles.comentario "+
        "FROM grupo, UploadFiles WHERE uploadfiles.nombre = '" + nombre + "' "+
        "and uploadfiles.idgrupo = '"+idgrupo+"' and grupo.idgrupo = '"+idgrupo+"' ";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    string[] mensaje;
    try { da.Fill(dt); }
    catch (Exception ex) {
        Console.WriteLine(ex.Message);
        return null;
    }
}

```

```

        finally {
            if (myConnection != null)
                myConnection.Close();
        }
        mensaje = new string[dt.Rows.Count * 6];
        if (dt.Rows.Count > 0) {
            for (int i = 0; i < dt.Rows.Count; i++) {
                mensaje[i * 6] = dt.Rows[0][0].ToString();
                mensaje[i * 6 + 1] = dt.Rows[0][1].ToString();
                mensaje[i * 6 + 2] = dt.Rows[0][2].ToString();
                mensaje[i * 6 + 3] = dt.Rows[0][3].ToString();
                mensaje[i * 6 + 4] = dt.Rows[0][4].ToString();
                mensaje[i * 6 + 5] = dt.Rows[0][5].ToString();
            }
        }
        else
            mensaje = null;
        return mensaje;
    }

    [WebMethod(Description = " Metodo que retorna los nombres de los archivos de un usuario")]
    public string[] CargaNarchivos(string usuario) {
        string strConnectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
        SqlConnection myConnection = new SqlConnection(strConnectionString);
        string strCommandText = "SELECT nombre FROM UploadFiles WHERE usuario= '"+ usuario + "' ORDER BY
        fecha DESC";
        da = new SqlDataAdapter(strCommandText, myConnection);
        SqlCommandBuilder cb = new SqlCommandBuilder(da);
        da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
        DataTable dt = new DataTable();
        try { da.Fill(dt); }
        catch (Exception ex) {
            Console.WriteLine(ex.Message);
            return null;
        }
        finally {
            if (myConnection != null)
                myConnection.Close();
        }
        string[] mensaje = new string[dt.Rows.Count];
        if (dt.Rows.Count > 0) {
            for (int i = 0; i < dt.Rows.Count; i++)
                mensaje[i] = dt.Rows[i][0].ToString();
        }
        else
            mensaje = null;
        return mensaje;
    }
}

```

Herramienta colaborativa “Buzón de mensajes”

Servicio Web “MensajesUsuarios.aspx”. Este Servicio Web proporciona los métodos: AddMessage, EliminaMensaje, ConsultaMensajesUsuario, ConsultaExistenMensajes y ConsultaMensajeUsuario, los cuales proveen los medios necesarios para que un mensaje sea almacenado en la base de datos y posteriormente el usuario pueda manipularlos desde la GUI.

```

using System.Web.Services;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System;

namespace Samples.AspNet{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.Web.Script.Services.ScriptService]
    public class MensajesUsuarios : System.Web.Services.WebService {
        protected SqlDataAdapter da;
        protected DataSet ds;
        private DataRow dr;
        protected SqlConnection myConnection = new SqlConnection();

        [WebMethod(Description = " Metodo que Crea un Mensaje")]
        public void AddMessage(string idusuario, string idusdestino, string asunto, string mensaje, string fecha) {
            string strConnectionString =
                ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
            SqlConnection myConnection = new SqlConnection(strConnectionString);
            string strCommandText = "SELECT * FROM mensajesusuarios ";
            da = new SqlDataAdapter(strCommandText, myConnection);
            SqlCommandBuilder cb = new SqlCommandBuilder(da);
            da.UpdateCommand = cb.GetUpdateCommand();
            da.InsertCommand = cb.GetInsertCommand();
            da.DeleteCommand = cb.GetDeleteCommand();
            da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
            DataTable dt = new DataTable();
            try {
                da.Fill(dt);
                dr = dt.NewRow();
                dr["idusuario"] = idusuario;
                dr["idusdestino"] = idusdestino;
                dr["asunto"] = asunto;
                dr["mensaje"] = mensaje;
                dr["fecha"] = fecha;
                dt.Rows.Add(dr);
                da.Update(dt);
                dt.AcceptChanges();
            }
            catch (Exception ex) { Console.WriteLine(ex.Message); }
            finally {
                if (myConnection != null)
                    myConnection.Close();
            }
        }

        [WebMethod(Description = " Metodo que retorna todos los mensajes pertenecientes a un usuario")]
        public string[] ConsultaMensajesUsuario(string idusuario) {
            string strConnectionString =
                ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
            SqlConnection myConnection = new SqlConnection(strConnectionString);
            string strCommandText = "SELECT MensajesUsuarios.idusuario, MensajesUsuarios.asunto,
                MensajesUsuarios.fecha, usuarios.nusuario, MensajesUsuarios.clave FROM MensajesUsuarios, usuarios " +
                "where MensajesUsuarios.idusdestino ='" + idusuario + "'" +
                " and usuarios.idusuario = MensajesUsuarios.idusuario ";
            da = new SqlDataAdapter(strCommandText, myConnection);
            SqlCommandBuilder cb = new SqlCommandBuilder(da);
            da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
            DataTable dt = new DataTable();
            string[] mensaje;

```

```

        try {da.Fill(dt); }
        catch (Exception ex) {
            Console.WriteLine(ex.Message);
            return null;
        }
    finally{
        if (myConnection != null)
            myConnection.Close();
    }
    mensaje = new string[dt.Rows.Count * 5];
    if (dt.Rows.Count > 0) {
        for (int i = 0; i < dt.Rows.Count; i++){
            mensaje[i * 5] = dt.Rows[i][0].ToString();
            mensaje[i * 5 + 1] = dt.Rows[i][1].ToString();
            mensaje[i * 5 + 2] = dt.Rows[i][2].ToString();
            mensaje[i * 5 + 3] = dt.Rows[i][3].ToString();
            mensaje[i * 5 + 4] = dt.Rows[i][4].ToString();
        }
    }
    return mensaje;
}

