INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA
DE INGENIERÍA Y CIENCIAS SOCIALES
Y ADMINISTRATIVAS

"PROYECTO DENOMINADO FRAMEWORK 3.5

ASP.NET."

INFORME DE PRÁCTICA PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA INFORMÁTICA

P R E S E N T A:

M A R C O A N T O N I O M I R A N D A A R A I Z A

MÉXICO, D.F. 2010

Índice

Resumen

Este proyecto trata acerca de la actualización del framework de la herramienta BAX (Business Application Express Developer)a la versión 3.5 de ASP.NET.

Para el desarrollo de este proyecto fue necesario el estudio y comprensión del modelo de desarrollo en el que se basa la herramienta BAX para la construcción de aplicaciones . Dicho modelo es un concepto propietario de la empresa Fractanet S.A. de C.V. llamado MOOSS (Método Orientado a Objetos para la Solución de Sistemas).

Fue necesario aprender el manejo de la versión actual del framework el cual trabajaba sobre la versión 1.1 de ASP.NET.

Se hizo un estudio de las nuevas tendencias tecnológicas tanto en el desarrollo en .NET como JavaScript para definir la estrategia y nuevo modelo de generación de aplicaciones WEB.

Este nuevo modelo de generación permitirá desarrollar aplicaciones más poderosas, eficientes, rápidas con una gran calidad de presentación visual la cual será una ventaja competitiva para Fractanet en el mercado.

Introducción

Para iniciar daré información acerca de la empresa Fractanet S.A de C.V. en donde se llevó a cabo el proyecto. 

El área de negocio es: desarrollo de aplicaciones WEB para diferentes plataformas.

Tecnología y ventaja competitiva: se cuenta con tecnología propia que permite hacer prototipos que aseguran el concepto y desarrollos en tiempo real, de alta calidad y a costos atractivos.

Característica primordial: los proyectos se concluyen en los tiempos, con los costos y con los alcances esperados. Las desviaciones necesarias en los alcances, son manejables y no vuelven críticos los proyectos gracias a la flexibilidad de nuestra tecnología.

Principales logros: exportar aplicaciones a otros países de Latinoamérica y Canadá.

Fecha de inicio : 1998

Principales Clientes: Avaya Communications México, EDS de México, Thomson, Afore Profuturo GNP, CONAGUA, etc.

Propiedad intelectual: metodología de diseño y desarrollo de aplicaciones.

Sectores atendidos:

- Productivo.
- Financiero.
- Comunicaciones.
- Servicios.
- Comercio Exterior.
- Etc.

Productos y servicios:

- Software a la medida.
- Consultoría en el diseño de procesos de negocios.
- Desarrollo de aplicaciones para empaquetar.
- Soporte al diseño de aplicaciones.

Página de internet: www.fractanet.com.mx

Actualmente las oficinas principales se encuentran en el Cluster de IT Ciudad de México

Es un programa de Fomento para el Desarrollo de la Industria del Software en el Distrito Federal, establecido por el Gobierno del DF y la Secretaría de Economía junto con un grupo de empresas e instituciones afines al sector.

Prosoftware es la asociación civil fundada para integrar a las empresas participantes en el proyecto inmobiliario de la primera fase del programa DsoFtware.

Entre sus objetivos principales están:

- Apoyar las actividades de los asociados,
- Facilitar el acceso a programas gubernamentales y privados
- Articular a las empresas con asociaciones y otros actores.

Se encuentra ubicada en: Poniente 140 #839-9, colonia Industrial Vallejo.

Entrando en materia, lo que ha buscado la empresa durante muchos años y se ha vuelto una de las prioridades principales es buscar optimizar el **desarrollo de sistemas**.

Dentro de la búsqueda de esa optimización para el desarrollo, se ha identificado que independientemente de cualquier metodología que se use, teoría que se aplique ó procedimientos que se sigan siempre habrá el problema al codificar las cosas, puesto que es la parte en donde se tienen varios factores como:

- Velocidad con que una persona teclea.
- La cantidad de errores que el programador tiene por errores de sintaxis.
- Cuando por una razón otro programador tiene que seguir con el código de otro programador, si este código no está bien comentado o bien documentado, es muy lento que la otra persona entienda bien lo que se está desarrollando.
- Que el programador no entienda a la perfección lo que se está pidiendo desarrollar y programe cosas diferentes, lo que implica que se tuvo que codificar una cosa innecesaria o inservible y se llevó tiempo y así mismo el tiempo para recodificar y desarrollar lo que se pide.
- Etc.

Por eso es que hemos llegado a la conclusión, de que la codificación es el cuello de botella dentro del desarrollo.

Dentro de la empresa se han dedicado por once años a este problema, analizarlo, sacar varias propuestas y obtener una solución que optimice el desarrollo.

Para comenzar se ha acotado el problema para abarcar únicamente **sistemas de información**, dejando fuera otro tipo de software como el software de control de hardware, juegos, simulaciones, etc., bajo la premisa de que los sistemas de información es el negocio más rentable al que se pudieron dedicar.

Para tal fin se parte de la premisa de que todos los sistemas de información son muy similares unos a otros y bajo esta afirmación se han dedicado a analizar la problemática completa para sistematizar la construcción de sistemas desde el análisis hasta la liberación y en fechas recientes la administración del proyecto.

Como por ejemplo en todos los sistemas se necesita un modulo de seguridad, uno de entrada al sistema (login), altas, bajas, consultas de información, un modulo de administración de proyectos, etc.

En este trabajo se tratará únicamente la parte del **framework** que se utiliza como piedra angular de todo desarrollo bajo la premisa de que la codificación es la parte más difícil, artesanal, propensa a errores, costosa y tardada de todo el ciclo sea cual sea el modelo de desarrollo que se elija.

Nuestro objetivo es que esto no sea así.

Como ya se ha dicho, la empresa lleva más de once años investigando la manera de hacer eficiente y rentable el proceso de desarrollo, en esos años se ha ensayado con diferentes tecnologías y arquitecturas. Se comenzó cuando la arquitectura **Cliente / Servidor** mandaba, las computadoras tenían muy poca capacidad de proceso y almacenamiento e importaba mucho emplear técnicas complejas de programación para aprovechar esos escasos recursos. Como referencia, la arquitectura C/S era compleja, difícil de implementar y costosa partiendo de que las redes eran inmaduras (complejas, difíciles y costosas), pensar en algo como Internet o incluso una Intranet era simplemente imposible.

Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes.

Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD).

Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor.

Ambas partes deben estar conectadas entre sí mediante una red. Una representación gráfica de este tipo de arquitectura sería la siguiente.

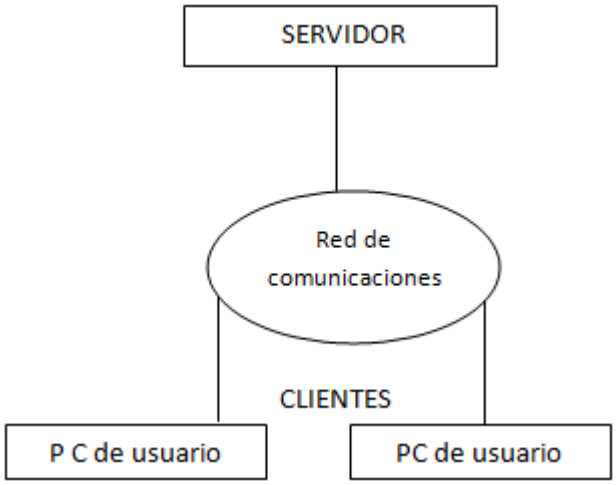

Figura 1.- Representa las peticiones de los usuarios hacia un servidor

Este tipo de arquitectura es la más utilizada en la actualidad, debido a que es la más avanzada y la que mejor ha evolucionado en estos últimos años.

Podemos decir que esta arquitectura necesita tres tipos de software para su correcto funcionamiento:

- Software de gestión de datos: este software se encarga de la manipulación y gestión de los datos almacenados y requeridos por las diferentes aplicaciones. Normalmente este software se aloja en el servidor.

- Software de desarrollo: este tipo de software se aloja en los clientes y solo en aquellos que se dedique al desarrollo de aplicaciones.

- Software de interacción con los usuarios: también reside en los clientes y es la aplicación gráfica de usuario para la manipulación de datos, siempre claro a nivel usuario (consultas principalmente).

A parte de estos existen más aplicaciones software para el correcto funcionamiento de esta arquitectura pero ya están condicionados por el tipo de sistema operativo instalado, el tipo de red en la que se encuentra, etc.

Actualmente todo eso ha cambiado, las máquinas portátiles tienen más memoria de lo que tuvieron los servidores de hace 10 años, la computación de 64 bits es una realidad, el performance ha seguido casi religiosamente la ley de Moore y las redes es algo que prácticamente se saca de la caja y se usa y de Internet ya ni hablamos.

La **Ley de Moore** expresa que aproximadamente cada 18 meses se duplica el número de transistores en un circuito integrado. Se trata de una **ley empírica**, formulada por el cofundador de Intel, Gordon E. Moore el 19 de abril de 1965. No es una ley que se cumpla sí o sí, porque es una constatación, algo práctico sin una teoría definida pero lo cierto es que aproximadamente cada dos años se duplica el número de transistores que hay en un chip y viene ocurriendo desde hace 40 años aunque es posible que deje de cumplirse dentro de 15.

En 1965, el ingeniero Gordon Moore afirmó que el número de transistores por pulgada en circuitos integrados se duplicaba cada año y que la tendencia continuaría durante las siguientes dos décadas.

Más tarde, en 1975, modificó su propia ley al afirmar que el ritmo bajaría, y que la capacidad de integración se duplicaría aproximadamente cada 24 meses. Esta progresión de crecimiento exponencial, duplicar la capacidad de los circuitos integrados cada dos años, es lo que se considera la Ley de Moore. Sin embargo, el propio Moore puso en el año 2007 fecha de caducidad a su ley: "*Mi ley dejará de cumplirse dentro de 10 o 15 años*". Según aseguró durante la conferencia en la que hizo su predicción afirmó, no obstante, que una nueva tecnología vendrá a suplir a la actual ya que con las actuales tecnologías sería difícil reducir los chips por debajo de los 12 nanómetros de superficie.

Figura 2.- Representación de de la Ley de Moore; numero de transistores/año.

La **consecuencia directa de la Ley de Moore** es que los precios bajan al mismo tiempo que las prestaciones de los ordenadores suben: la computadora que hoy vale 3000 dólares costará la mitad al año siguiente y estará obsoleta en dos años. En 26 años el número de transistores en un chip se ha incrementado 3200 veces.

Actualmente se aplica a ordenadores personales. Sin embargo, cuando se formuló no existían los procesadores, inventados en 1971, ni los ordenadores personales, popularizados en los años 1980, es una constatación en el mundo de la **microelectrónica**.

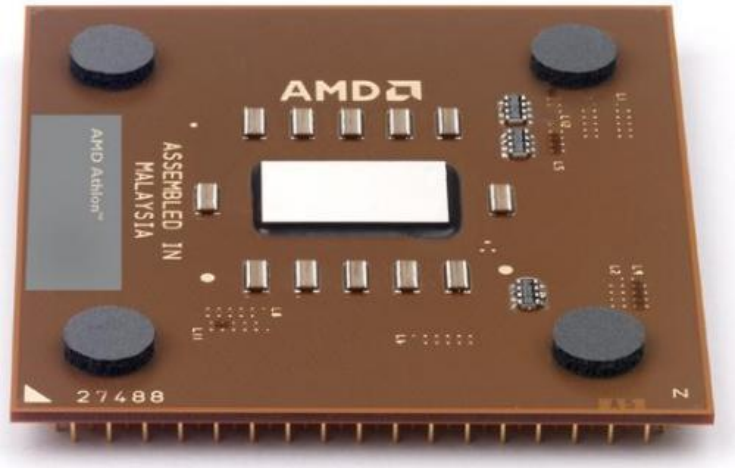

Figura 3.- Ejemplo de procesador

Cada dos años el tamaño de un chip con determinado número de transistores se reduce a la mitad, que en la misma superficie de silicio cabe el doble de transistores pasa a costar la mitad porque el coste es proporcional a la superficie. Una vez desarrollados los chips de tamaño diminuto, se tardan un par de años en poner a punto los equipos, un año y poco después se mejora la producción y se comercializan los nuevos equipos, muchísimos más baratos y mucho más potentes a la vez.

En el año 2006, se introdujo una tecnología de fabricación de semiconductores de 64 nanómetros, en 2008, la de 45 nanómetros, y a finales de 2009 serán los chips de 32 nanómetros. Si la tendencia continúa y la ley de Moore se sigue aplicando, llegará un límite porque en el año 2014 tendrían que desarrollarse procesadores de 16 nanómetros, pero si la cosa sigue avanzando sería muy difícil bajar aún más.

En nuestra empresa se ha decidido que en este panorama las **aplicaciones web** llegaron para quedarse desplazando, al menos en lo que se refiere a sistemas de información, a las aplicaciones de escritorio tradicionales.

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

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como los web mails, wikis, tiendas en línea, etc. son ejemplos bien conocidos de aplicaciones web.

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

Las interfaces web tienen ciertas limitaciones en las funcionalidades que se ofrecen al usuario. Hay funcionalidades comunes en las aplicaciones de escritorio como dibujar en la pantalla o arrastrar-y-soltar que no están soportadas por las tecnologías web estándar.

Los desarrolladores web generalmente utilizan lenguajes interpretados o script en el lado del cliente para añadir más funcionalidades, especialmente para ofrecer una experiencia interactiva que no requiera recargar la página cada vez (lo que suele resultar molesto a los usuarios).

Recientemente se han desarrollado tecnologías para coordinar estos lenguajes con tecnologías en el lado del servidor..

Como ejemplo, AJAX, es una técnica de desarrollo web que usa una combinación de varias tecnologías.

Una ventaja significativa es que las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación web se escribe una vez y se ejecuta igual en todas partes. Sin embargo, hay aplicaciones inconsistentes escritas con HTML, CSS, DOM y otras especificaciones para navegadores web que pueden causar problemas en el desarrollo y soporte de las aplicaciones web. Adicionalmente, la posibilidad de los usuarios de personalizar muchas de las características de la interfaz (tamaño y color de fuentes, tipos de fuentes, inhabilitar Java Script) puede interferir con la consistencia de la aplicación web.

En el 2002 se comenzó a cambiar nuestros esquemas de desarrollo para desarrollar exclusivamente aplicaciones web y en 2004 se implantaron las herramientas con que actualmente se trabajan.

Estas herramientas están implementadas en **ASP.NET 1.1** programadas con **C#**. Actualmente hay 9 sistemas desarrollados y operando en ésta plataforma y otros 4 que se desarrollaron, se usaron pero dejaron de usarse por cambios en políticas del negocio, en total 13 aplicaciones desarrolladas más 5 ó 6 prototipos que nunca progresaron y otras tantas aplicaciones para capacitación interna.

Sin embargo, esta plataforma, aunque todavía es útil y productiva, ya está quedando obsoleta rápidamente. Por ejemplo, no soporta 64 bits, Microsoft ya ha retirado el soporte oficial y ya no hay mejoras. Así que se ha decidido actualizar las herramientas para soportar **ASP.NET 3.5** y superior (está próximo a liberarse la 4.0) y de paso revisar la arquitectura de las aplicaciones para aprovechar lo que la empresa ha aprendido en estos 5 años de desarrollo web y las tecnologías que han surgido.

La arquitectura del framework actual, está muy limitada:

1. Se basa en **componentes visuales** que dependen 100% de la maquinaria de componentes de .NET.
2. Únicamente trabaja con Internet Explorer.

3. El comportamiento de la interfaz de usuario debe programarse en **JavaScript**.
4. La programación de ciertas características se ha vuelto muy compleja.
5. Trabaja con **AJAX** pero no está optimizada.
6. Es complicado integrar componentes ó características nuevas.