[WebMethod(Description = " Metodo que checa si existen mensajes para un usuario ")]
public bool ConsultaExistenMensajes(string idusuario) {
    string strConnectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT * FROM MensajesUsuarios WHERE idusdestino= '" + idusuario + "'";
    da = new SqlDataAdapter(strCommandText, myConnection);
    ds = new DataSet();
    try{ da.Fill(ds, "MensajesUsuarios"); }
    catch (Exception ex) { Console.WriteLine(ex.Message); }
    finally{
        if (myConnection != null)
            myConnection.Close();
    }
    if (ds.Tables["MensajesUsuarios"].Rows.Count > 0)
        return true;
    else
        return false;
}

[WebMethod(Description = " Metodo que retorna los datos de un mensaje apartir de su clave")]
public string[] ConsultaMensajeUsuario(string idusuario, string clave) {
    string strConnectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection(strConnectionString);
    string strCommandText = "SELECT
usuarios.nusuario,MensajesUsuarios.asunto,MensajesUsuarios.fecha,MensajesUsuarios.mensaje FROM usuarios,
MensajesUsuarios " +
        "where MensajesUsuarios.idusuario = '" + idusuario + "' and usuarios.idusuario='" + idusuario + "'"+
        "and MensajesUsuarios.clave='" + clave + "'";
    da = new SqlDataAdapter(strCommandText, myConnection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
    DataTable dt = new DataTable();
    string[] mensaje;
    try { da.Fill(dt); }
    catch (Exception ex) {
        Console.WriteLine(ex.Message);
        return null;
    }
}

```

```

        finally{
            if (myConnection != null)
                myConnection.Close();
        }
        mensaje = new string[4];
        if (dt.Rows.Count > 0) {
            mensaje[0] = dt.Rows[0][0].ToString();
            mensaje[1] = dt.Rows[0][1].ToString();
            mensaje[2] = dt.Rows[0][2].ToString();
            mensaje[3] = dt.Rows[0][3].ToString();
        }
        return mensaje;
    }

    [WebMethod(Description = " Metodo Elimina un mensaje de un Usuario")]
    public Boolean EliminaMensaje(string clave) {
        string strConnectionString =
        ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
        SqlConnection myConnection = new SqlConnection(strConnectionString);
        SqlCommand cmd = new SqlCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "DELETE FROM MensajesUsuarios WHERE clave = '" + clave + "'";
        cmd.Connection = myConnection;
        myConnection.Open();
        cmd.ExecuteNonQuery();
        myConnection.Close();
        return true;
    }
}

```

Herramienta colaborativa “Blog”

Servicio Web “Blog.aspx”. Este Servicio Web proporciona los métodos: CrearBlog, retorna_Idblog y ConsultaExisteBlog, los cuales proveen los medios necesarios para la creación de un nuevo registro en la base de datos el cual contenga la información concerniente al blog de un usuario.

```

using System.Web.Services;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System;

namespace Samples.Aspnet {
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.Web.Script.Services.ScriptService]
    public class Blog : System.Web.Services.WebService {
        private SqlDataAdapter da;
        private DataRow dr;
        DataSet ds;

        public Blog() { }

        [WebMethod(Description = " Metodo que Crea el Blog de un Usuario")]
        public void CrearBlog(string IdUsuario) {
            string strConnectionString =
            ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
            SqlConnection myConnection = new SqlConnection(strConnectionString);

```

```

string strCommandText = "SELECT * FROM Blog";
da = new SqlDataAdapter(strCommandText, myConnection);
SqlCommandBuilder cb = new SqlCommandBuilder(da);
da.UpdateCommand = cb.GetUpdateCommand();
da.InsertCommand = cb.GetInsertCommand();
da.DeleteCommand = cb.GetDeleteCommand();
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
DataTable dt = new DataTable();
try {
    da.Fill(dt);
    dr = dt.NewRow();
    dr["IdUsuario"] = IdUsuario;
    dt.Rows.Add(dr);
    da.Update(dt);
    dt.AcceptChanges();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }
finally{
    if (myConnection != null)
        myConnection.Close();
}
}

[WebMethod(Description = " Metodo que retorna el Id de un blog a partir del id ded usuario")]
public string retorna_Idblog(string idusuario) {
    string strConnectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
SqlConnection myConnection = new SqlConnection(strConnectionString);
string strCommandText = "SELECT idblog FROM Blog where idusuario=" + idusuario + """;
da = new SqlDataAdapter(strCommandText, myConnection);
SqlCommandBuilder cb = new SqlCommandBuilder(da);
da.UpdateCommand = cb.GetUpdateCommand();
da.InsertCommand = cb.GetInsertCommand();
da.DeleteCommand = cb.GetDeleteCommand();
da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
DataTable dt = new DataTable();
string id;
try{
    da.Fill(dt);
}
catch (Exception ex) {
    Console.WriteLine(ex.Message);
    return null;
}
finally{
    if (myConnection != null)
        myConnection.Close();
}
if (dt.Rows.Count > 0)
    id = dt.Rows[0][0].ToString();
else
    id = "no_blog";
return id;
}

[WebMethod(Description = " Metodo que checa si existe un Blog para un usuario determinado")]
public bool ConsultaExisteBlog(string idusuario) {
    string strConnectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
SqlConnection myConnection = new SqlConnection(strConnectionString);
string strCommandText = "SELECT * FROM blog WHERE idusuario= " + idusuario + """;
da = new SqlDataAdapter(strCommandText, myConnection);

```

```

        ds = new DataSet();
        try{da.Fill(ds, "blog"); }
        catch (Exception ex) { Console.WriteLine(ex.Message); }
        finally{
            if (myConnection != null)
                myConnection.Close();
        }
        if (ds.Tables["blog"].Rows.Count > 0)
            return true;
        else
            return false;
    }
}
}

```

El archivo “BlogAndTools.js” contiene las funciones que invocan los métodos del Servicio Web que implementa esta herramienta haciendo uso de DOM para la manipulación de los datos retornados.

```

function crearBlog(){
    Samples.Aspnet.Blog.CrearBlog(document.getElementById("lbl_idusuario").innerHTML);
}

function subirMensaje() {
    if (document.getElementById("TextArea1").value != "") {
        Samples.Aspnet.MensajesBlog.AgregarMensaje(document.getElementById("lbl_idblog").innerHTML,
        document.getElementById("TextArea1").value, fecha())
        document.getElementById("TextArea1").value = "";
        alert("Tu comentario ah sido agregado a tu Blog");
    }
    else
        alert("Debes agregar un comentario");
}

function fecha() {
    var f = new Date();
    var d = f.getDate();
    if (d < 10) d = "0" + d + "/";
    else d = d + "/";
    var m = f.getMonth();
    if (m < 10) m = "0" + m + "/";
    else m = m + "/";
    var y = f.getFullYear();
    return d+m+y;
}

function retornaImagenesBlog() {
    Samples.Aspnet.Imagen.retornaImagenes(document.getElementById("lbl_idblog").innerHTML,
    document.getElementById("lbl_idusuario").innerHTML, success_retornaImagenesBlog);
}

function success_retornaImagenesBlog(resultado) {
    if (resultado != null) {
        document.getElementById("fotos").innerHTML = "<br/><br/>";
        for (var i = 0; i < resultado.length / 3; i++) {
            var tabla = '<table><tr><td width="75px" height="auto" rowspan="2" style="vertical-align:text-top" ><img alt="" src="" + resultado[i * 3 + 1] + "" style="width:75px; height:auto" /></td><td width="550" height="auto" style="font-size:10px">' + resultado[i * 3] + '</td></tr>';
            tabla+='<tr><td width="550" height="auto" style="font-size:15px;text-align:justify;">' + resultado[i*3+2] + '</td></tr></table>';
            document.getElementById("fotos").innerHTML += '<br/>' + tabla;
        }
    }
}

```

```

    }
}

function retorna_mensajesBlog() {
Samples.Aspnet.MensajesBlog.ConsultaMensajes(document.getElementById("lbl_idblog").innerHTML,succes_retorna_mensajesBlog)
}

function success_retorna_mensajesBlog(resultado) {
    if (resultado != null) {
        document.getElementById("com").innerHTML += "";
        for (var i = 0; i < resultado.length / 2; i++) {
            var res = i % 2;
            if (res == 0) {
                var mens = '<div class="msgblog"><table><tr><td width="200px" style="background:#dfdfc5; font-size:11px; color:#9e1d04">' + resultado[i * 2] + '</td></tr>';
                mens += '<tr><td width="200px">' + resultado[i * 2 + 1] + '</td></tr></table></div>';
                document.getElementById("com").innerHTML += mens + '<br/>';
            }
            else{
                var mens = '<div class="msgblog"><table><tr><td width="200px" style="background:#dfdfc5; font-size:11px; " >' + resultado[i * 2] + '</td></tr>';
                mens += '<tr><td width="200px" style="color:#9e1d04">' + resultado[i * 2 + 1] + '</td></tr></table></div>';
                document.getElementById("com").innerHTML += mens + '<br/>';
            }
        }
    }
}

function retorna_nomcom() {
Samples.Aspnet.Perfil.getnomcom(document.getElementById("lbl_idusuario").innerHTML,succesretornanomcom);
}

function success_retorna_nomcom(resultado) {
    document.getElementById("lbl_nombre").innerHTML = resultado[0];
    document.getElementById("lbl_comentario").innerHTML = resultado[1];
    var im= ''
    document.getElementById("foto_usuario").innerHTML = im;
    document.getElementById("datos").innerHTML = resultado[0] + '<br/>' + resultado[3] + '<br/>' + resultado[4] + '<br/>' + resultado[5] + '<br/>' + resultado[6] + '<br/>';
}

function rcd() {
Samples.Aspnet.Contactos.RCD(document.getElementById("lbl_idusuario").innerHTML, success_rcd);
}

function success_rcd(resultado) {
    if (resultado != null) {
        for (var i = 0; i < resultado.length / 3; i++) {
            var tabla = '<div class="contactos" width="195px"><table><tr><td rowspan="2" width:"50px" height="auto"><a href="Blog.aspx?name=' + resultado[i * 2 + 1] + "' ><img alt="" src="" + resultado[i * 2] + "' style="width:50px; height:auto" /></a></td><td width:"145px" height="auto" >' + resultado[i * 2 + 1] + '</td></tr>';
            tabla += '<tr><td style="font-size:11px">' + resultado[i * 2 + 2].substring(0, 30) + '...</td></tr></table><div>';
            document.getElementById("fcontactos").innerHTML += tabla+'<br/>';
        }
    }
}

```