Por todas estas razones y más, se tomo la decisión de llevar a cabo la actualización del framework, se ha revisado la arquitectura completamente y casi reescrito al 100% el framework para buscar las siguientes características:

1. Independencia total entre la tecnología de implementación en el Client Side y la del Server Side.
2. Generar aplicaciones que puedan ejecutarse en Internet Explorer, FireFox, Safari y Chrome. (Cross Browser)
3. Facilidad de implementación de los problemas más comunes que se nos presentan.
4. Uso de AJAX al 100% y optimizado.
5. Uso de las características más nuevas de los DBMS.
6. Facilidad para integrar características nuevas.
7. Facilidad para cambiar al diseño gráfico de las aplicaciones.
8. Facilidad para programar y centralizar las reglas de negocio.
9. Facilitar la detección de errores.
10. Facilidad para cambiar las etiquetas de la IU.

Todo este trabajo no es sencillo, es muy extenso y requiere de muchas capas de revisión y perfección; por lo que este proyecto no pudo ser completado en los 4 meses que se había planteado inicialmente pero se estima concluir en 2 ó 3 meses a más tardar. Sin embargo se realizo un avance muy significativo que da la pauta para seguir y terminar con todo este proceso de reestructuración del framework, para poderlo integrar a nuestra herramienta y dar pie a la innovación tecnológica que se tiene planeada a corto plazo para la empresa.

Este reporte muestra los antecedentes teóricos en que se basa la construcción del framework y los resultados que se han obtenido a partir de ellos. Después de un tiempo significativo de investigación, análisis y aprendizaje de nuevas tecnologías, con el fin de ser siempre la punta del iceberg en innovación tecnológica, estar siempre a la vanguardia para así lograr estar por encima de las expectativas de nuestros clientes y mantenernos dentro de la competencia de nuestro mercado, ofreciendo siempre la mejor alternativa en soluciones de tecnologías de información.

Antecedentes

Sistema

En el sentido más amplio, un sistema es simplemente un conjunto de componentes que interactúan para alcanzar un objetivo.

Un negocio es un sistema. Sus partes tienen nombres tales como: mercadotecnia, producción, ventas, investigación y personal. Estos componentes trabajan todos juntos para crear una utilidad que beneficia a alguien. Ahora bien, cada una de estas partes es un sistema en sí mismo. El departamento de contabilidad puede consistir en cuentas por cobrar por pagar, facturación, auditoria, etc. Cuando se comienza a ver lo abundante que son los sistemas no sorprende darse cuenta que cada sistema del negocio depende de uno o más entidades abstractas llamadas Sistemas de Información, por medio de estos sistemas los datos pasan de una persona o departamento a otro y puede realizarse cualquier cosa, desde comunicaciones entre oficinas telefónicamente hasta un sistema de computadoras que genere informaciones periódicas para diferentes usuarios.

Los sistemas de información de hecho sirven a todos los sistemas de un negocio.

Ellos son el lazo que mantiene unidos a diferentes componentes en forma total que pueden trabajar de manera efectiva hacia el mismo objetivo.

Los sistemas utilizan un modelo de control básico que consisten en:

- Un estándar para rendimiento aceptables
- Un método de medición de ese rendimiento real
- Una forma para comparar el rendimiento real contra el estándar
- Un método para retroalimentación

Los sistemas que pueden ajustar sus actividades a niveles aceptables continuaran funcionando, los que no pueden hacerlo se detienen.

El concepto de interacción dentro de un medio ambiente que caracteriza a los sistemas abiertos es esencial para el control por medio de la recepción de la entrada y la evaluación de la misma un sistema puede determinar que está bien operado.

Sistemas de información aplicado para los negocios.

Los sistemas de información son como cualquier otro sistema dentro de la empresa en cuanto tiene propósitos de interactuar con otros componentes de la compañía.

La tarea de los sistemas de información consiste en procesar la entrada, mantener archivos de datos en relación con la empresa y producir información informes y otras salidas.

Los sistemas de información están integrados por subsistemas que incluyen el hardware, software y almacenamiento de los datos para los archivos y bases de datos. Procedimientos específicos describen los sistemas utilizados.

El conjunto particular de subsistema, es decir, el equipo especifico, programa, archivos y procedimientos, comprenden una aplicación de sistema de información. Por lo tanto, los sistemas de información pueden tener aplicaciones de compras, contabilidad o ventas.

Dado que los sistemas de información dan apoyo a otros sistemas de la empresa, los analistas deben estudiar primero el sistema de la compañía como un todo y después los detalles de los sistemas de información

Con frecuencia el personal utiliza organigrama para describir las relaciones de los componentes de la empresa, como divisiones, departamentos, oficinas y personal. Aunque los organigramas pueden mostrar las relaciones formales entre los componentes con cierta exactitud, no dicen cómo opera el sistema de negocios, dado que muchos detalles de importancia no se ponen en las cajas del diagrama. Ejemplos de otros datos en relación con el sistema que son importantes para los analistas de sistemas incluye:

- Canales no formales: ¿qué interacciones existen entre el personal y los departamentos, pero que no aparecen en el organigrama o en los procedimientos de operación previamente establecidos?
- Interdependencias: ¿en qué otros departamentos y componentes de la empresa se encuentra una dependencia específica?
- El personal clave y las funciones: ¿cuáles individuos y elementos del sistema son más importantes para su existencia exitosa?
- Relaciones criticas de comunicaciones: ¿cómo circulas la información y las instrucciones entre los componentes de la empresa y cómo interactúan las diferentes arreas con las demás?

En las empresas, los analistas desarrollan dos tipos diferentes de sistemas de información: los sistemas de procesamiento de transacciones mejoran las actividades diarias de las cuales dependen la compañía. Las aplicaciones normales incluyen el proceso de datos contables, preparación de nominas y manejo de pedidos de ventas. Las actividades son de rutina, ocurren con frecuencia y en la misma forma.

Los sistemas de procesamientos de transacciones automatizadas postulan con su objetivo básico la eficiencia la velocidad y exactitud en el procesamiento de grandes cantidades de datos.

Los sistemas de decisión administrativos se utilizan para dar apoyo directo a los gerentes responsables de la toma de decisiones dentro de la empresa, aunque los sistemas de decisiones administrativas no les dicen a los gerentes como tomar las decisiones, les ayudan proporcionándoles información importante que servirán de entrada al proceso de decisiones.

El análisis de sistema es el conocimiento de situaciones y no la solución de los problemas. Los buenos analistas, por lo tanto, hacen hincapié en la investigación y las preguntas para aprender cómo opera un sistema actualmente y para identificar los requerimientos que los usuarios tienen para uno nuevo o modificarlo. Solo después de que los analistas han entendido por completo el sistema, son capaces de analizarlo y conjuntar las recomendaciones para el diseño de sistema

Tipos de sistemas de información.

Según la función a la que vayan destinados o el tipo de usuario final del mismo, los SI pueden clasificarse en:

- Sistema de procesamiento de transacciones (TPS).- Gestiona la información referente a las transacciones producidas en una empresa u organización.
- Sistemas de información gerencial (MIS).- Orientados a solucionar problemas empresariales en general.
- Sistemas de soporte a decisiones (DSS).- Herramienta para realizar el análisis de las diferentes variables de negocio con la finalidad de apoyar el proceso de toma de decisiones.
- Sistemas de información ejecutiva (EIS).- Herramienta orientada a usuarios de nivel gerencial, que permite monitorizar el estado de las variables de un área o unidad de la empresa a partir de información interna y externa a la misma.
- Sistemas de automatización de oficinas (OAS).- Aplicaciones destinadas a ayudar al trabajo diario del administrativo de una empresa u organización.
- Sistema experto (SE).- Emulan el comportamiento de un experto en un dominio concreto.
- Sistema Planificación de Recursos (ERP).- Integran la información y los procesos de una organización en un solo sistema.

Importancias de los sistemas de información.

Las organizaciones siempre utilizaron sistemas que les permitieron administrar el manejo de su información, con lo cual no necesariamente debe existir una computadora para reconocer la existencia de estos tipos de sistemas pues estos pueden ser también del tipo manuales; por ejemplo una distribuidora pequeña que no tenga informatizada la totalidad de sus esquemas de logística y comercialización. Lo importante es que el sistema permita almacenar, recuperar, procesar y distribuir información.

Sin embargo, es cada vez más necesario el disponer de sistemas de información basados en computadoras por los beneficios que estos proporcionan: reducción de errores provocados por las personas a través del control de las entradas, velocidad en el procesamiento de datos, posibilidad de realizar tediosos análisis sobre los mismos, reducción de espacio físico destinado a su almacenamiento, agilidad al momento de buscar algún dato en particular, y otros tipos de ventajas que podrían lograrse en caso de enfocarse en el uso estratégico de los mismos.

En los entornos de negocio actuales, el disponer de una buena gestión en el uso de los sistemas de información se convierte en una estrategia que pueden utilizar las empresas para hacer frente a sus fuerzas competitivas.

Para valorar aún más la importancia del concepto tratado, en (BRIEN, 2006) se hace la pregunta **"¿Por qué es importante los sistemas de información y la tecnología de la información?"**, a lo que responde lo siguiente: **por qué es importante contabilidad, finanzas, administración de operaciones, mercadotecnia, administración de recursos humanos o cualquier otra función principal de negocios**. Creo que tal respuesta es más que suficiente para visualizar la importancia que en la actualidad se le otorga a los sistemas de información, pues, en palabras de los de autores de (BRIEN, 2006), "estos tipos de conocimientos se constituyen en un elemento vital de las organizaciones y negocios exitosos".

(BRIEN, 2006). O`Brien, James; Marakas, George. Sistemas de información gerencial. Editorial: Mc Graw Hill

Codificación

Dado que los proyectos de sistemas de información se diseñan teniendo en mente ahorro de espacio, costo y tiempo, los sistemas de codificación en los cuales las condiciones, palabras ideas o relaciones se expresan a través de un código, se desarrollan para reducir errores de control y acelerar el proceso en su totalidad. Un código es un número breve, titulo o símbolo utilizado en vez de descripciones más largas o ambiguas. Cuando ocurre un suceso los detalles del mismo se resumen a menudo mediante un código. Son necesarios menos detalles en la entrada y no se tienen como resultado ninguna pérdida de información. Los seis tipos de métodos de codificación presentados en esta sección son:

1.- Código Clasificación

2.- Código Función

3.- Código Tarjeta

4.- Código Secuencia

5.- Código Dimito significativo

6.- Código mnemotécnico.

Código Clasificación

Los Código Clasificación colocan entidades separadas, como hechos, personas y objetos en grupos distintos llamados clases. Un código se utiliza para distinguir una clase de otra. El usuario registra el código en un documento fuente o puede teclearlo directamente al sistema a través de la terminal en un sistema en línea. El usuario (Ya sea que haya aprendido los códigos o los consulte) clasifica el hecho en una o varias categorías posibles y registra el código.

Ejemplo: para confeccionar las nominas de la Empresa de Servicios Informáticos se requiere la categoría ocupacional de cada trabajador, ya que se debe informar al departamento de estadísticas el aporte que debe realizar la empresa a la ONAT por concepto de la seguridad social por categoría ocupacional, para ello se crea un clasificador de categoría ocupacional

Los Código Clasificación simplifican el proceso de entrada porque solo se requiere un código de un solo dígito. La necesidad de anotar descripciones largas o realizar juicios se elimina, en una palabra es más sencillo.

Código de funciona

Los codillos de función establecen las actividades o el trabajo que han de llevar a cabo sin tener que anotar todos los detalles en forma descriptiva. Los analistas utilizan este tipo de código en forma frecuente en los datos de transacción para señalar al sistema como procesar los mismos.

Ejemplo: el diseño para el proceso de archivo puede especificar lo siguiente.

- Adicionar los registros en una transacción por medio de una "A" de un "1" o de cualquier otro esquema de codificación que el analista seleccione.

- Borrarlos registros, el código de función puede ser "D" 0 "2"

- Modificar datos almacenados (cambiar o actualizar) el codillo de función puede ser "M" o "C" o "3".

El código de función en particular puede determinar el contenido del registro de entrada; por ejemplo, para que el sistema añada un registro, todos los elementos de datos del registro se deben incluir en los datos de entrada. Por otro lado, para borrar un registro solo se especifica el código de borrado y los números de identificación o llave del regidor que va a desaparecer. Si se

va a cambiar un campo en el registro, la entrada debe incluir el código del cambio, una indicación de que campo se va a modificar y los datos para ese campo.

Código de secuencia

Los códigos secuenciales son números o letras asignado en serie. Señalan el orden en el cual han ocurrido los hechos: por ejemplo un sistema bancario debe ser capaz de mantener un seguimiento del orden de las transacciones, de manera que sea claro que transacción procesar primero, cual después y así sucesivamente; por lo tanto se debe especificar el número de secuencia en el diseño con el propósito de ordenar las transacciones.

Los códigos secuenciales son utilizados también con el propósito de identificación, pero se asignan en el orden en el cual los clientes entran al sistema.

Los de secuencia asignados en un orden cuidadoso no permiten inserción de nuevos miembros entre los ya existentes, Los analistas especifican la asignación de códigos de secuencia a intervalos de 10 / 20, o algún otro rango a fin de permitir expansiones posteriores.

Código Dimito significativo

Un esquema de codificación bien concebido que utilice sub códigos dentro de códigos más grandes o números, puede proporcionar una buena información a los usuarios. Supóngase que los números de artículos se asignan a los diferentes materiales y productos que una compañía almacena o vende. Una forma es asignar números en secuencia, comenzando por el primero y continuando hasta el último, o pueden añadir un prefijo para los números de identificación, de manera que pueda describirse más adelante el tipo de artículo: acero tiene un prefijo "A", plástico uno "P".

Los códigos pueden dividirse en subconjuntos o su códigos, caracteres que son parte del número de identificación que tienen un significado especial. Los sub códigos indican al usuario información adicional sobre el artículo, Ejemplo La clase de producto, el proveedor, esta información se lleva a través de dígitos significativos.

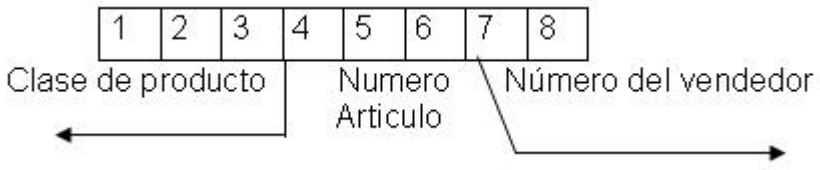

Figura 4.- Ejemplo de subconjuntos.

Código mnemotécnico.

Los **Código mnemotécnico** utilizan letras y símbolos del producto para describirlo en una forma que comunique visualmente. Por ejemplo; para describir un televisor a color de 21", un código útil es TV-CL-21", (en blanco y negro seria TV-BW-21"). Es difícil confundir el **mnemotécnico** TV con el de otro producto, las universidades siempre utilizan mnemotécnico para codificar a los estudiantes: Msc. Máster en ciencias, Dra. Doctores en ciencias.

La codificación de los datos y las transacciones reduce la cantidad de datos que se necesita introducir y simplifican el proceso, de manera que se reduce la probabilidad de errores. La selección de un código sobre otro depender` por su puesto, de los datos y de los objetivos del analista.

Existe forma para validar los datos y además modificarlos, son dos metidos: corrección automatiza de errores y dígitos de auto verificación en los campos de llave.