```

function rci() {
    Samples.Aspnet.Contactos.RCI(document.getElementById("lbl_idusuario").innerHTML, success_rci);
}

function success_rci(resultado) {
    if (resultado != null) {
        for (var i = 0; i < resultado.length / 3; i++) {
            var tabla = '<div class="contactos" width="195px"><table><tr><td rowspan="2" width="50px" height="auto"><a href="Blog.aspx?name=' + resultado[i * 2 + 1] + "'><img alt="" src="" + resultado[i * 2] + "' style="width:50px; height:auto" /></a></td><td width="145px" height="auto" >' + resultado[i * 2 + 1] + '</td></tr>';
            tabla += '<tr><td style="font-size:11px">' + resultado[i * 2 + 2].substring(0, 30) + '...</td></tr></table></div>';
            document.getElementById("fcontactos").innerHTML += tabla + '<br/>';
        }
    }
}

function retorna_mbinvitado() {
    Samples.Aspnet.MBInvitado.ConsultaMensajes(document.getElementById("lbl_idblog").innerHTML, success_retorna_mbinvitado);
}

function success_retorna_mbinvitado(resultado) {
    if (resultado != null) {
        for (var i = 0; i < resultado.length / 3; i++) {
            var tabla = '<div class="contactos" width="195px"><table><tr><td colspan="2" width="195px" style="font-size:9px; background:#dfdfc5">' + resultado[i * 3 + 1] + '</td></tr>';
            tabla += '<tr><td rowspan="2" width="50px" height="auto" style="vertical-align:super"><img alt="" src="" + resultado[i * 3] + "' style="width:50px; height:auto" /></a></td><td width="145px" height="auto" style="font-size:12px"><i>' + resultado[i * 3 + 2] + '</i></td></tr>';
            tabla += '<tr><td></td></tr></table></div>';
            document.getElementById("fcomus").innerHTML += tabla + '<br/>';
        }
    }
}

function imgActual() {
    var im = document.getElementById("lbl_imagen").innerHTML;
    var image = '<fieldset><legend> Imagen Actual </legend><img alt="" src="" + im + "' style="width:350px; height:auto" /></fieldset>';
    document.getElementById("foto").innerHTML = image;
}

```

Hojas de estilo (CSS)

Los principales archivos CSS que se utilizaron para la implementación de las herramientas colaborativas son mostrados a continuación.

El archivo “foro.css” implementa el formato de la pagina web “Foro.aspx”. Dicho archivo se muestra a continuación.

```

#comentario {
float:left;
height:auto;
font: italic 12px Georgia,"Times New Roman",Times,serif;
color:black;
text-align:left;
width: 591px;
}

#left{
width: 200px;
background-color: #ffffff;
float:left;
margin-left: 25px;
} // end left
#centro{
width:610px;
padding-left:8px;
background-color :#ffffff;

```

```

float:left;
text-align:center;
} // endcentro
#titulo{
float:none;
width:500px;
text-align:center;
color:#000000;
}
#Preguntas{
text-align:left;
color:#68729E;
margin-left:2px;
margin-right:2px;
}
#contador{
font:bold 25px Georgia,"Times New
Roman",Times,serif;
text-align:right;
color:#960f00;
}
#question{
font:normal 15px Georgia,"Times New
Roman",Times,serif;
text-align:justify;
}

```

```

#us{
font:bold 11px Georgia,"Times New
Roman",Times,serif;
text-align:left;
color:#960f00;
}
.date{
font:normal 10px Georgia,"Times New
Roman",Times,serif;
text-align:center;
color:#960f00;
}
.avatar{
font:normal 10px Georgia, Times New
Roman",Times,serif;
text-align:left;
color:#960f00;
}
#div_Respuestas{
border-style:solid;
border-color:#C4c4a3;
border-width:1px;
}

```

El archivo “blog.css” es el que da el formato para la herramienta colaborativa “Blog” y se muestra a continuación.

```

body {
margin-top: 0px;
padding: 0;
background:#ffffff url(../images/sisaac_blog.png)
no-repeat;
color:#000000;
}
#bar_nav{
width: 700px;
height:25px;
text-align:center;
margin-top:25px;
margin-left:270px;
background:#ce1126;
font-size: 15px;
color:#ffffff;
padding-top:10px;
}
A.enlacenav, A.enlacenav:VISITED,
A.enlacenav:ACTIVE, A.enlacenav:FOCUS,
A.enlacenav:LINK{
color: #f1a85d;
text-decoration:none;
}
A.enlacenav:HOVER {
background:#f1a85d;
color: #ffffff;
}
#principal{
clear:both;
width:1075px;

```

```

float:none;
margin-top: 0px;
padding: 25px;
clear:both;
margin: 0 auto;
} // end principal
#left{
width:205px;
float:left
}
#righ{
width:200px;
float:right
}
#center{
width:650px;
float:left;
color:#000000;
}
#foto_usuario{
width:200px;
height:auto;
float:none;
text-align:center;
}
#mensajes{
margin-top:65px;
width:auto;
border: 1px solid #ffffff;
background:#f4f8f0;
}

```

```

.mensajes_botones{
width:auto;
height:auto;
font-family:Times New Roman;
background:#f4f8f0;
}
#contactos{
width:205px;
height:auto;
border: 1px solid #ffffff;
}
#com{
float:none;
color:#000000;
padding-top:85px;

```

```

font-family:Times New Roman;
font-size:12px;
font-style:italic;
} // end com
.msgblog{
width:200px;
border: 1px solid #f1a85d;
background:#f4f8f0;
}
.contactos{
width:190px;
float:none;
border: 1px solid #f1a85d;
background:#f4f8f0;
}

```

Script de la Base de Datos

El código del archivo “sisaac.sql” que se presenta a continuación corresponde a la generación de la base de datos y fue obtenido por medio de “Erwin Data Modeler v 7.0” a partir del Modelo Relacional diseñado con la misma herramienta.