Corrección automática de entrada

A veces los analistas especifican que los programas se escriban para corregir errores en los datos. Este método de validación de la entrada se utiliza a fin de reducir el número de pasos de corrección de errores por separado o rechazado de las transacciones durante el procesamiento. Este modo requiere solo que el programa detecte un error y realice la corrección en forma automática, por ejemplo, el personal de captación de datos que esta tecleando los datos de un campo numérico de 6 dígitos puede introducir nada mas 3, además todo el campo debe contener número (un blanco no es un número) en vez de que el programa rechace la transacción debido a que faltan los ceros iniciales (los ceros en la posición sin utilizar al principio de los datos del campo), puede diseñarse para insertarlo automáticamente. Solo se insertan los ceros iniciales, no los de arrastre.

Otro punto de vista de codificación

La codificación es la fase más penada del trabajo de programación al tener que escribirse líneas de código entendibles por la computadora.

Y aunque si es bien cierto que hoy en día los entornos de desarrollo integrados (IDE) nos facilitan mucho esta tarea, también es cierto que la mayoría de ellos utilizan técnicas que a veces resultan muy costosas en cuanto al performance de lo que se está desarrollando y también es bien cierto que la mayoría de estos IDE´s escriben muchas líneas de código que muchas veces se podrían

resolver con menor cantidad, siguiendo el principio que mientras menos líneas de código tengo más fácil es su mantenimiento y entendimiento.

Revisiones de código y estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, establezca un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Aunque el propósito principal para llevar a cabo revisiones del código a lo largo de todo el desarrollo es localizar defectos en el mismo, las revisiones también pueden afianzar los estándares de codificación de manera uniforme. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo.

Técnicas de codificación

Las técnicas de codificación incorporan muchos aspectos del desarrollo del software. Aunque generalmente no afectan a la funcionalidad de la aplicación, sí contribuyen a una mejor compresión del código fuente. En esta fase se tienen en cuenta todos los tipos de código fuente, incluidos los lenguajes de programación, de secuencias de comandos, de marcado o de consulta.

Las técnicas de codificación aquí definidas no pretenden formar un conjunto inflexible de estándares de codificación. Más bien intentan servir de guía en el desarrollo de un estándar de codificación para un proyecto específico de software.

Las técnicas de codificación están divididas en tres secciones:

- Nombres
- Comentarios
- Formato

Nombres

El esquema de nombres es una de las ayudas más importantes para entender el flujo lógico de una aplicación. Un nombre debe más bien expresar el "qué" que el "cómo". Si evita nombres que se refieran a la implementación subyacente (sujeta a cambios), estará conservando un grado de abstracción que lo simplificará todo.

Por ejemplo, puede usar GetNextStudent() en vez de GetNextArrayElement().

El interés de poner un nombre es que la dificultad para escoger uno adecuado puede indicar que se necesita analizar o definir con mayor precisión el propósito de un elemento. Ponga nombres lo suficientemente largos para que sean elocuentes, pero lo bastante cortos como para que no pequen de palabrería. Desde el punto de vista de la programación, un nombre único sirve solamente para diferenciar un elemento de otro. Los nombres expresivos funcionan como ayuda para el lector, por eso, es lógico dar nombres que sean fáciles de comprender. No obstante, asegúrese de que los nombres escogidos sean compatibles con las reglas de cada lenguaje y con los estándares.

Los siguientes puntos son técnicas de nomenclatura recomendadas.

Rutinas

- Evite nombres imprecisos que permitan interpretaciones subjetivas, como por ejemplo AnalyzeThis() para una rutina, o bien xxK8 para una variable. Tales nombres contribuyen más a la ambigüedad que a la abstracción.

- En lenguajes orientados a objetos es redundante incluir nombres de clases en el nombre de las propiedades de clases, como por ejemplo Book.BookTitle. En su lugar, utilice Book.Title.

- Use el método verbo-sustantivo para dar nombre a las rutinas que ejecuten alguna operación en un determinado objeto, como por ejemplo CalculateInvoiceTotal().

- En lenguajes que permitan sobrecarga de funciones, todas las sobrecargas deberían llevar a cabo una función similar. Para los lenguajes que no permitan la sobrecarga de funciones, establezca una nomenclatura estándar relacionada con funciones similares.

Variables

- Añada calificadores de computación (Avg, Sum, Min, Max, Index) después de un nombre de variable donde le resulte apropiado.

- Utilice pares complementarios en nombres de variables, como min/max, begin/end, y open/close.

- Dado que la mayoría de nombres se construyen concatenando varias palabras, emplee una mezcla de mayúsculas y minúsculas para simplificar la lectura. Además, para ayudar a distinguir entre variables y rutinas, utilice el método Pascal de mayúsculas y minúsculas (CalculateInvoiceTotal) para los nombres de rutinas, en el que la primera letra de cada palabra está en mayúscula. Para las variables, ponga la primera letra de cada palabra en mayúscula, exceptuando la primera (documentFormatType).

- Los nombres de variables booleanas deberían contener "Is", lo que implica valores del tipo Yes/No o True/False, como por ejemplo fileIsFound.

- Evite usar términos del tipo Flag cuando ponga nombre a variables de estado, que difieren de las variables booleanas en que aquéllas deben tener más de dos valores posibles. En vez de documentFlag, utilice un nombre más descriptivo, del tipo documentFormatType.

- Incluso para el caso de una variable de poco uso, que deba aparecer sólo en unas cuantas líneas de código, emplee un nombre descriptivo. Utilice nombres de variables de una sola letra, como i o j sólo para índices cortos.

- No utilice números o cadenas literales, como por ejemplo For i = 1 To 7. En su lugar, emplee constantes con nombre, del tipo For i = 1 To NUM_DAYS_IN_WEEK para que resulten fáciles de mantener y comprender.

Tablas

- Cuando ponga nombres a tablas, hágalo en singular. Por ejemplo, use Employee en lugar de Employees.

- Cuando ponga nombre a las columnas de las tablas, no repita el nombre de la tabla; por ejemplo, evite un campo llamado EmployeeLastName de una tabla llamada Employee.

- No incorpore el tipo de datos en el nombre de una columna. Así reducirá el esfuerzo que podría ser necesario posteriormente para cambiar el tipo de datos.

Varios

- Minimice el uso de abreviaturas; pero si las emplea, use coherentemente las que haya creado. Una abreviatura sólo debe tener un significado y, del mismo modo, a cada palabra abreviada sólo debe corresponder una abreviatura. Por ejemplo, si utiliza "min." para abreviar "mínimo", hágalo siempre así, y no use también "min." para abreviar "minuto".

- Cuando ponga nombre a las funciones, incluya una descripción del valor que vaya a ser devuelto, como por ejemplo GetCurrentWindowName().

- Los archivos y los nombres de carpetas, al igual que los nombres de procedimientos, deben describir claramente su finalidad.

- Evite reutilizar nombres para elementos diferentes, como por ejemplo una rutina llamada ProcessSales() y una variable iProcessSales.

- Evite los homónimos, como *write* y *right*, para evitar confusiones durante las revisiones del código.

- Al dar nombre a los elementos, evite las palabras mal escritas comúnmente. Tenga también cuidado con las diferencias que existen entre las diferentes variedades regionales del idioma.

- Evite las marcas tipográficas para identificar tipos de datos, como $ para las cadenas, o % para los enteros.

Comentarios

Existen dos tipos de documentación de software: externa e interna. La documentación externa, como por ejemplo las especificaciones, los archivos de ayuda y los documentos de diseño, se mantiene fuera del código fuente. La documentación interna está formada por los comentarios que los programadores escriben dentro del código fuente durante la fase de desarrollo.

Pese a la disponibilidad de la documentación externa, debe contarse con listados independientes del código fuente, por si se perdiera la documentación de recuperación impresa. La documentación externa puede constar de especificaciones, documentos de diseño, peticiones de cambios, historial de errores y el estándar de codificación empleado.

Uno de los problemas de la documentación de software interna es garantizar que se mantienen y actualizan los comentarios al mismo tiempo que el código fuente. Aunque unos buenos comentarios en el código fuente no tienen ningún valor en el tiempo de ejecución, resultan valiosísimos para un programador que tenga que mantener una parte de software particularmente intrincada o compleja.

Los siguientes puntos son técnicas de comentarios recomendadas.

- Si programa en C#, utilice la función de documentación XML.
- Cuando modifique el código, mantenga siempre actualizados los comentarios circundantes.
- Al principio de cada rutina, resulta útil hacer comentarios estándar, repetitivos, que indiquen el propósito de la rutina, las suposiciones y las limitaciones. Un comentario repetitivo podría consistir en una breve introducción que explicara por qué existe y qué puede hacer.
- Evite añadir comentarios al final de una línea de código, porque lo hacen más difícil de leer. Sin embargo, los comentarios de final de línea sí son apropiados al anotar declaraciones de variables. En este caso, alinee todos los comentarios de final de línea en la misma posición de tabulación.

- Evite los comentarios recargados, como las líneas enteras de asteriscos. En su lugar, utilice espacios para separar los comentarios y el código.

- Evite rodear un bloque de comentarios con un marco tipográfico. Puede resultar agradable, pero es difícil de mantener.

- Antes de la implementación, quite todos los comentarios temporales o innecesarios, para evitar cualquier confusión en la futura fase de mantenimiento.

- Si necesita realizar comentarios para explicar una sección de código compleja, examine el código para decidir si debería volver a escribirlo. Siempre que sea posible, no documente un código malo, vuelva a escribirlo. Aunque, por regla general, no debe sacrificarse el rendimiento para hacer un código más simple para el usuario, es indispensable un equilibrio entre rendimiento y mantenibilidad.

- Use frases completas cuando escriba comentarios. Los comentarios deben aclarar el código, no añadirle ambigüedad.

- Vaya comentando al mismo tiempo que programa, porque probablemente no tenga tiempo de hacerlo más tarde. Por otro lado, aunque tuviera oportunidad de revisar el código que ha escrito, lo que parece obvio hoy es posible que seis semanas después no lo sea.

- Evite comentarios superfluos o inapropiados, como comentarios divertidos al margen.

- Use los comentarios para explicar el propósito del código. No los use como si fueran traducciones interlineales.

- Comente cualquier cosa que no sea legible de forma obvia en el código.

- Para evitar problemas recurrentes, haga siempre comentarios al depurar errores y solucionar problemas de codificación, especialmente cuando trabaje en equipo.

- Haga comentarios en el código que esté formado por bucles o bifurcaciones lógicas. Se trata en estos casos de áreas clave que ayudarán a los lectores del código fuente.

- Realice los comentarios en un estilo uniforme, respetando una puntuación y estructura coherentes a lo largo de toda la aplicación.

- Separe los comentarios de sus delimitadores mediante espacios. Si respeta estas normas, los comentarios serán más claros y fáciles de localizar si trabaja sin indicaciones de color.

Formato

El formato hace que la organización lógica del código sea más clara. Si toma el tiempo de comprobar que el código fuente posee un formato coherente y lógico, les resultará de gran utilidad a usted y a otros programadores que tengan que descifrarlo.

Los siguientes puntos son técnicas de formato recomendadas.

- Establezca un tamaño estándar de sangría (por ejemplo, cuatro espacios) y úselo siempre. Alinee las secciones de código mediante la sangría predeterminada.

- Use un único tipo de letra cuando publique versiones impresas del código fuente.

- Alinee verticalmente las llaves de apertura y cierre donde los pares de llaves se alinean, como por ejemplo en:

```
for (i = 0; i < 100; i++)

{

   ...

}
```

También puede usar un estilo inclinado, en el que las llaves de apertura aparezcan al final de la línea y las de cierre al principio, como por ejemplo en:

```
for (i = 0; i < 100; i++){

   ...

}
```

Sea cual sea el estilo que escoja, úselo siempre en todo el código fuente.

- Aplique una sangría al código en todas las líneas de construcción lógica. Si no aplica la sangría, el código resulta difícil de seguir, como ocurre en:

```
If ... Then

If ... Then

...

Else

End If

Else

...

End If
```

La sangría aplicada al código hace que éste sea más fácil de leer, como por ejemplo en:

```
If ... Then

  If ... Then

  ...

  Else

  ...

  End If

Else

...

End If
```

- Establezca una longitud de línea máxima para los comentarios y el código. Así no tendrá que desplazarse en el editor del código fuente, y conseguirá presentaciones impresas claras.

- Utilice espacios antes y después de los operadores siempre que eso no altere la sangría aplicada al código. Una excepción es, por ejemplo, la pointer notation usada en C++.

- Use espacios en blanco para proporcionar indicaciones organizativas del código fuente. Así, creará "párrafos" de código, que ayudarán al lector a comprender la segmentación lógica del software.

- Cuando tenga que dividir una línea en varias, aclare que el código sigue en la línea de más abajo mediante un operador de concatenación colocado al final de cada línea, y no al principio.

- Siempre que sea posible, no coloque más de una instrucción por línea. Una excepción es un bucle en C, C++, C#, o JScript, como en for (i = 0; i < 100; i++).

- Al escribir en HTML, establezca un formato estándar para las etiquetas y los atributos; como por ejemplo, las etiquetas siempre en mayúscula y los atributos en minúscula. Como alternativa, siga las convenciones de la especificación XHTML para asegurarse de que los documentos HTML son válidos. Aunque existe una tamaño razonable que debe tenerse en cuenta a la hora de crear páginas Web, utilice valores de atributo entre comillas y etiquetas de cierre, para una mejor mantenibilidad.

- Cuando escriba instrucciones SQL utilice mayúsculas para las palabras clave y mayúsculas y minúsculas para los elementos de la base de datos, como tablas, columnas y vistas.

- Divida el código fuente de manera lógica entre diferentes archivos.

- Coloque las cláusulas SQL principales en líneas separadas, de modo que las instrucciones sean más fáciles de leer y editar. Por ejemplo:

```
SELECT FirstName, LastName

FROM Customers

WHERE State = 'WA'
```

- Divida las secciones de código grandes y complejas en otras más pequeñas y comprensibles.

Framework

Los frameworks son la piedra angular de la moderna ingeniería del software. El desarrollo del framework está ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente (source code). Los frameworks son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.

Como ejemplo, considere la construcción de un kit de herramientas de interface gráfica del usuario (GUI Tool Kit). Puede ser que elijamos diseñar y poner un solo kit de herramientas en ejecución. Por otra parte, si diseñamos el kit de herramientas como framework, este puro diseño nos permitirá generar una colección de los kits de herramientas para una variedad de aplicaciones del tipo GUI (Graphic User Interface). Los frameworks deben generar las aplicaciones para un dominio entero. Por lo tanto, debe haber puntos de flexibilidad que se puedan modificar los requisitos particulares para ajustarse a la aplicación. Por ejemplo, un punto de extensión puede ser el algoritmo usado para trazar elementos gráficos.

Los puntos flexibles de un framework se llaman los puntos calientes (hot-spots). Los puntos calientes o Hot-spots son las clases o los métodos abstractos que deben ser implementados o puestos en ejecución. Los frameworks no son ejecutables. Generar un ejecutable, uno debe "instantiar" el framework (llámese Instantiar, al hecho de producir y completar un objeto llenando con valores en lugar de variables en un class template) poniendo el código específico de la aplicación en ejecución para cada punto caliente. Una vez que los puntos calientes sean "instanciados", el framework utilizará aquellas clases usando el callback o repetición de la llamada (acto de repetir la autentificación del número de usuario en caso de reconexión) . En esta repetición de la llamada o callback, el código del usuario del servicio declara que desea ser llamado en la ocurrencia un determinado evento. Entonces, el código del proveedor del servicio realiza la repetición de la llamada o callback con el código del usuario del servicio al momento de ocurrir ese determinado evento. Por esta razón, en primera instancia, el framework se caracteriza a veces como " el viejo código que llama al nuevo código."

Algunas de las características del framework no son mutables ni tampoco pueden ser alteradas fácilmente. Estos puntos inmutables constituyen el núcleo o kernel de un framework, también llamados como los puntos congelados o frozen-spots del framework. A diferencia de los puntos calientes o hot-spots, los puntos congelados o inmutables son los pedacitos del código puestos en ejecución ya dentro del framework que llaman a uno o más puntos calientes proporcionados por el ejecutor. El núcleo o Kernel será la constante y presentará siempre la parte de cada instancia del framework.

Desarrollo del Framework

Las tres etapas principales del desarrollo del framework son análisis del dominio, diseño del framework, y la "instanciación" del framework.

El análisis del dominio procura descubrir los requisitos del dominio y los posibles requerimientos futuros. Para completar los requerimientos sirven las experiencias previamente publicadas, los sistemas de software similares existentes, las experiencias personales, y los estándares considerados. Durante el análisis del dominio, los puntos calientes y los puntos congelados se destapan parcialmente.

La fase del diseño del framework define las abstracciones de éste. Se modelan los puntos calientes y los puntos congelados (quizás con diagrama de UML, Modelo Unificado del Lenguaje, *Unified Modeling Language*) , y la extensión y la flexibilidad propuesta en el análisis del dominio se esboza en líneas generales. Según lo mencionado arriba, los modelos del diseño se utilizan en esta fase.

Finalmente, en la fase de "instanciación", los puntos calientes del framework son implementados, generando un software del sistema. Es importante observar que cada uno de estas aplicaciones tendrá los puntos congelados del framework en común. Las fases del proceso del desarrollo del framework son comparados con las tradicionales fases del diseño orientados al objeto, según se puede apreciar en el cuadro 3. En esta figura, nombramos las fases del desarrollo según lo descrito.

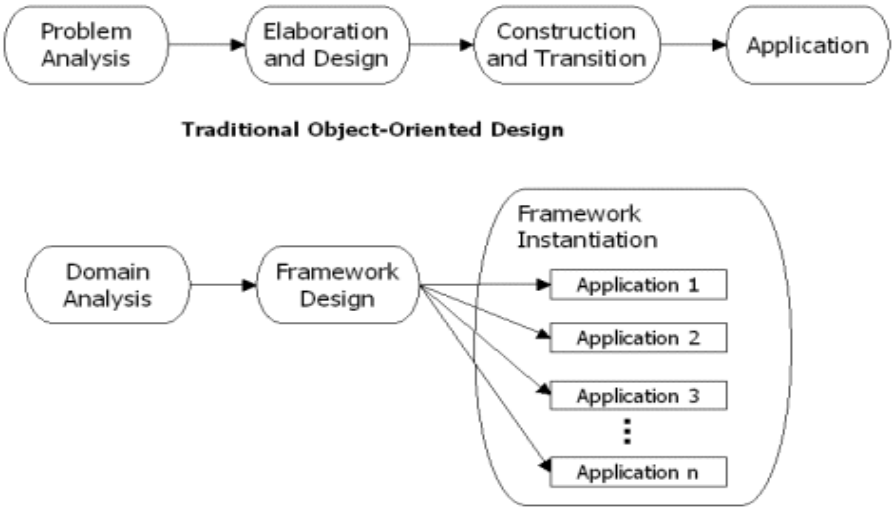

Figura 5.- Ejemplo del diseño tradicional de la orientación a objetos.

.Proceso de desarrollo del framework

Según la representación anterior, el desarrollo tradicional Orientado al Objeto se diferencia del desarrollo del framework. En el desarrollo tradicional Orientado al Objeto, la fase de análisis del problema, también llamada inicio, estudia solamente los requisitos de un solo problema. En cambio, en el desarrollo del framework captura los requisitos para un dominio entero. Además, el resultado final del desarrollo orientado al objeto tradicional es una aplicación que es completamente ejecutable, mientras que muchas aplicaciones resultan a partir de la fase de "instanciación" del desarrollo del framework. La fase del "instanciación" abarca las fases de construcción y de transición del desarrollo tradicional. Así, la construcción y las fases separadas de la transición están presentes en cada uno de las instancias del framework. Para cada una de las instancias del framework hay un esfuerzo de implementación o puesta en práctica introducido por estas fases.

¿Qué ventajas tiene utilizar un framework?

Las que se derivan de utilizar un estándar; entre otras:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el *framework* le proporciona un esqueleto que hay que "rellenar".

- Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (¡o incluso con el propio, pasado algún tiempo!) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.

- Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al *framework* concreto para facilitar el desarrollo.

En el ámbito del desarrollo para la web, los patrones de diseño más utilizados son aquellos que se centran en separar la presentación (páginas html, css) de la lógica o back-end. Esto porque un típico equipo de desarrollo consiste en programadores por un lado y diseñadores por el otro. Separando efectivamente las tareas de cada uno mediante una arquitectura estándar comprendida por todos -un patrón de diseño- facilita enormemente el trabajo del equipo.
De estos patrones, el más popular es MVC (Modelo Vista Controlador), muy conocido en el mundo de Java y el implementado por Ruby on Rails.

Como su nombre lo dice, MVC consiste en separar lo mejor posible las capas de Modelo (los

objetos que interactúan con la base de datos y efectúan los procesos pesados o “lógica de negocios”), la Vista (la presentación final de los datos procesados al cliente, comúnmente en formato HTML) y el Controlador (la capa que se encarga de recibir el input del usuario, delegar el trabajo a los Modelos apropiados e invocar las Vistas que correspondan).

Los frameworks, entonces, suelen ser implementaciones de patrones de diseño conocidos, aderezados con funciones que asisten al desarrollador.

La utilización de un *framework* en el desarrollo de una aplicación implica un cierto coste inicial de aprendizaje, aunque a largo plazo es probable que facilite tanto el desarrollo como el mantenimiento.

Un enfoque a los frameworks debe considerar cuando los requisitos del producto cambian rápidamente. Los frameworks se pueden también utilizar para aumentar el desarrollo; implementando al principio un código hot-spot simple y posteriormente actualizarlo. Una buena referencia de frameworks es, aquella que hace una buena evaluación de su metodología y sus últimos avances.

Ejemplos de frameworks

Framework Hibernate de código abierto basado en lenguaje Java, de persistencia relacional

Framework Nhibernate de código abierto basado en tecnología .NET, de persistencia relacional

Framework Spring de código abierto basado en Java

Framework Symfony de código abierto basado en PHP, siguiendo el patrón de arquitectura Modelo Vista Controlador

Frameworks de Microsoft .NET

Client Side(lado-cliente, del lado del cliente).

En redes de computadoras, el término client-side hace referencia a las operaciones que son realizadas por el cliente en una relación cliente/servidor. Su recíproco es server-side.

Generalmente un cliente es una aplicación de computadora (ejemplo: un navegador web), que se ejecuta en la computadora local del usuario o estación de trabajo, y se conecta a un servidor.

Las operaciones pueden ser realizadas en el lado-cliente porque:
* Requieren acceso a información o funcionalidades que están disponible en el cliente y no en el servidor.
* El usuario necesita ver o proveer las entradas.
* El servidor carece de la potencia necesaria como para realizar las operaciones en tiempo para todos los clientes a los que sirve.

También, si las operaciones pueden ser hechas del lado del cliente, sin enviar datos a través de la

red hacia el servidor, puede tomar menos tiempo, usar menos ancho de banda e incluso disminuir los riesgos de seguridad

En el sentido estricto de la palabra nuestro Navegador ("Explorer" o "Netscape") constantemente ejecuta Aplicaciones o "Programas", de aquí surge el nombre "Aplicación en Cliente". Cuando se nos envía un documento de un "Servidor de Páginas" esta información viene en HTML / XHTML ("Hypertext Markup Language") normalmente, al llegar a nuestro navegador esta información es "ejecutada" o desplegada de acuerdo al formato correspondiente.

Sin embargo, en muchas ocasiones (casi siempre) esta información no solo viene en HTML / XHTML ,además de las imágenes GIF o JPEG , la información puede venir en otro lenguaje como JavaScript, Tcl/Tk o VBScript, estos lenguajes son llamados "Scripting Languages"; inclusive también puede venir un programa en Java (este no es "Scripting language") que lleva por nombre "Applet" . Estos "Scripting Languages" o Java en sí, permiten dar una mayor flexibilidad a la interface (lo que usted observa en pantalla) de aquella que tiene HTML / XHTML, ya que HTML / XHTML es solo un lenguaje de marcación, como sus siglas lo indican (ML) Markup Language , solo define como será desplegada la información, debido a esto surgieron los "Scripting Languages" que permiten ir más allá del despliegue de información.

Quizás el "Scripting Language" de mayor uso sea "JavaScript" uno de los pocos lenguajes ("Scripting Languages") compatible tanto con "Netscape Navigator" e "Internet Explorer" , los demás "Scripting Languages" así como Java, han empezado a ser utilizados conforme las versiones de Navegadores han sido capaces de ejecutarlos, aunque cuando algún Navegador popular no soporta algún lenguaje es muy posible que requiera un Plug-in.

Plug-ins...Plug-ins...Plug-ins

Seguramente en alguna ocasión se le ha solicitado que baje un "plug-in" ya sea para observar un vídeo u oír alguna grabación. Este "plug-in" es un adaptador que permite al "Navegador" ejecutar y desplegar apropiadamente la información que usted está bajando de Internet, esto es, si la página que usted desea observar contiene "Active X" y usted está utilizando "Netscape Navigator" es muy probable que su "Navegador" requiera de un "plug-in" sobre todo si está utilizando una versión antigua de "Netscape".

Antiguamente (1997) era muy probable que toda página que usted observara ocupara de un "plug-in", esto limitaba a los desarrolladores y diseñadores de páginas, ya que ellos no tenían mucha flexibilidad de presentarle información, era preferible presentarle páginas sencillas a tener que hacer al cliente final (usted) tener que bajar ("download") un "plug-in", y en dado caso que requirieran al cliente bajar el plug-in,era muy probable que este plug-in no funcionara

apropiadamente y en el peor de los casos dañara o paralizara su Navegador algo MUY común cuando se visitaba un sitio que tuviera Java o "Active X".

Esto obviamente ha cambiado y actualmente los principales navegadores "Netscape" e "Internet Explorer" son compatibles con varios "Scripting Languages" así como con Java y "Active X" , lo cual permite a los desarrolladores de páginas tener mayor flexibilidad para desarrollar páginas, y a la vez le permite al usuario final tener una interface más agradable, aunque de eso depende el tipo de conexión que tenga.

Lenguajes del lado cliente

HTML 

El lenguaje llamado HTML indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página.

El lenguaje consta de etiquetas que tienen esta forma <B> o <P>. Cada etiqueta significa una cosa, por ejemplo <B> significa que se escriba en negrita (bold) o <P> significa un párrafo, <A> es un enlace, etc. Casi todas las etiquetas tienen su correspondiente etiqueta de cierre, que indica que a partir de ese punto no debe de afectar la etiqueta. Por ejemplo </B> se utiliza para indicar que se deje de escribir en negrita. Así que el HTML no es más que una serie de etiquetas que se utilizan para definir la forma o estilo que queremos aplicar a nuestro documento. <B>Esto está en negrita</B>.

Esta página es un claro ejemplo de uso del HTML.

JAVASCRIPT 

Javascript es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario.

Las sentencias escritas en JavaScript se encapsulan entre las etiquetas <script> y </script>. por ejemplo, si en el código de una página Web incluimos la sentencia

```
<script>
window.alert("Bienvenido a mi sitio web. Gracias...")
</script>
```

al abrir la página con el navegador se nos mostrará una ventana de bienvenida

APPLETS DE JAVA

Es otra manera de incluir código a ejecutar en los clientes que visualizan una página web. Se trata de pequeños programas hechos en Java, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página.

Los applets de Java están programados en Java y pre compilados, es por ello que la manera de trabajar de éstos varía un poco con respecto a los lenguajes de script como Javascript. Los applets son más difíciles de programar que los scripts en Javascript y requerirán unos conocimientos básicos o medios del lenguaje Java.

La principal ventaja de utilizar applets consiste en que son mucho menos dependientes del navegador que los scripts en Javascript, incluso independientes del sistema operativo del ordenador donde se ejecutan. Además, Java es más potente que Javascript, por lo que el número de aplicaciones de los applets podrá ser mayor.

Como desventajas en relación con Javascript cabe señalar que los applets son más lentos de procesar y que tienen espacio muy delimitado en la página donde se ejecutan, es decir, no se mezclan con todos los componentes de la página ni tienen acceso a ellos. Es por ello que con los applets de Java no podremos hacer directamente cosas como abrir ventanas secundarias, controlar Frames, formularios, capas, etc.

VISUAL BASIC SCRIPT

Es un lenguaje de programación de scripts del lado del cliente, pero sólo compatible con Internet Explorer. Es por ello que su utilización está desaconsejada a favor de Javascript.

Está basado en Visual Basic, un popular lenguaje para crear aplicaciones Windows. Tanto su sintaxis como la manera de trabajar están muy inspiradas en él. Sin embargo, no todo lo que se

puede hacer en Visual Basic lo podremos hacer en Visual Basic Script, pues este último es una versión reducida del primero.

El modo de funcionamiento de Visual Basic Script para construir efectos especiales en páginas web es muy similar al utilizado en Javascript y los recursos a los que se puede acceder también son los mismos: el navegador.

FLASH

Flash es una tecnología, y un programa, para crear efectos especiales en páginas web. Con Flash también conseguimos hacer páginas dinámicas del lado del cliente. Flash en realidad no es un lenguaje; Sin embargo, si tuviéramos que catalogarlo en algún sitio quedaría dentro del ámbito de las páginas dinámicas de cliente.

Para visualizar las "películas" Flash, nuestro navegador debe tener instalado un programita (plug-in) que le permita visualizarlas.

CSS

CSS, es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores... Incluso podemos definir nuestros propios estilos en un archivo externo a nuestras páginas; así, si en algún momento queremos cambiar alguno de ellos, automáticamente se nos actualizarán todas las páginas vinculadas de nuestro sitio.

CSS son las siglas de Cascading Style Sheets, en español Hojas de estilo en Cascada.

Navegadores

Un navegador o navegador web (del inglés, *web browser*) es un programa que permite visualizar la información que contiene una página web (ya esté está alojada en un servidor dentro de la World Wide Web o en uno local).

El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Los documentos pueden estar ubicados en la computadora en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado a la computadora del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor web). Tales documentos, comúnmente denominados *páginas web*, poseen *hipervínculos* que enlazan una porción de texto o una imagen a otro documento, normalmente relacionado con el texto o la imagen.

El seguimiento de enlaces de una página a otra, ubicada en cualquier computadora conectada a la Internet, se llama *navegación*; que es de donde se origina el nombre de navegador. Por otro lado, hojeador es una traducción literal del original en inglés, *browser*, aunque su uso es minoritario.

La comunicación entre el servidor web y el navegador se realiza mediante el protocolo HTTP, aunque la mayoría de los hojeadores soportan otros protocolos como FTP, Gopher, y HTTPS (una versión cifrada de HTTP basada en Secure Socket Layer o Capa de Conexión Segura (SSL)).

La función principal del navegador es descargar documentos HTML y mostrarlos en pantalla. En la actualidad, no solamente descargan este tipo de documentos sino que muestran con el documento sus imágenes, sonidos e incluso vídeos *streaming* en diferentes formatos y protocolos. Además, permiten almacenar la información en el disco o crear marcadores (*bookmarks*) de las páginas más visitadas.

Algunos de los navegadores web más populares se incluyen en lo que se denomina una Suite. Estas Suite disponen de varios programas integrados para leer noticias de Usenet y correo electrónico mediante los protocolos NNTP, IMAP y POP.

Los primeros navegadores web sólo soportaban una versión muy simple de HTML. El rápido desarrollo de los navegadores web propietarios condujo al desarrollo de dialectos no estándares de HTML y a problemas de interoperabilidad en la web. Los más modernos (como Amaya, Mozilla, Netscape, Opera y versiones recientes de Internet Explorer) soportan los estándares HTML y XHTML (comenzando con HTML 4.01, los cuales deberían visualizarse de la misma manera en todos ellos).