```

CREATE TABLE Blog(
    Idblog          int IDENTITY (1000,1) ,
    IdUsuario       char(9) NOT NULL
)go
ALTER TABLE Blog
    ADD CONSTRAINT XPKBlog PRIMARY KEY CLUSTERED (Idblog ASC)
go
CREATE TABLE Contactos(
    iduscontactado  char(9) NULL ,
    IdUsuario       char(9) NOT NULL ,
    clave           int IDENTITY (1000,1)
)go
ALTER TABLE Contactos
    ADD CONSTRAINT XPKContactos PRIMARY KEY CLUSTERED (clave ASC)
go
CREATE TABLE Foro(
    nomforo         varchar(255) NULL ,
    nucforo         varchar(25) NULL ,
    descripcion     text NULL ,
    fechacreacion  varchar(10) NULL ,
    Idforo          int IDENTITY (1000,1) ,
    IdGrupo         int NOT NULL
)go
ALTER TABLE Foro
    ADD CONSTRAINT XPKForo PRIMARY KEY CLUSTERED (Idforo ASC)
go
CREATE TABLE Grupo(
    IdGrupo         int IDENTITY (1000,1) ,
    nomgrupo        char(255) NOT NULL ,
    descripcion     text NULL ,
    nucgrupo        varchar(25) NULL ,
    fechacreacion  char(10) NULL
)go
ALTER TABLE Grupo
    ADD CONSTRAINT XPKGrupo PRIMARY KEY CLUSTERED (IdGrupo ASC)
go
CREATE TABLE ImagenesBlog(
    idImagen        int IDENTITY (1000,1) ,

```

```

        MimeType          varchar(255) NULL ,
        nombre            varchar(255) NULL ,
        comentario        text NULL ,
        fecha             datetime NULL ,
        Idblog            int NOT NULL ,
        ruta              varchar(255) NULL
    )go
    ALTER TABLE ImagenesBlog
        ADD CONSTRAINT XPKImagenes PRIMARY KEY CLUSTERED (idImagen ASC)
go
    CREATE TABLE MBInvitado(
        Idmensaje          int IDENTITY (1000,1) ,
        mensaje            text NULL ,
        fecha              datetime NULL ,
        Idblog            int NOT NULL ,
        nusuario           varchar(25) NULL
    )go
    ALTER TABLE MBInvitado
        ADD CONSTRAINT XPKMBInvitado PRIMARY KEY CLUSTERED (Idmensaje ASC)
go
    CREATE TABLE MensajesBlog(
        Idmensaje          int IDENTITY (1000,1) ,
        mensaje            text NULL ,
        fecha              datetime NULL ,
        Idblog            int NOT NULL
    )go
    ALTER TABLE MensajesBlog
        ADD CONSTRAINT XPKMensajesBlog PRIMARY KEY CLUSTERED (Idmensaje ASC)
go
    CREATE TABLE MensajesGrupo(
        IdMensaje          int IDENTITY (1000,1) ,
        mensaje            text NULL ,
        fechacreacion      char(10) NULL ,
        nucmensaje         varchar(25) NULL ,
        IdGrupo           int NOT NULL
    )go
    ALTER TABLE MensajesGrupo
        ADD CONSTRAINT XPKMensajesGrupo PRIMARY KEY CLUSTERED (IdMensaje ASC)
go
    CREATE TABLE MensajesUsuarios(
        clave              int IDENTITY (1000,1) ,
        idusdestino        char(9) NOT NULL ,
        mensaje            text NULL ,
        fecha              char(10) NULL ,
        asunto             text NULL ,
        IdUsuario          char(9) NOT NULL
    )go
    ALTER TABLE MensajesUsuarios
        ADD CONSTRAINT XPKMensajesUsuarios PRIMARY KEY CLUSTERED (clave ASC)
go
    CREATE TABLE Preguntas(
        Idpregunta         int IDENTITY (1000,1) ,
        nupregunta         varchar(25) NULL ,
        pregunta           text NULL ,
        fecha              char(10) NULL ,
        hora               char(8) NULL ,
        Idforo             int NOT NULL
    )go
    ALTER TABLE Preguntas
        ADD CONSTRAINT XPKPreguntas PRIMARY KEY CLUSTERED (Idpregunta ASC)
go
    CREATE TABLE Respuestas(

```