Los estándares web son publicados por el World Wide Web Consortium.

Ejemplos de los navegadores más comunes

Google Chrome 

Google Chrome es un navegador web que ejecuta aplicaciones y páginas web a gran velocidad

Características importantes:

Velocidad: rápido a la hora de iniciarse y de cargar páginas web

- Google Chrome se inicia rápidamente desde cualquier equipo.
- Google Chrome carga páginas web de forma instantánea.
- Google Chrome ejecuta páginas web interactivas, aplicaciones web y JavaScript a una velocidad sin precedentes.

Sencillez: diseñado para ser eficaz y fácil de utilizar

- Realiza búsquedas y desplázate a páginas web desde el mismo cuadro.
- Ordena y organiza tus pestañas de la forma que desees (rápida y fácilmente).
- Accede a tus sitios web favoritos con un solo clic desde las miniaturas de tus sitios más visitados que aparecerán en la página Nueva pestaña.

Internet Explorer 

Windows Internet Explorer (anteriormente Microsoft Internet Explorer; abreviado MSIE), conocido comúnmente como Internet Explorer y abreviado IE, es un navegador web desarrollado por Microsoft para el sistema operativo Windows desde 1995 y más tarde para Sun Solaris y Apple Macintosh, estas dos últimas discontinuadas en el 2002 y 2006 respectivamente. Ha sido el navegador web más utilizado desde 1999, con un pico sostenido de cuota de utilización durante el 2002 y 2003 del 95% en sus versiones 5 y 6. Esa cuota de mercado ha disminuido paulatinamente debido a una renovada competencia por parte de otros navegadores, principalmente Mozilla Firefox. Microsoft gastó más de 100 millones de dólares (USD) al año en el decenio de 1990, con más de 1000 personas trabajando en IE para 1999.

Su versión más reciente es la 8.0, la cual está disponible gratuitamente como actualización para Windows XP Service Pack 2, Windows Server 2003 con Service Pack 1 o posterior, Windows Vista, y Windows Server 2008. Internet Explorer 8 se incluirá de forma nativa en los próximos sistemas operativos de Microsoft, Windows 7 y Windows Server 2008 R2.

Características importantes:

- Visita cualquier sitio con facilidad: ahora puedes ver rápidamente sitios web diseñados para exploradores anteriores. Si estás en una página donde el texto o las imágenes no están alineadas, simplemente usa el botón Vista de compatibilidad junto al botón Actualizar en la barra de direcciones.
- Realiza tus tareas más rápido con los aceleradores: con solo unos pocos clics puedes obtener indicaciones de rutas, traducir palabras, compartir lo que encuentras en la Web con tus amigos y mucho más. Los aceleradores te ayudan a realizar tus acciones cotidianas sin necesidad de abrir una nueva ventana. De esta manera puedes realizar rápidamente tus tareas de exploración diarias, como obtener un mapa de la dirección de una empresa o reenviar un vínculo a un amigo.
- Seguridad en línea: explora con más confianza seguro de que Internet Explorer 8 te ayuda a estar protegido contra las amenazas emergentes en línea. El nuevo filtro SmartScreen y otras funciones de seguridad integradas te ayudan a protegerte contra sitios web engañosos y malintencionados que pueden poner en peligro tus datos, tu privacidad y tu identidad.

Mozilla Firefox 

Firefox es un navegador web gratuito de Mozilla Corporation que te permite navegar por Internet de forma segura y rápida.

Firefox, además de ser más rápido y seguro, es un navegador completamente personalizable

- Características importantes navegación con pestañas
- Posibilidad de restaurar una sesión si se ha cerrado el navegador porque el ordenador se ha reiniciado por algún problema
- Corrector ortográfico
- Sugerencias de búsqueda
- Canales RSS
- Títulos dinámicos
- Búsqueda integrada: tiene una barra de búsqueda integrada con motores de búsqueda de Google, Yahoo!, eBay, Diccionario RAE, Wikipedia y Creative Commons por defecto. Puedes añadir más.
- Marcadores dinámicos

Bloqueador de ventanas emergentes

En cuanto a seguridad Firefox incluye:

- Protección antiphishing.
- Actualizaciones automáticas.
- Protección contra programas espías.
- Posibilidad de limpiar la información privada.

OPERA 

Opera es un navegador Web **muy rápido** que ha sido diseñado para hacer más práctica y cómoda la navegación del usuario.

- Navegar con los gestos del ratón: permite navegar y/o ejecutar funciones del navegador utilizando movimientos del mouse.
- Speed Dial: página de inicio con miniaturas de sitios seleccionados.
- BitTorrent: cliente de Torrent incorporado.
- Protección anti-fraude: permite detectar sitios inseguros y phishing.
- Widgets: pequeñas aplicaciones desarrolladas por usuarios.
- Vista previa en miniatura: al pasar el cursor sobre una pestaña, aparece la vista previa.
- Soporte de RSS y Atom (sindicación de contenido): cuando un sitio web ofrece este servicio, Opera muestra un icono en la barra de dirección para suscribirse y comprobar automáticamente el nuevo contenido del mismo.
- Administrador de descargas: permite, entre otras cosas, pausar cualquier descarga y continuarla cuando así se requiera (especialmente útil con archivos grandes), o realizar múltiples descargas simultáneamente llevando una mejor monitorización, así como poder llevar un historial de las mismas. Además, una vez descargado el archivo, ofrece todas las opciones del menú contextual normal.
- Navegación por sesiones: permite guardar la sesión de navegación actual y volver a retomarla. Así, el usuario puede cerrar el navegador, y abrirlo más tarde teniendo exactamente las mismas páginas abiertas e historial por pestaña que cuando lo cerró. Se pueden guardar múltiples sesiones.
- Editar opciones por sitio: permite guardar la configuración de las cookies, las ventanas emergentes, el comportamiento de java entre otras cosas.

Safari 

Carga páginas web a la velocidad de la luz, ofrece prestaciones nunca vistas que hacen que navegar sea mucho más divertido y funciona tanto en el Mac como en PC, el iPhone y el iPod touch. Presentamos Safari, el navegador más innovador del mundo.

Safari ha sido diseñado para realzar la navegación, no el navegador. Su marco apenas mide un píxel de ancho y vez la barra de desplazamiento sólo cuando la necesitas. Por omisión, no hay barra de estado, sino un indicador de progreso que gira mientras la página se carga. Encontrarás las pestañas en la parte superior del navegador, en una ventana más amplia todavía para ver los sitios web. Safari, un navegador excepcional, te permite disfrutar de la Red: ya la estés explorando en un Mac, un PC, un iPhone o un iPod touch.

Safari, el primer navegador en ofrecer Internet de verdad para dispositivos móviles, carga las páginas web en el iPhone y el iPod touch como si las vieras en tu ordenador. Es mucho más que una mera versión móvil reducida del original: Safari saca partido de las tecnologías integradas en estos dispositivos Multi-Touch y, cuando ladeas tu iPhone o iPod touch, la página se coloca y ajusta su forma a la ventana, y puedes alejar y acercar la pantalla de un pellizco. Independientemente del dispositivo en que lo utilices, Safari siempre es veloz y fácil de usar.

HTTP

El protocolo de transferencia de hipertexto (HTTP, *HyperText Transfer Protocol*) es el protocolo usado en cada transacción de la Web (WWW). HTTP fue desarrollado por el consorcio W3C y la IETF, colaboración que culminó en 1999 con la publicación de una serie de RFC, siendo el más importante de ellos el RFC 2616, que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador o un spider) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un URL. Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

Una transacción HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato. El encabezado especificará cosas como la acción requerida del servidor, o el tipo de dato retornado, o el código de estado.

El uso de campos de encabezados enviados en las transacciones HTTP le da gran flexibilidad al protocolo. Estos campos permiten que se envíe información descriptiva en la transacción, permitiendo así la autenticación, cifrado e identificación de usuario.

Un encabezado es un bloque de datos que precede a la información propiamente dicha, por lo que muchas veces se hace referencia a él como metadato —porque tiene datos sobre los datos—.

Si se reciben líneas de encabezado del cliente, el servidor las coloca en las variables de ambiente de CGI con el prefijo HTTP_ seguido del nombre del encabezado. Cualquier carácter guión (-) del nombre del encabezado se convierte a caracteres "_".

El servidor puede excluir cualquier encabezado que ya esté procesado, como *Authorization*, *Content-type* y *Content-length*. El servidor puede elegir excluir alguno o todos los encabezados si

incluirlos exceden algún límite del ambiente de sistema. Ejemplos de esto son las variables HTTP_ACCEPT y HTTP_USER_AGENT.

- HTTP_ACCEPT. Los tipos MIME que el cliente aceptará, dado los encabezados HTTP. Otros protocolos quizás necesiten obtener esta información de otro lugar. Los elementos de esta lista deben estar separados por una coma, como lo dice la especificación HTTP: tipo, tipo.

- HTTP_USER_AGENT. El navegador que utiliza el cliente para realizar la petición. El formato general para esta variable es: software/versión biblioteca/versión.

El servidor envía al cliente:

- Un código de estado que indica si la petición fue correcta o no. Los códigos de error típicos indican que el archivo solicitado no se encontró, que la petición no se realizó de forma correcta o que se requiere autenticación para acceder al archivo.

- La información propiamente dicha. Como HTTP permite enviar documentos de todo tipo y formato, es ideal para transmitir multimedia, como gráficos, audio y video. Esta libertad es una de las mayores ventajas de HTTP.

- Información sobre el objeto que se retorna.

Ten en cuenta que la lista no es una lista completa de los campos de encabezado y que algunos de ellos sólo tienen sentido en una dirección.

HTTPS

Hypertext Transfer Protocol Secure (en español *Protocolo seguro de transferencia de hipertexto*), más conocido por sus siglas HTTPS, es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

Es utilizado principalmente por entidades bancarias, tiendas en línea, y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.

La idea principal de https es la de crear un canal seguro sobre una red insegura. Esto proporciona una protección razonable contra ataques eavesdropping y man-in-the-middle, siempre que se empleen métodos de cifrado adecuados y que el *certificado del servidor sea verificado y resulte de confianza*.

La confianza inherente en HTTPS está basada en una Autoridad de certificación superior que viene preinstalada en el software del navegador (Es el equivalente a decir "Confío en la autoridad de certificación (por ejemplo. VeriSign/Microsoft/etc.) Para decirme en quien debería confiar"). Sin embargo una conexión HTTPS a un website puede ser validada si y solo si todo lo siguiente es verdad:

1. El usuario confía en la Autoridad de certificación para dar fe solo para websites legítimos sin nombres engañosos.

2. El website proporciona un certificado válido (y un certificado inválido muestra una alerta en la mayoría de los navegadores), lo que significa que está firmado por una autoridad confiable.

3. El certificado identifica correctamente al website (por ejemplo. visitando https://algunsitio y recibiendo un certificado para "AlgunSitio S.A." y no AlgunZitio S.A."

4. Cada uno de los nodos involucrados en internet son dignos de confianza, o que el usuario confíe en que la capa de cifrado del protocolo (TLS o SSL) es inquebrantable por un eavesdropper(escuchar secretamente).

Integración con el navegador

Cabe mencionar que el uso del protocolo HTTPS no impide que se pueda utilizar HTTP. Es aquí cuando nuestro navegador nos advertirá sobre la carga de elementos no seguros (HTTP), estando conectados a un entorno seguro (HTTPS).

Los protocolos HTTPS son utilizados por navegadores como: Safari, Internet Explorer, Mozilla Firefox, Opera y Google Chrome, entre otros.

Algunos navegadores utilizan un icono (generalmente un candado) en la parte derecha de la barra de direcciones para indicar la existencia de un protocolo de comunicaciones seguro e incluso cambian el color del fondo de la barra de direcciones por amarillo (Firefox) o verde (Internet Explorer) para identificar páginas web seguras.

Cuando se conecta a un sitio con un certificado inválido, los navegadores antiguos podrían presentar al usuario una caja de diálogo preguntando si desean continuar. Los navegadores más modernos muestran una alerta a lo largo de toda la ventana. Los navegadores más modernos también muestran mucha más información de seguridad sobre el sitio en la barra de direcciones.

Los certificados de validación extendida vuelven la barra de direcciones verde en navegadores modernos. La mayoría de los navegadores también despliegan una alerta cuando el usuario visita un sitio que contiene una mezcla de contenidos cifrados y sin cifrar.

Para conocer si una página web que estamos visitando, utiliza el protocolo https y es, por tanto, segura en cuanto a la trasmisión de los datos que estamos transcribiendo, debemos observar si en la barra de direcciones de nuestro navegador, aparece https al comienzo, en lugar de http, o si no hacer un scan

SSL

Secure Sockets Layer -Protocolo de Capa de Conexión Segura-

Como funciona

El protocolo SSL intercambia registros; opcionalmente, cada registro puede ser comprimido, cifrado y empaquetado con un código de autentificación del mensaje (MAC). Cada registro tiene un campo de *content_type* que especifica el protocolo de nivel superior que se está usando.

Cuando se inicia la conexión, el nivel de registro encapsula otro protocolo, el protocolo *handshake*, que tiene el *content_type* 22.

El cliente envía y recibe varias estructuras *handshake*:

- Envía un mensaje *ClientHello* especificando una lista de conjunto de cifrados, métodos de compresión y la versión del protocolo SSL más alta permitida. Éste también envía bytes aleatorios que serán usados más tarde (llamados *Challenge de Cliente* o *Reto*). Además puede incluir el identificador de la sesión.

- Después, recibe un registro *ServerHello*, en el que el servidor elige los parámetros de conexión a partir de las opciones ofertadas con anterioridad por el cliente.

- Cuando los parámetros de la conexión son conocidos, cliente y servidor intercambian certificados (dependiendo de las claves públicas de cifrado seleccionadas). Estos certificados son actualmente X.509, pero hay también un borrador especificando el uso de certificados basados en OpenPGP.

- El servidor puede requerir un certificado al cliente, para que la conexión sea mutuamente autentificada.

- Cliente y servidor negocian una clave secreta (simétrica) común llamada *master secret*, posiblemente usando el resultado de un intercambio Diffie-Hellman, o simplemente cifrando una clave secreta con una clave pública que es descifrada con la clave privada de cada uno. Todos los datos de claves restantes son derivados a partir de este *master secret* (y los valores aleatorios generados en el cliente y el servidor), que son pasados a través una *función pseudoaleatoria* cuidadosamente elegida.

TLS

Transport Layer Security -Seguridad de la Capa de Transporte

Como funciona

TLS/SSL poseen una variedad de medidas de seguridad:

- Numerando todos los registros y usando el número de secuencia en el MAC.

- Usando un resumen de mensaje mejorado con una clave (de forma que solo con dicha clave se pueda comprobar el MAC). Esto se especifica en el RFC 2104).

- Protección contra varios ataques conocidos (incluyendo ataques man-in-the-middle), como los que implican un degradado del protocolo a versiones previas (por tanto, menos seguras), o conjuntos de cifrados más débiles.

- El mensaje que finaliza el protocolo *handshake* (*Finished*) envía un *hash* de todos los datos intercambiados y vistos por ambas partes.

- La función pseudo aleatoria divide los datos de entrada en 2 mitades y las procesa con algoritmos hash diferentes (MD5 y SHA), después realiza sobre ellos una operación XOR. De esta forma se protege a sí mismo de la eventualidad de que alguno de estos algoritmos se revelen vulnerables en el futuro.

Server Side (Server-side, del lado del servidor o lado-servidor).

Hace referencia a las operaciones que son llevadas a cabo por el servidor en una relación cliente-servidor en una red de computadoras (como internet u otras redes). Su recíproco es client-side(lado del cliente).

Una computadora actúa como servidora si posee un programa servidor ejecutándose. Por ejemplo, un servidor web que ofrece los servicios necesarios para que funcione un sitio web). Si la computadora está conectada a internet y con las configuraciones adecuadas, puede servir como prestadora de servicio para que otras personas puedan ver un sitio web. En este caso, las computadoras remotas que acceden al sitio serían los clientes (el navegador web se considera un programa cliente).

Un servidor sirve información a las computadoras que se conecten a él. Cuando los usuarios se conectan a un servidor pueden acceder a programas, archivos y otra información del servidor.

En la web, un servidor web es una computadora que usa el protocolo http para enviar páginas web a la computadora de un usuario cuando el usuario las solicita.

Los servidores web, servidores de correo y servidores de bases de datos son a lo que tiene acceso la mayoría de la gente al usar Internet.

Algunos servidores manejan solamente correo o solamente archivos, mientras que otros hacen más de un trabajo, ya que una misma computadora puede tener diferentes programas de servidor funcionando al mismo tiempo.

Los servidores se conectan a la red mediante una interfaz que puede ser una red verdadera o mediante conexión vía línea telefónica o digital.

Tipos de servidores:

- **Plataformas de Servidor** (Server Platforms): un término usado a menudo como sinónimo de sistema operativo, la plataforma es el hardware o software subyacentes para un sistema, es decir, el motor que dirige el servidor.
- **Servidores de Aplicaciones** (Application Servers): designados a veces como un tipo de middleware (software que conecta dos aplicaciones), los servidores de aplicaciones ocupan una gran parte del territorio entre los servidores de bases de datos y el usuario, y a menudo los conectan.
- **Servidores de Audio/Video** (Audio/Video Servers): los servidores de Audio/Video añaden capacidades multimedia a los sitios web permitiéndoles mostrar contenido multimedia en forma de flujo continuo (streaming) desde el servidor.
- **Servidores de Chat** (Chat Servers): los servidores de chat permiten intercambiar información a una gran cantidad de usuarios ofreciendo la posibilidad de llevar a cabo discusiones en tiempo real.
- **Servidores de Fax** (Fax Servers): un servidor de fax es una solución ideal para organizaciones que tratan de reducir el uso del teléfono pero necesitan enviar documentos por fax.
- **Servidores FTP** (FTP Servers): uno de los servicios más antiguos de Internet, File Transfer Protocol permite mover uno o más archivos.
- **Servidores Groupware** (Groupware Servers): un servidor groupware es un software diseñado para permitir colaborar a los usuarios, sin importar la localización, vía Internet o vía Intranet corporativo y trabajar juntos en una atmósfera virtual.
- **Servidores IRC** (IRC Servers): otra opción para usuarios que buscan la discusión en tiempo real, Internet Relay Chat consiste en varias redes de servidores separadas que permiten que los usuarios conecten el uno al otro vía una red IRC.
- **Servidores de Listas** (List Servers): los servidores de listas ofrecen una manera mejor de manejar listas de correo electrónico, bien sean discusiones interactivas abiertas al público o listas unidireccionales de anuncios, boletines de noticias o publicidad.
- **Servidores de Correo** (Mail Servers): casi tan ubicuos y cruciales como los servidores web, los servidores de correo mueven y almacenan el correo electrónico a través de las redes corporativas (vía LANs y WANs) y a través de Internet.
- **Servidores de Noticias** (News Servers): los servidores de noticias actúan como fuente de distribución y entrega para los millares de grupos de noticias públicos actualmente accesibles a través de la red de noticias USENET.

- **Servidores Proxy** (Proxy Servers): los servidores proxy se sitúan entre un programa del cliente (típicamente un navegador) y un servidor externo (típicamente otro servidor web) para filtrar peticiones, mejorar el funcionamiento y compartir conexiones.
- **Servidores Telnet** (Telnet Servers): un servidor telnet permite a los usuarios entrar en una computadora huésped y realizar tareas como si estuviera trabajando directamente en esa computadora.
- **Servidores Web** (Web Servers): básicamente, un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red

Las operaciones que se realizan del lado del servidor, requieren acceso a información o funcionalidades que no están disponibles del lado del cliente.

Por ejemplo, una base de datos o un motor intérprete de PHP o de ASP.

Otro ejemplo: un cliente solicita una página web como: patito.c**om.mx/index.php**, del lado del servidor el código de programación en PHP es interpretado, y al cliente se le envía sólo la página web resultado de ese proceso (el código HTML).

El cliente jamás se entera de las tareas ni de los códigos del lado del servidor.

Las operaciones que pueden realizarse en el server side por que requieren el acceso a la información o a alguna funcionalidad que no está disponible en el client side, o simplemente requiera de un comportamiento que es necesario validar o realizar en el server side ya que el client side no es del todo seguro.

Otras operaciones del server side también incluyen el proceso y almacenamiento de de datos.

Por ejemplo almacenar información en una base de datos.

Esta operación requiere que los datos que se van a enviar estén con el formato y tipo correcto que se espera en la base de datos, al igual que su distribución de los datos en las tablas correspondientes.

Ejemplo de ejecución de un código en ASP.NET del lado del servidor

En el editor de páginas de Visual Studio se escribe el siguiente código

```
1  <%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inhe
2
3  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
4
5  <html xmlns="http://www.w3.org/1999/xhtml">
6  <head runat="server">
7      <title>Untitled Page</title>
8  </head>
9  <body>
10     <form id="form1" runat="server">
11     <div>
12
13     </div>
14     </form>
15     <p>
16         Hola, esta es mi primera página Web en este entorno...</p>
17         Hoy es <% =Now%>
18 </body>
19 </html>
20
```

Figura 6.-Ejemplo de codigo

Una vez escrito se compila y se ejecuta la página.

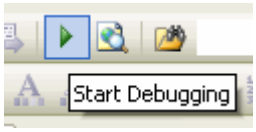

Figura 7.- Se muestra el botón con el que se corre y ejecuta la página.

Y el resultado es:

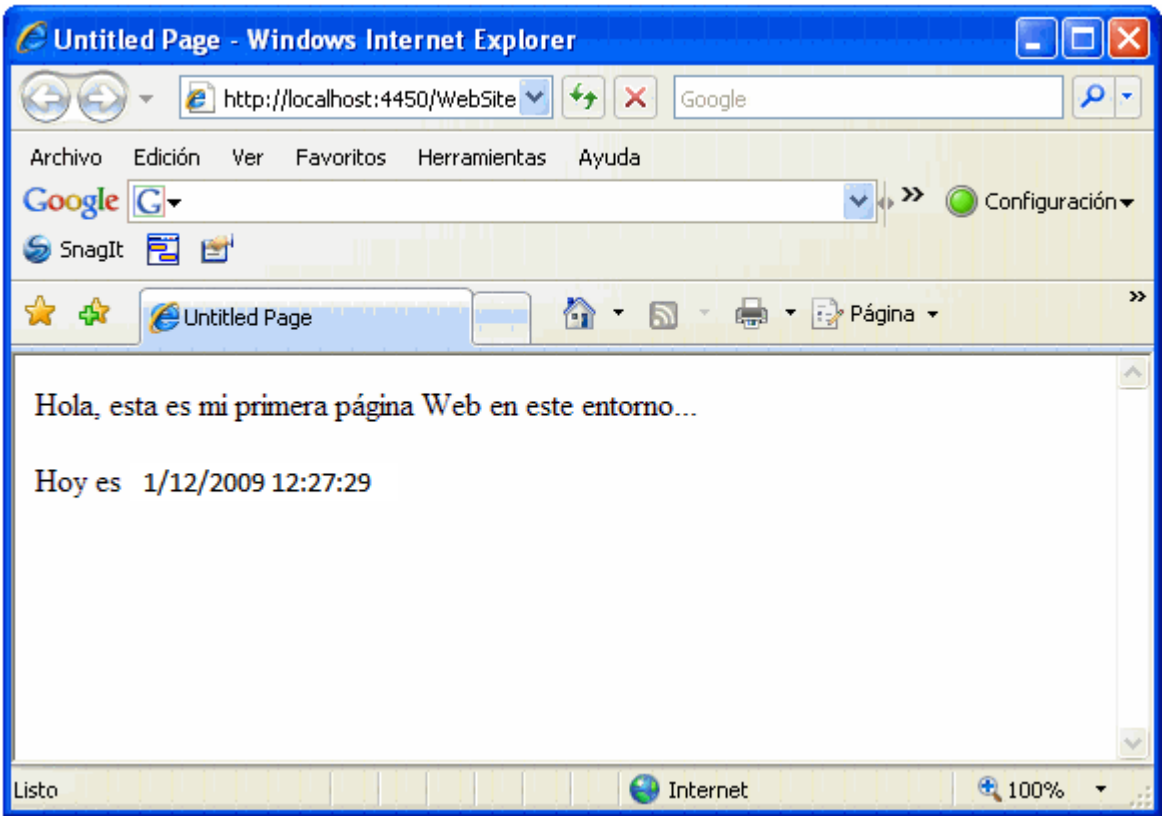

Figura 8.- Resultado del ejemplo.

¿Qué pasa?

"Now" es una función que devuelve la fecha y hora del sistema.

Nuestro servidor web ha ejecutado la página, esto es, ha localizado el código ASP.NET que hemos insertado y lo ha ejecutado antes de devolver la página. Como en este caso era una simple llamada a la función Now, le hemos dicho con ese código que devuelva el resultado de ejecutar esa función, y por tanto, la fecha y hora que ves en pantalla. Esto es la ejecución del lado del servidor, o lo que es lo mismo, la ejecución de una página por el servidor de páginas web. Si hubiésemos ido por el explorador de archivos y le hubiésemos dicho que abriera esa página, no habría funcionado porque debe pasar por el servidor web para poder hacer esa ejecución.

El otro tipo de ejecución es la del "lado del cliente", es decir código que está dentro de la página pero que se ejecuta en el Internet Explorer. Este código es muy distinto ya que solo hace pequeñas operaciones sobre el código de la página. Este código se suele escribir en Java o en Javascript y

no tiene comparación en cuanto a potencia porque no tenemos a nuestra disposición todas las funciones de ASP.NET ejecutándose en el servidor.

Base de datos (BD)

Una base de datos es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Diferentes programas y diferentes usuarios deben poder utilizar estos datos. Por lo tanto, el concepto de base de datos generalmente está relacionado con el de red ya que se debe poder compartir esta información. De allí el término **base**. "Sistema de información" es el término general utilizado para la estructura global que incluye todos los mecanismos para compartir datos que se han instalado.

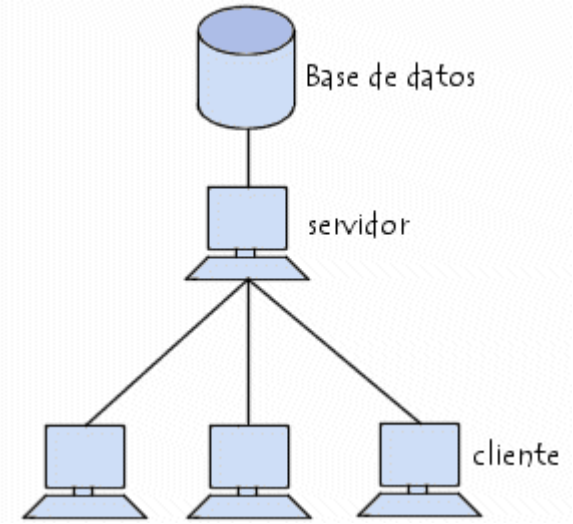

Figura 9.- Ejemplo del acceso de varios clientes a una base de datos.

Importancia de una BD

Una base de datos proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado. Se convierte más útil a medida que la cantidad de datos almacenados crece.

Una base de datos puede ser local, es decir que puede utilizarla sólo un usuario en un equipo, o puede ser distribuida, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red.

La principal ventaja de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo.

Administración de bases de datos

Rápidamente surgió la necesidad de contar con un sistema de administración para controlar tanto los datos como los usuarios. La administración de bases de datos se realiza con un sistema llamado **DBMS** (Database management system [Sistema de administración de bases de datos]). El DBMS es un conjunto de servicios (aplicaciones de software) para administrar bases de datos, que permite:

- un fácil acceso a los datos
- el acceso a la información por parte de múltiples usuarios
- la manipulación de los datos encontrados en la base de datos (insertar, eliminar, editar)

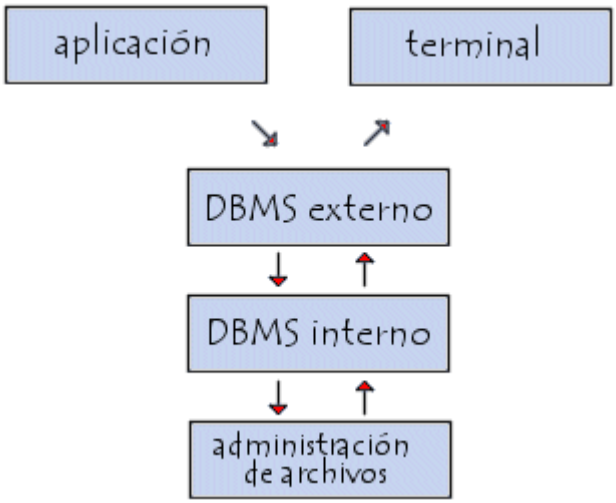

Figura 10.- Ejemplo de la administración de base de datos.

Características de un DBMS

La arquitectura de tres niveles definida por el modelo ANSI/SPARC mantiene los datos y el procesamiento separados. Hablando en general, un DBMS debe tener las siguientes características:

- **Independencia física**: el nivel físico puede ser modificado independientemente del nivel conceptual. Esto significa que el usuario no puede ver todos los componentes de hardware de la base de datos, que es simplemente una estructura transparente para representar la información almacenada.

- **Independencia lógica**: el nivel conceptual debe poder modificarse sin alterar el nivel físico. En otras palabras, el administrador de la base de datos debe poder introducir mejoras sin afectar la experiencia de los usuarios.

- **Facilidad de uso**: las personas que no estén familiarizadas con la base de datos deben poder describir su consulta sin hacer referencia a los componentes técnicos de la base de datos.

- **Acceso rápido**: el sistema debe poder responder a las consultas lo más rápido posible. Esto requiere algoritmos de búsqueda rápidos.

- **Administración centralizada**: el DBMS debe permitirle al administrador manipular los datos, agregar elementos y verificar su integridad de manera centralizada.

- **Redundancia controlada**: el DBMS debe poder evitar la redundancia de datos siempre que sea posible, tanto para minimizar los errores como para prevenir el desperdicio de memoria.

- **Verificación de integridad**: los datos deben ser internamente coherentes y, cuando algunos elementos hacen referencia a otros, estos últimos deben estar presentes.

- **Uso compartido de datos**: el DBMS debe permitir que múltiples usuarios accedan simultáneamente a la base de datos.

- **Seguridad de los datos**: el DBMS debe poder administrar los derechos de acceso a los datos de cada usuario.

- **Abstracción de la información**. los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos

de archivos, este hecho se hace transparente al usuario. Así, se definen varios *niveles de abstracción*.

- **Consistencia**. en aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.

- **Manejo de transacciones**. una Transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que luego de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

Modelos de DBMS

Existen cinco modelos de DBMS, que se distinguen según cómo representan los datos almacenados:

- **El modelo jerárquico**: los datos se organizan jerárquicamente mediante un árbol invertido. Este modelo utiliza punteros para navegar por los datos almacenados. Fue el primer modelo DBMS.