```

        Idrespuesta      int IDENTITY (1000,1) ,
        nurespuesta      varchar(25) NULL ,
        respuesta        text NULL ,
        fecha            char(10) NULL ,
        hora             char(8) NULL ,
        Idpregunta       int NOT NULL
    )go
    ALTER TABLE Respuestas
        ADD CONSTRAINT XPKRespuestas PRIMARY KEY CLUSTERED (Idrespuesta ASC)
go
    CREATE TABLE Sesion(
        IdSesion          int IDENTITY (1000,1) NOT FOR REPLICATION ,
        IdGrupo           int NOT NULL ,
        IdUsuario         char(9) NOT NULL ,
        fecha            char(10) NULL
    )go
    ALTER TABLE Sesion
        ADD CONSTRAINT XPKSesion PRIMARY KEY CLUSTERED (IdSesion ASC,IdUsuario ASC,IdGrupo
    ASC)go
    CREATE TABLE UploadFiles(
        clave            int IDENTITY (1000,1) ,
        nombre           varchar(255) NULL ,
        ruta            varchar(255) NULL ,
        Mimetype        varchar(255) NULL ,
        comentario      text NULL ,
        fecha           datetime NULL ,
        usuario         varchar(25) NULL ,
        IdGrupo         int NOT NULL
    )go
    ALTER TABLE UploadFiles
        ADD CONSTRAINT XPKUploadFiles PRIMARY KEY CLUSTERED (clave ASC)
go
    CREATE TABLE Usuarios(
        IdUsuario       char(9) NOT NULL ,
        pswd            varchar(40) NOT NULL ,
        nombre          varchar(25) NOT NULL ,
        appat           varchar(25) NOT NULL ,
        apmat           varchar(25) NOT NULL ,
        facultad        varchar(50) NULL ,
        carrera         varchar(50) NULL ,
        campus          varchar(25) NULL ,
        email           varchar(35) NULL ,
        fechan          char(10) NULL ,
        nusuario        varchar(25) NOT NULL ,
        comentario      text NULL ,
        genero          varchar(6) NULL ,
        imagen          varchar(255) NULL ,
        estado          varchar(5) NULL
    )go
    ALTER TABLE Usuarios
        ADD CONSTRAINT XPKUsuarios PRIMARY KEY CLUSTERED (IdUsuario ASC)
go
    ALTER TABLE Blog
        ADD CONSTRAINT R_102 FOREIGN KEY (IdUsuario) REFERENCES Usuarios(IdUsuario)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go
    ALTER TABLE Contactos
        ADD CONSTRAINT R_100 FOREIGN KEY (IdUsuario) REFERENCES Usuarios(IdUsuario)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go

```

```

ALTER TABLE Foro
    ADD CONSTRAINT R_93 FOREIGN KEY (IdGrupo) REFERENCES Grupo(IdGrupo)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION

go
ALTER TABLE ImagenesBlog
    ADD CONSTRAINT R_103 FOREIGN KEY (Idblog) REFERENCES Blog(Idblog)
        ON DELETE CASCADE
        ON UPDATE CASCADE

go
ALTER TABLE MBInvitado
    ADD CONSTRAINT R_106 FOREIGN KEY (Idblog) REFERENCES Blog(Idblog)
        ON DELETE CASCADE
        ON UPDATE CASCADE

go
ALTER TABLE MensajesBlog
    ADD CONSTRAINT R_104 FOREIGN KEY (Idblog) REFERENCES Blog(Idblog)
        ON DELETE CASCADE
        ON UPDATE CASCADE

go
ALTER TABLE MensajesGrupo
    ADD CONSTRAINT R_96 FOREIGN KEY (IdGrupo) REFERENCES Grupo(IdGrupo)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION

go
ALTER TABLE MensajesUsuarios
    ADD CONSTRAINT R_101 FOREIGN KEY (IdUsuario) REFERENCES Usuarios(IdUsuario)
        ON DELETE CASCADE
        ON UPDATE CASCADE

go
ALTER TABLE Preguntas
    ADD CONSTRAINT R_98 FOREIGN KEY (Idforo) REFERENCES Foro(Idforo)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION

go
ALTER TABLE Respuestas
    ADD CONSTRAINT R_97 FOREIGN KEY (Idpregunta) REFERENCES Preguntas(Idpregunta)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION

go
ALTER TABLE Sesion
    ADD CONSTRAINT R_71 FOREIGN KEY (IdGrupo) REFERENCES Grupo(IdGrupo)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION

go
ALTER TABLE Sesion
    ADD CONSTRAINT R_73 FOREIGN KEY (IdUsuario) REFERENCES Usuarios(IdUsuario)
        ON DELETE CASCADE
        ON UPDATE CASCADE

go
ALTER TABLE UploadFiles
    ADD CONSTRAINT R_107 FOREIGN KEY (IdGrupo) REFERENCES Grupo(IdGrupo)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION