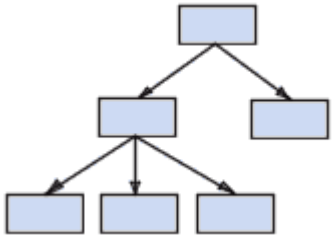

Figura 11.- Ejemplo modelo jerárquico.

- **El modelo de red**: al igual que el modelo jerárquico, este modelo utiliza punteros hacia los datos almacenados. Sin embargo, no necesariamente utiliza una estructura de árbol invertido.

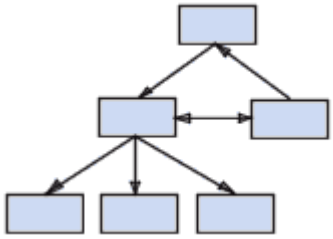

Figura 12.- Ejemplo modelo de red.

- **El modelo relacional** (**RDBMS**, *Relational database management system [Sistema de administración de bases de datos relacionales]*): los datos se almacenan en tablas de dos dimensiones (filas y columnas). Los datos se manipulan según la teoría relacional de

matemáticas.

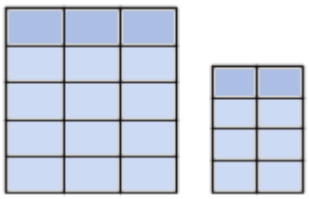

Figura 13.- Ejemplo modelo relacional.

- **El modelo de orientación a objetos** (**ODBMS**, *object-oriented database management system [sistema de administración de bases de datos orientadas a objetos]*): los datos se almacenan como objetos, que son estructuras denominadas *clases* que muestran los datos que contienen. Los campos son instancias de estas clases

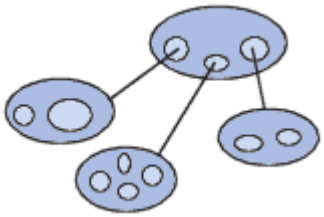

Figura 14.- Ejemplo modelo de orientación a objetos.

Ejemplos de manejadores de base de datos

	**Sybase**	www.sybase.com	**Adaptive Server**
	**Oracle**	www.oracle.com	**Oracle8, Oracle8i, Oracle8iEE, Oracle9i**
	**Microsoft**	www.microsoft.com	**Access, MS-SQL Server**
	**PostgreSQL**	www.postgresql.org	**PostgreSQL**
	**MySQL**	www.mysql.com	**MySQL**
	**Informix**	www.informix.com	**Illustra, Universal Server, Dynamic Server**
	**IBM**	www.ibm.com	**DB2**
	Firebird	http://firebird.sourceforge.net	Firebird

Tecnologías de desarrollo de aplicaciones web

Aplicaciones CGI

CGI es una norma para establecer comunicación entre un servidor web y un programa, de tal modo que este último pueda interactuar con internet. También se usa la palabra CGI para referirse al programa mismo, aunque lo correcto debería ser script. De todos modos en este tutorial nos interesa aprender a escribir estos programas por lo cual usaremos la palabra indistintamente, como es costumbre en castellano.

Un CGI (Common Gateway Interface) es un programa que se ejecuta en tiempo real en un Web Server en respuesta a una solicitud de un Browser. Cuando esto sucede el Web Server ejecuta un proceso hijo que recibirá los datos que envía el usuario (en caso de que los haya), pone a deposición del mismo algunos datos en forma de variables de ambiente y captura la salida del programa para enviarlo como respuesta al Browser.
El propósito de los CGI es proveer "inteligencia" e interactividad a un sitio web, por ejemplo encontrar un sitio en **Yahoo** utilizando solo los links que se proveen puede ser una labor frustrante, sin embargo usar el formulario y solicitar una búsqueda personalizada suele frustrarnos (un poco) menos, ya que un CGI nos provee de una respuesta hecha a la medida (eso dice la teoría) de nuestra consulta.

La interfaz define una forma cómoda y simple de ejecutar programas que se encuentran en la máquina en la que se aloja el servidor. Para el cliente presenta una ventaja en el aspecto de la seguridad, ya que no tendrá que ejecutar ningún programa de efectos desconocidos en su sistema local. Además de eliminar la necesidad de aprendizaje, se resuelven los problemas de mantenimiento, operación y distribución de clientes ya que el acceso se realizará a través de cualquier cliente estándar de WWW y la comunicación se realizará según el protocolo HTTP.

Será necesario tener en cuenta algunos aspectos sobre seguridad, ya que tener un programa CGI equivale a que "TODO EL MUNDO podrá ejecutar un programa en MI sistema". Esto se solventa incluyendo los programas CGI en un directorio determinado, denominado generalmente *cgi-bin*, directorio que no suele ser accesible a todos los usuarios. Cuando un servidor de WWW recibe como petición un documento alojado en este directorio, entenderá que se trata de un programa ejecutable.

Ahora vamos a analizar más detalladamente cómo funcionan las CGI y la relación entre cliente-servidor.

- Normalmente cuando en nuestro browser indicamos un URL (Uniform Resource Locator) pasa lo siguiente:
- Nuestra computadora toma contacto con el servidor HTTP utilizando el URL que indicamos nosotros;
- El servidor HTTP busca el nombre del archivo solicitado por nuestro ordenador y, cuando lo encuentra, nos devuelve el archivo;

- Por último, nuestro ordenador, recibido el archivo, lo elabora y lo muestra en la pantalla nueve veces de diez como una normal página web (es decir en formato HTML).

En algunos casos es posible configurar el servidor HTTP de forma que éste, cada vez que se solicita un archivo en un determinado directorio, no devuelva directamente el archivo. Este archivo se pone en marcha como si fuera un verdadero programa del servidor y sólo el output de este programa del servidor se envía a nuestro ordenador para que lo visualice. Un ejemplo evidente puede ser el contador. El ordenador que entra en el sitio donde está el contador no hace nada sino visualizar la gif preparada por el servidor (siempre que sea un contador gráfico), que lleva a cabo todo el trabajo de localizar el visitador para preparar una imagen con el número de los accesos. Ésta es precisamente la función que se llama Common Gateway Interface. Y estos programas se llaman script CGI.

Lenguajes de programación CGI

CGI, como la propia palabra indica es una interfaz entre servidores de información y programas de aplicación. Por tanto, define una serie de reglas que deben cumplir tanto las aplicaciones como los servidores para hacer posible la presentación de resultados de programas ejecutables en tiempo real a través de servicios de información estandarizados. Por ello, se habla de *gateway* o pasarela entre una y otra dimensión. Al tratarse de una interfaz, no existe ningún tipo de dependencia con el lenguaje de programación empleado.

En principio, cualquier lenguaje es susceptible de ser utilizado para desarrollar programas CGI, ya sea interpretado o compilado. En "la red" existen aplicaciones CGI desarrolladas en C, C++, Fortran, Perl, Tcl, Visual Basic, AppleScript y cualquier shell de UNIX. Los lenguajes interpretados como sh, tcl y, sobre todo, perl, tienen mayor facilidad de mantenimiento y depuración. Otra ventaja es que, por lo general, suelen ser de más alto nivel, con lo que no permiten realizar ciertas maniobras con la memoria y es más difícil dejar procesos sueltos descontrolados en el sistema. Uno de los lenguajes más utilizados es el Perl que, siendo interpretado, proporciona una potencia similar a la de C con una sintaxis muy parecida.

La contrapartida es que un lenguaje compilado es siempre mucho más rápido que uno interpretado. En el caso de CGI la velocidad de ejecución es importante, ya que habrá que sumar el tiempo de ejecución a la latencia de red y a la velocidad de transmisión, tiempos de espera que tendrá que sufrir el usuario del cliente.

ISAPI (Internet Server Application Programming Interface)

Muchos servidores de WWW proveen interfaces para la programación de aplicaciones (*application program interfaces*, *API*s) que brindan funcionalidad similar a la de aplicaciones *CGI*, pero con mayor eficiencia y más posibilidades. Como ejemplo de estas *API*s se puede mencionar *Isapi* (*Internet Server API*) de *Microsoft* y *Nsapi* de *Netscape*.

A diferencia de las aplicaciones *CGI*, las *ISA*s (aplicaciones *Isapi*) se ejecutan en el espacio de direcciones del servidor http y tienen acceso a todos los recursos disponibles. Las *ISA*s tienen menor sobrecarga ya que no requieren la creación de tareas adicionales y no consumen tiempo en la comunicación con otros procesos. Cada llamada *CGI* lanza un nuevo proceso, lo que conlleva cargas y descargas de información, mientras que en las *ISA*s la carga ocurre una sola vez, y son liberadas sólo si otro proceso necesita la memoria.

Una aplicación ISAPI es una DLL de Windows que se ejecuta en el mismo espacio de direcciones que el servidor de web. Estas aplicaciones pueden soportar las peticiones simultáneas de diversos clientes con una sola imagen en memoria.

Pero esta tecnología no se libra de sus contras. La creación de estas aplicaciones es costosa por su complejidad técnica, a la vez que las compilaciones pertinentes se han de hacer en servidor de web. Además las pruebas no se pueden realizar más que en una máquina que esté dando servicios de red.

Usando las llamadas a funciones ISAPI, las páginas web pueden invocar programas que están escritos en DLLs en el servidor, por ejemplo para acceder a datos en una base de datos. El IIS viene con un DLL que permite consultas integradas para acceder a bases de datos obedientes a ODBC.

ISAPI es una alternativa al uso de scripts CGI en servidores web de Microsoft.

La contraparte de ISAPI en el lado del cliente es WinInet.

ISAPI también ha sido implementado en Apache (con el módulo mod_isapi), por lo tanto, las aplicaciones web del lado del servidor escritas para IIS, también pueden ser usadas con Apache.

Algunas características de ISAPI

Las aplicaciones web ISAPI se ejecutan más rápidamente comparadas con las aplicaciones PHP y ASP. Pero una desventaja de ISAPI, es que generalmente se requiere más tiempo de desarrollo de éstas aplicaciones; además, carece de soporte nativo para muchas características estándares de aplicaciones web (como manejo de sesiones), que son características comunes en tecnologías como ASP y PHP.

ISAPI consiste de dos componentes: extensiones y filtros. Estos son los dos únicos tipos de aplicaciones que pueden ser desarrolladas usando ISAPI. Ambas deben ser escritas en C++ y Delphi Pascal, y compiladas en archivos DLL, que luego son registradas en IIS para ser ejecutadas en el servidor web.

¿Qué es una extensión de servidor de Internet?

Una extensión de servidor ISAPI es un archivo DLL que puede cargarse y ser llamado por un servidor HTTP. Las extensiones de servidor de Internet, también conocidas como aplicaciones de servidor de Internet (ISA), se utilizan para ampliar las capacidades de un servidor compatible con la API de servidor de Internet (ISAPI). Las aplicaciones ISA se invocan desde un explorador Web y proporcionan funcionalidad similar a una aplicación CGI (Common Gateway Interface). Para obtener información, vea los artículos Extender servidores Web mediante ISAPI y Extensiones y filtros de servidor ISAPI.

Ventajas de las extensiones de servidor ISAPI

Los usuarios pueden rellenar formularios y hacer clic en un botón "Enviar" para enviar datos a un servidor Web e invocar la aplicación ISA, que es capaz de procesar la información para proporcionar contenido personalizado o almacenarlo en una base de datos. Las extensiones de servidor Web pueden usar la información de una base de datos para generar páginas Web dinámicamente y después enviarlas a los equipos cliente para su presentación. Su aplicación puede añadir funcionalidad personalizada y proporcionar datos al cliente mediante HTTP y HTML.

Tanto las extensiones como los filtros de servidor se ejecutan en el espacio de procesos del servidor Web, proporcionando una forma eficiente de extender la capacidad del servidor.

Qué es un filtro ISAPI

Un filtro ISAPI es un archivo DLL que se ejecuta en un servidor HTTP habilitado para ISAPI con objeto de filtrar los datos que viajan hacia o desde el servidor. El filtro registra notificaciones sobre eventos, como el inicio de sesión o la asignación de direcciones URL. Cuando se producen los eventos seleccionados, se llama al filtro y es posible supervisar y cambiar los datos (en su recorrido entre el servidor y el cliente, y al contrario). Los filtros ISAPI pueden utilizarse para proporcionar un registro mejorado de las solicitudes HTTP (por ejemplo, para controlar quién inicia sesión en el servidor), cifrado y compresión de archivos personalizados o métodos de autenticación adicionales. Para obtener más información, vea Filtros ISAPI.

¿Qué tipo de servidor HTTP se necesita para ejecutar ISAPI?

Para alojar sitios Web debe disponerse de un servidor de Internet compatible con el protocolo HTTP (Hypertext Transfer Protocol). Si eligió un servidor Web compatible con ISAPI (por ejemplo, Servicios de Internet Information Server), podrá aprovechar los archivos DLL de extensión de servidor para crear aplicaciones rápidas y de pequeño tamaño para servidores de Internet.

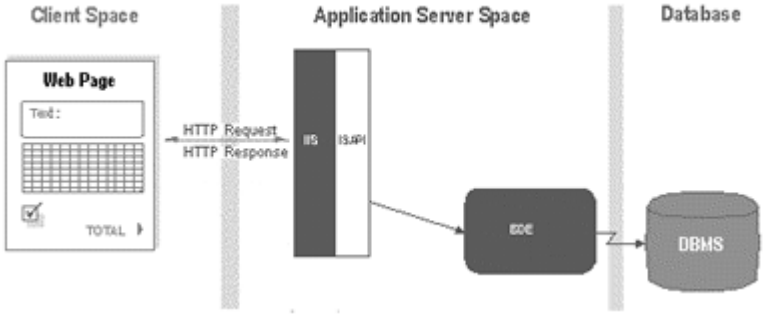

Figura 15.- Ejemplo servidor web.

Java Servlets y JSP

¿Qué son los Servlets Java?

Los Servlets son las respuestas de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y construyen páginas Web. Construir páginas Web al vuelo es útil (y comúnmente usado) por un número de razones:

- **La página Web está basada en datos enviados por el usuario**. Por ejemplo, las páginas de resultados de los motores de búsqueda se generan de esta forma, y los programas que procesan pedidos desde sites de comercio electrónico también.
- **Los datos cambian frecuentemente**. Por ejemplo, un informe sobre el tiempo o páginas de cabeceras de noticias podrían construir la página dinámicamente, quizás devolviendo una página previamente construida y luego actualizándola.
- **Las páginas Web que usan información desde bases de datos corporativas u otras fuentes**. Por ejemplo, usaríamos esto para hacer una página Web en una tienda on-line que liste los precios actuales y el número de artículos en stock.

Utilizar Servlets en lugar de Scripts CGI!

Los Servlets son un reemplazo efectivo para los scripts CGI. Proporcionan una forma de generar documentos dinámicos que son fáciles de escribir y rápidos en ejecutarse. Los Servlets también solucionan el problema de hacer la programación del lado del servidor con APIs específicos de la plataforma: están desarrollados con el API Java Servlet, una extensión estándar de Java.

Por eso se utilizan los Servlets para manejar peticiones de cliente HTTP. Por ejemplo, tener un servlet procesando datos Posteados sobre HTTP utilizando un formulario HTML, incluyendo datos del pedido o de la tarjeta de crédito. Un servlet como este podría ser parte de un sistema de procesamiento de pedidos, trabajando con bases de datos de productos e inventarios, y quizás un sistema de pago on-line.

. Otros usos de los Servlets

- Permitir la colaboración entre la gente. Un servlet puede manejar múltiples peticiones concurrentes, y puede sincronizarlas. Esto permite a los Servlets soportar sistemas como conferencias on-line
- Reenviar peticiones. Los Servlets pueden reenviar peticiones a otros servidores y Servlets. Con esto los Servlets pueden ser utilizados para cargar balances desde varios servidores que reflejan el mismo contenido, y para particional un único servicio lógico en varios servidores, de acuerdo con los tipos de tareas o la organización compartida

¿Cuáles son las ventajas de los Servlets sobre el CGI tradicional?