go

```

REFERENCIAS

- [AL 2008] AL Zabir Omar. "Building a Web 2.0 Portal with ASP.NET 3.5", O'REILLY, 2008.
- [Albrecht 1999] J. Albrecht. Geospatial information standards a comparative study of approaches in the standardisation of geospatial information. *Computers & Geosciences*, 25:9-24, 1999.
- [Amiano 2006] Amiano Mitch, D'Cruz Conrad, Ethier Kay, Thomas D. Michael. "XML Problem – Design – Solution", Wiley Publishing Inc., 2006.
- [Archer 2001] Archer Tom, "A fondo C#". McGraw-Hill Profesional, Microsoft, 2001.
- [Bean 2010] Bean James. "SOA and Web Services Interface Design", Morgan Kauffman Publishers, 2010.
- [BSCW 2010] "BSCW", 2010. Disponible en: <http://public.bscw.de/>
- [BSCW-w 2010] "BSCW", 2010. Disponible en: <http://es.wikipedia.org/wiki/BSCW>
- [Chat 2005] "Chat", 2005. Disponible en: <http://www.monografias.com/trabajos/chat/chat.shtml>
- [Coad 1991] Coad P., Yourdon E. "Object-Oriented Analysis, 2da. Edición", Prentice-Hall, 1991.
- [Crespo 2006] Crespo K. "Los weblogs: un desafío para docentes innovadores". Educar, Educación y TIC Diciembre 2006. Disponible en: <http://weblog.educ.ar/educacion-tics/archives/008551.php>
- [Date 1993] Date C.J. "Introducción a los Sistemas de Bases de Datos", Addison Wesley Iberoamericana, 1993.
- [Delaney 2001] Delaney Kalen. "A fondo Microsoft SQL Server 2000", McGraw-Hill Profesional / Interamericana de España, 2001.
- [Dikes 2005] Dikes Lucinda, Tittel Ed. "XML for dummies 4th edition", Wiley Publishing Inc., 2005.
- [DOM 2000] "Document Object Model (DOM) Level 2 Events Specification". Disponible en: <http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/>

- [Ellis 1991] Ellis CA., Guibs S.J., Rein G.L. "Groupware: Some Issues and Experiences", Communication of the ACM Vol. 34, No. 1 (January 1991).
- [Eguiluz 2009] Eguiluz Perez Javier. "Introducción a JavaScript", Creative Commons, 2009. Disponible en: <http://www.librosweb.es/javascript>
- [Ferguson 2003] Ferguson Jeff, Patterson Brian, Beres Jason, Boutquin Pierre, Gupta Meeta. "La Biblia de C#", Ediciones ANAYA Multimedia, 2003.
- [Foggon 2006] Foggon Damien, "ASP.NET 2.0 Databases: from novice to profesional", Apress, 2006.
- [Foro 2010] Foro, 2010. Disponible en: [http://es.wikipedia.org/wiki/Foro_\(Internet\)](http://es.wikipedia.org/wiki/Foro_(Internet))
- [Goodchild 1977] M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. Kottman. *Interoperating Geographical Information Systems*. Kluwer, 1997.
- [Goodman 2001] Goodman Danny. "JavaScript Bible, Gold Edition", Hungry Minds, 2001.
- [Hall 2010] Hall Mike. "The DOM Event Model", 2010. Disponible en: <http://www.brainjar.com/dhtml/events/>
- [Harder 1999] T. Harder, G. Sauter, and J. Thomas. The intrinsic problems of structural heterogeneity and an approach to their solution. *The VLDB Journal*, 8(1):25-43, 1999.
- [Hart 2006] Hart Chris, Kauffman John, Sussman David, Ullman David. "Beginning ASP.NET 2.0 with C#", Wiley Publishing Inc., 2006.
- [Jiménez 2009] Jiménez Jesús. "Arquitecturas Orientadas a Servicios (SOA), metodología, normalización y calidad del software", Universidad de Murcia España, 2009. Disponible en: http://mnscs.atiica.um.es/soa/ManualDelCurso_2009.pdf
- [Martínez 05] Martínez Carreras Ma. Antonia. "Aplicación a la Plataforma de Aprendizaje" Tesis Doctoral. Departamento de la ingeniería de la Información y las comunicaciones. Universidad de Murcia. 2005.
- [Northrup 2006] Northrup Tony, Wildermuth Shawn, Ryan Bill of GrandMasters. "Microsoft .NET Framework 2.0 Application Development Foundation, self-paced Training Kit", Microsoft Press, 2006.
- [Rumbaugh 2000] Rumbaugh James, Jacobson Ivar, Booch Grady. "El lenguaje unificado de modelado, manual de referencia", Addison Wesley, 2000.

- [Pressman 2002] Pressman Roger S. “Ingeniería del software, un enfoque practico”, Quinta Edición. McGraw-Hill/Interamericana de España, 2002.
- [San Martin 2007] San Martin Calera Marco A. “Curso de AJAX y DOM”, 2007.
- [Schmuller 2002] Schmuller Joseph. “Aprendiendo UML en 24 horas”, Prentice-Hall, 2002.
- [Silberschatz 2002] Silberschatz Abraham, Bell Laboratories. Korth Henry F., Bell Laboratories. Sudarshan S., Instituto Indio de Tecnología, Bombay. “Fundamentos de Bases de Datos, Cuarta edición”, McGraw-Hill/Interamericana de España, 2002.
- [Snell 2009] Snell Mike, Johnson Glen, Northrup Tony, and GrandMasters. “Microsoft .NET FrameWork 3.5-ASP.NET Application Development, self-paced Training Kit”, Microsoft Press, 2009.
- [Turtschi 2002] Turtschi Adrian, Werry Jason, Hack Greg, Albahari Joseph, Nandu Saurabh, Meng Wei Lee. “C# .NET Web Developer’s Guide”, Syngress Publishing Inc., 2002.
- [UCOL 2010] UCOL. Universidad Autónoma de colima, 2010 disponible en: <http://www.ucol.mx/acerca/coordinaciones/cgv/dgee/#info>
- [UDDI 2010] “UDDI”, 2010. Disponible en: http://www.w3schools.com/WSDL/wsdl_uddi.asp
- [Videoconf 2005] “videoconf”, 2005. Disponible en: <http://www.monografias.com/trabajos/videoconferencia/videoconferencia.shtml>
- [Villa 2003] Villa Fermi, “Manual práctico de JavaScript”, 2003.
- [WSDL 2001] “WSDL Web Service Description Language 1.1”, 2003. Disponible en: <http://www.w3.org/TR/wsdl>
- [WS-I 2004] “Web Services Interoperability”, 2004. Disponible en: <http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html>
- [XML 1999] “XML Diez puntos”, 1999. Disponible en: <http://www.w3c.org/XML/1999/XML-in-10-points.es.html>
- [XML 2005] “XML Guía breve de tecnologías”, 2005. Disponible en: <http://www.w3c.es/divulgacion/guiasbreves/tecnologiasxml>

Índice de figuras

Figura 2.1	Arquitectura orientada a servicios básica.....	10
Figura 2.2	Proceso de publicación y obtención de servicios mediante UDDI (UBR).....	21
Figura 2.3	Símbolos usados en la notación E-R.....	30
Figura 2.4	Diagrama E-R correspondiente a clientes y prestamos.....	31
Figura 2.5	La tabla préstamo.....	33
Figura 2.6	La relación cuenta.....	34
Figura 2.7	Esquema General del .NET Framework.....	50
Figura 2.8	Árbol de nodos generado automáticamente por DOM a partir del código XHTML de la página.....	60
Figura 2.9	Nodos generados automáticamente por DOM para una etiqueta XHTML sencilla.....	60
Figura 2.10	Nodos generados automáticamente por DOM para una etiqueta XHTML con otras etiquetas XHTML en su interior.....	60
Figura 2.11	Modelo lineal secuencial.....	64
Figura 2.12	Modelo de construcción por prototipos.....	65
Figura 2.13	Modelo espiral típico.....	65
Figura 2.14	Modelo fuente.....	67
Figura 2.15	Pirámide del diseño OO.....	72
Figura 2.16	Transformación de un modelo de análisis OO a un modelo de diseño OO.....	73
Figura 2.17	Ejemplo de un documento de casos de uso.....	74
Figura 2.18	Ejemplo de un diagrama de casos de uso.....	75
Figura 2.19	Forma general de un diagrama de clases.....	75
Figura 2.19a	Representación general de la asociación entre dos clases.....	75
Figura 2.19b	Representación general de la multiplicidad entre dos clases.....	76
Figura 2.19c	Representación general de una asociación calificada.....	76
Figura 2.19d	Forma general de la generalización.....	76
Figura 2.19e	Forma general de la dependencia entre clases.....	76
Figura 2.20	Ejemplo de un diagrama de objetos.....	76
Figura 2.21	Ejemplo de diagrama de secuencia.....	77
Figura 2.22	Ejemplo de diagrama de colaboración.....	78
Figura 2.23	Ejemplo de diagrama de estado.....	78
Figura 2.24	Adición de detalles al icono de estado.....	78
Figura 2.25	Ejemplo de diagrama de actividades.....	79
Figura 2.26	Ejemplo de diagrama de componentes.....	79
Figura 3.1	Documento de caso de uso del esquema de las herramientas colaborativas.....	83
Figura 3.1a	Diagrama de caso de uso de las “Herramientas Colaborativas de SisAAC”.....	83
Figura 3.2	Documento de caso de uso del foro.....	84
Figura 3.2a	Diagrama de caso de uso “Foro”.....	84
Figura 3.3	Documento de caso de uso del espacio para compartir archivos.....	85

Figura 3.3a	Diagrama de caso de uso “Espacio para compartir Archivos”.....	85
Figura 3.4	Documento de caso de uso del blog.....	86
Figura 3.4a	Diagrama de caso de uso “Blog”.....	86
Figura 3.5	Documento de caso de uso del buzón de mensajes.....	87
Figura 3.5a	Diagrama de caso de uso “Buzón de mensajes”.....	87
Figura 3.6	Diagrama de clases de las herramientas colaborativas de SisAAC.....	90
Figura 3.7	Diagrama de secuencia del blog.....	91
Figura 3.8	Diagrama de secuencia del foro.....	91
Figura 3.9	Diagrama de secuencia del buzón de mensajes.....	92
Figura 3.10	Diagrama de secuencia del espacio para compartir archivos.....	92
Figura 3.11	Diagrama Entidad-Relación de la base de datos SisAAC.....	97
Figura 3.12	Modelo Relacional.....	97
Figura 4.1	Interfaz de un foro de discusión.....	104
Figura 4.2	Interfaz para enviar un mensaje.....	107
Figura 4.3	Interfaz para subir o bajar un archivo.....	109
Figura 4.4	Interfaz del Blog personal.....	111
Figura A.1	Pantalla que muestra el formulario para crear un nuevo foro, un ListBox de los foros existentes y el hipervínculo del espacio para compartir archivos.....	113
Figura A.2	formulario para crear un nuevo foro.....	114
Figura A.3	Elementos de la pantalla de un foro creado por un usuario.....	114
Figura A.4	Pantalla que muestra el mensaje de confirmación de una pregunta enviada al foro.....	115
Figura A.5	Formato de una pregunta enviada a un foro.....	115
Figura A.6	Pantalla que muestra el cuadro de texto para enviar un mensaje y un mensaje enviado.....	115
Figura A.7	Formato de una respuesta y los botones de “Regresar”.....	116
Figura A.8	Pantalla del espacio para compartir archivos.....	116
Figura A.9	Formulario para subir un archivo.....	116
Figura A.10	Búsqueda de un archivo en la computadora.....	117
Figura A.11	Mensaje de éxito y archivo mostrado en el área de descargas.....	117
Figura A.12	Formulario “Área de descargas” que muestra un archivo seleccionado y sus detalles.....	118
Figura A.13	Mensaje de confirmación para bajar un archivo.....	118
Figura A.14	Ventana para dar una ruta donde se guarde el archivo.....	118
Figura A.15	Barra de navegación.....	119
Figura A.16.	Vínculo para ingresar a la interfaz de eliminación de archivos.....	119
Figura A.17	Interfaz para eliminar un archivo.....	119
Figura A.18	Vínculos del recuadro “Mis mensajes”.....	120
Figura A.19	Notificación del sistema cuando no existen mensajes.....	120
Figura A.20	Lista de los mensajes existentes en la bandeja de entrada.....	120
Figura A.21	Contenido de un mensaje.....	120
Figura A.22	Interfaz para enviar un mensaje.....	121
Figura A.23	Formato debidamente llenado.....	121
Figura A.24	Interfaz del blog de un usuario.....	122

Figura A.25	Encabezado.....	122
Figura A.26	Barra lateral izquierda.....	122
Figura A.27	Barra central.....	123
Figura A.28	Barra lateral derecha.....	123
Figura A.29	Vinculo para agregar una foto o un comentario al blog.....	123
Figura A.30	Interfaz para agregar una foto o un comentario al blog.....	124
Figura A.31	Vinculo “Contactos”.....	124
Figura A.32	Interfaz para agregar o eliminar un contacto.....	125