Los Servlets Java son más eficientes, fáciles de usar, más poderosos, más portables, y más baratos que el CGI tradicional y otras muchas tecnologías del tipo CGI.

- **Eficiencia**. Con CGI tradicional, se arranca un nuevo proceso para cada solicitud HTTP. Si el programa CGI hace una operación relativamente rápida, la sobrecarga del proceso de arrancada puede dominar el tiempo de ejecución. Con los Servlets, la máquina Virtual Java permanece arrancada, y cada petición es manejada por un thread Java de peso ligero, no un pesado proceso del sistema operativo. De forma similar, en CGI tradicional, si hay N peticiones simultáneas para el mismo programa CGI, el código de este problema se cargará N veces en memoria. Sin embargo, con los Servlets, hay N threads pero sólo una copia de la clase Servlet. Los Servlet también tienen más alternativas que los programas normales CGI para optimizaciones como los cachés de cálculos previos, mantener abiertas las conexiones de bases de datos, etc.

- **Conveniencia**. Junto con la conveniencia de poder utilizar un lenguaje familiar, los Servlets tienen una gran infraestructura para análisis automático y decodificación de datos de formularios HTML, leer y seleccionar cabeceras HTTP, manejar cookies, seguimiento de sesiones, y muchas otras utilidades.

- **Potencia**. Los Servlets Java nos permiten fácilmente hacer muchas cosas que son difíciles o imposibles con CGI normal. Por algo, los Servlets pueden hablar directamente con el servidor Web. Esto simplifica las operaciones que se necesitan para buscar imágenes y otros datos almacenados en situaciones estándares. Los Servlets también pueden compartir los datos entre ellos, haciendo las cosas útiles como almacenes de conexiones a bases de datos fáciles de implementar.

También pueden mantener información de solicitud en solicitud, simplificando cosas como seguimiento de sesión y el caché de cálculos anteriores.

- **Portable**. Los Servlets están escritos en Java y siguen un API bien estandarizado. Consecuentemente, los Servlets escritos, digamos en el servidor I-Planet Enterprise, se pueden ejecutar sin modificarse en Apache, Microsoft IIS, o WebStar. Los Servlets están soportados directamente o mediante plug-in en la mayoría de los servidores Web.

- **Barato**. Hay un número de servidores Web gratuitos o muy baratos que son buenos para el uso "personal" o el uso en sites Web de bajo nivel. Sin embargo, con la excepción de Apache, que es gratuito, la mayoría de los servidores Web comerciales son relativamente caros. Una vez que tengamos un servidor Web, no importa el coste del servidor, añadirle soporte para Servlets (si no viene pre configurado para soportarlos) es gratuito o muy barato.

Ejemplo de servlet

```
<servlet>
  <servlet-name>FormClientes</servlet-name>
  <servlet-class>docen_servlet01.JDBC01.presentacion.FormClientes</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>FormVentas</servlet-name>
  <servlet-class>docen_servlet01.JDBC01.presentacion.FormVentas</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>FormClientes</servlet-name>
  <url-pattern>/servlet/FormClientes</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>FormVentas</servlet-name>
  <url-pattern>/servlet/FormVentas</url-pattern>
 </servlet-mapping>
```

¿Qué es JSP?

Java Server Pages (JSP) es una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente. Muchas páginas Web que están construidas con programas CGI son casi estáticas, con la parte dinámica limitada a muy pocas localizaciones. Pero muchas variaciones CGI, incluyendo los Servlets, hacen que generemos la página completa mediante nuestro programa, incluso aunque la mayoría de ella sea siempre lo mismo. JSP nos permite crear dos partes de forma separada. Aquí tenemos un ejemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>

<HEAD><TITLE>Welcome to Our Store</TITLE></HEAD>

<BODY>

<H1>Welcome to Our Store</H1>

<SMALL>Welcome,

<!-- User name is "New User" for first-time visitors -->

<% out.println(Utils.getUserNameFromCookie(request)); %>

To access your account settings, click

<A HREF="Account-Settings.html">here.</A></SMALL>

<P>

Regular HTML for all the rest of the on-line store's Web page.

</BODY></HTML>
```

¿Cuáles son las Ventajas de JSP?

- **Contra Active Server Pages (ASP).** ASP es una tecnología similar de Microsoft. Las ventajas de JSP están duplicadas. Primero, la parte dinámica está escrita en Java, no en Visual Basic, otro lenguaje específico de MS, por eso es mucho más poderosa y fácil de usar. Segundo, es portable a otros sistemas operativos y servidores Web.

- **Contra los Servlets**. JSP no nos da nada que no pudiéramos en principio hacer con un servlet. Pero es mucho más conveniente escribir (y modificar!) HTML normal que tener que hacer un billón de sentencias println que generen HTML. Además, separando el formato del contenido podemos poner diferentes personas en diferentes tareas: nuestros expertos en diseño de páginas Web pueden construir el HTML, dejando espacio para que nuestros programadores de Servlets inserten el contenido dinámico.

- **Contra Server-Side Includes (SSI)**. SSI es una tecnología ampliamente soportada que incluye piezas definidas externamente dentro de una página Web estática. JSP es mejor porque nos permite usar Servlets en vez de un programa separado para generar las partes dinámicas. Además, SSI, realmente está diseñado para inclusiones sencillas, no para programas "reales" que usen formularios de datos, hagan conexiones a bases de datos, etc.

- **Contra JavaScript**. JavaScript puede general HTML dinámicamente en el cliente. Este una capacidad útil, pero sólo maneja situaciones donde la información dinámica está basada en el entorno del cliente. Con la excepción de las cookies, el HTTP y el envió de formularios no están disponibles con JavaScript. Y, como se ejecuta en el cliente, JavaScript no puede acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etc.

¿Los JSPs son en realidad Servlets?

Un JSP se compila a un programa en Java la primera vez que se invoca, y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet. La principal diferencia entre los Servlets y los JSPs es el enfoque de la programación: un JSP es una página Web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web

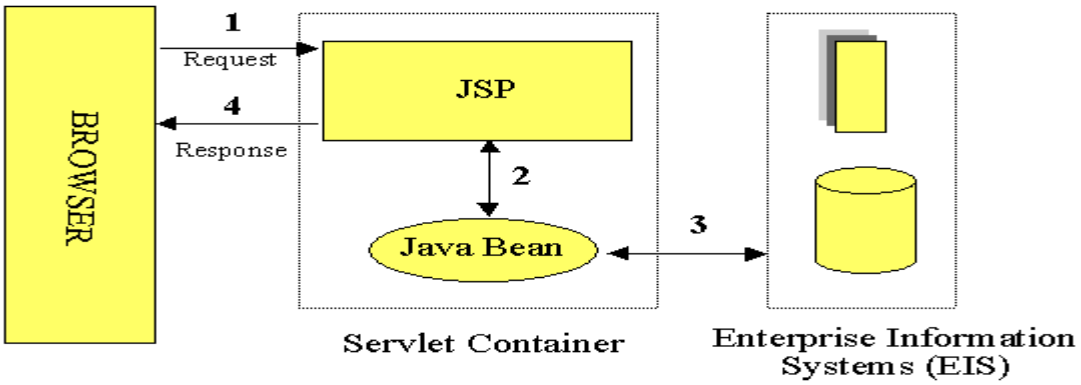

Figura 16.- Ejemplo JSP

Ambos necesitan un programa que los contenga, y sea el que envíe efectivamente páginas web al servidor, y reciba las peticiones, las distribuya entre los Servlets, y lleve a cabo todas las tareas de gestión propias de un servidor web. Mientras que servidores como el Apache están especialmente pensados para páginas web estáticas CGIs, y programas ejecutados por el servidor, tales como el PHP, hay otros servidores específicos para Servlets y JSPs llamados *contenedores de Servlets (servlet containers)* o *servlet engines*. Los principales son los siguientes:

- Resin, de Caucho Technologies, un motor especialmente enfocado al servicio de páginas XML, con una licencia libre para desarrolladores. Dice ser bastante rápido. Incluye soporte para Javascript además de Java. Incluye también un lenguaje de templates llamado XTP. Es bastante fácil de instalar, y en dos minutos, se pueden empezar a servir páginas JSP.

- BEA Weblogic es un servidor de aplicaciones de alto nivel, y también de alto precio. Está escrito íntegramente en Java, y se combina con otra serie de productos, tales como Tuxedo, un servidor de bases de datos para XML.

- JRun, de Macromedia, un servidor de aplicaciones de Java, de precio medio y probablemente prestaciones medias. Se puede bajar una versión de evaluación gratuita

- Lutris Enhydra, otro servidor gratuito y Open Source, aunque tiene una versión de pago. También enfocado a servir XML, y para plataformas móviles. Las versiones más actualizadas son de pago, como es natural

- El más popular, Open Source, y continuamente en desarrollo, es el Jakarta Tomcat, del consorcio Apache, un contenedor de Servlets con muchos desarrollos adicionales alrededor; por ejemplo, Cocoon para servir páginas XML.

Puede servir páginas sólo o bien como un añadido al servidor Apache. Es Open Source, relativamente rápido, y fácil de instalar.

Tecnologías de Microsoft

ASP

Al navegar más de alguna vez nos hemos topado con alguna página que tiene archivos con extensión ".asp" y nos hemos preguntado qué significa éste tipo de archivos.

Microsoft introdujo esta tecnología llamada Active Server Pages en diciembre de 1996, por lo que no es nada nueva. Es parte del Internet Information Server (IIS) desde la versión 3.0 y es una tecnología de páginas activas que permite el uso de diferentes scripts y componentes en conjunto con el tradicional HTML para mostrar páginas generadas dinámicamente, traduciendo la definición de Microsoft: *"Las Active Server Pages son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el web"*.

Los orígenes de la tecnología ASP es el VBScript, pero existe otra diversidad de lenguajes de programación que pueden ser utilizados como lo es Perl, JavaScript, etc.

El ASP es una tecnología dinámica funcionando del lado del servidor, lo que significa que cuando el usuario solicita un documento ASP, las instrucciones de programación dentro del script son ejecutadas para enviar al navegador únicamente el código HTML resultante. La ventaja principal de las tecnologías dependientes del servidor radica en la seguridad que tiene el programador sobre su código, ya que éste se encuentra únicamente en los archivos del servidor que al ser solicitado a través del web, es ejecutado, por lo que los usuario no tienen acceso más que a la página resultante en su navegador.

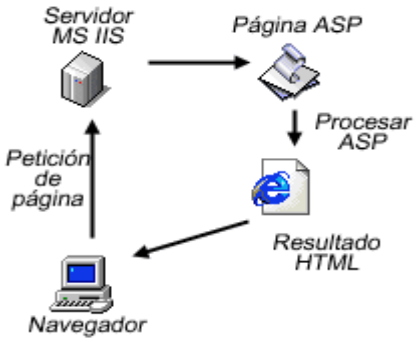

Figura 17.- Comunicación para ejecutar una página de ASP.

ASP es una tecnología que pertenece a la parte servidor, por esto no es necesario que el cliente o navegador la soporte ya que se ejecuta en el servidor, sí que deberemos buscar un servidor que nos soporte este tipo de tecnología para que nuestras páginas corran correctamente.

Hay que destacar que ASP es una tecnología propietaria de Microsoft, y que el uso de esta tecnología implica el uso de los productos de Microsoft: Microsoft Internet Information System y Microsoft Windows en el servidor.

Versiones de ASP

Historial de versiones de ASP:
• ASP versión 1.0 (diciembre de 1996).
• ASP versión 2.0 (septiembre de 1997).
• ASP versión 3.0 (noviembre de 2000)

Ejemplo: las páginas pueden ser generadas mezclando código de scripts del lado del servidor (incluyendo acceso a base de datos) con HTML.

Ejemplo1 (Hola Mundo):

```
1.<html>

2.<body>

3.<% Response.Write ("Hola Mundo") %>

4.</body>

5.</html>
```

Este código muestra en pantalla la frase: "Hola Mundo"

Ejemplo2:

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>

(...)

<!--#include virtual="/Conexion a la base de datos.asp" -->

(...)

<p>Deja un Comentario

<%If
(CStr(Recordset1.Fields.Item("ValorX").Value))=(CStr(Recordset2.Fields.Item("ValorY").Value))
Then%>

<strong><%=(Recordset2_total)%></strong>

<%else%>

<strong>0</strong>

<%end if%>

 Comentarios</p>
```

Este código trae como resultado en HTML el valor de un registro de una base de datos: cuando ValorX es igual a ValorY se nos muestra el número total de registros de una base de datos (previa implementación de los correspondientes recordsets).

ASP.NET

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Cualquier persona que este familiarizada con el desarrollo de aplicaciones web sabrá que el desarrollo web no es una tarea simple. Ya que mientras que un modelo de programación para aplicaciones de uso común está muy bien establecido y soportado por un gran número de lenguajes, herramientas de desarrollo, la programación web es una mezcla de varios lenguajes de etiquetas, un gran uso de lenguajes de script y plataformas de servidor. Por desgracia para el programador de nivel intermedio, el conocimiento y habilidades que se necesitan para desarrollar aplicaciones web tienen muy poco en común con las que son necesarias en el desarrollo tradicional de aplicaciones.

Diferencias con respecto a ASP

En el modelo de desarrollo web basado en páginas activas, la programación ASP actual tiene diversas limitaciones:

- Para que todo ocurra en una página web, es habitual escribir una gran cantidad de código para resolver necesidades sencillas. ASP.NET incorpora un modelo declarativo a la programación web: los controles de servidor funcionan en una página Web simplemente declarándolos. Cuando se carga la página ASP.NET, se instancian los controles listados en la página ASP y es responsabilidad del control emitir código HTML que el navegador pueda entender.

- ASP clásico es un tanto desorganizado. En una página ASP podemos incluir casi todo: HTML plano, código script, objetos COM y texto. No hay una distinción formal entre el contenido de una página y su comportamiento: simplemente, insertamos código en la página, y a ver qué pasa. ASP.NET impone un cierto orden sobre el modelo de programación estándar ASP. En cierto modo, esta "desorganización" puede evitarse fácilmente usando el sentido común y algunas de las nuevas tecnologías. Por ejemplo, podemos escribir en nuestras páginas ASP únicamente código VBScript. Dicho código generaría un mensaje XML, que luego sería interpretado por un archivo XSLT. De esta

forma conseguimos evitar el llamado "código spaguetti", aumentando la claridad del código y la velocidad de ejecución de las páginas ASP.

- La tercera limitación en el desarrollo con ASP es que con el tradicional utilizamos lenguajes de scripting no tipiados como VBScript o JScript. Podemos instalar otros motores de scripting que impongan verificación de tipos; sin embargo, no son universalmente conocidos o utilizados como los anteriores. ASP.NET claramente separa la porción basada en script de una página web de su contenido.

- ASP.Net, puede decirse que en nuevo nivel de abstracción en la construcción de sitios web, por que se pueden crear rápidamente aplicaciones web, basándose en los controles incluidos en el framework o muchos gratuitos que hay en la red, ocultando el código de autogenerado.

- Es muy sencilla la creación de páginas con AJAX, sólo incluyendo unos controles, así como descargar gratuitamente el ToolKit de ASP.Net Ajax.

¿Qué diferencias de sintaxis hay con respecto de ASP 3.0?

Partiendo de la base de que el lenguaje es otro, muchas, al ser programación orientada a objetos y al ser lenguaje de Visual Basic .net todo es muy diferente. Hay que cambiar el pensamiento y todos los viejos conceptos que teníamos de ASP 3.0.

ASP.net es más bien parecido a Visual Basic, en la página, todos los elementos son objetos, activos de servidor (que se generan ahí) y tienen propiedades y métodos.

Vale aclarar que en ASP.net existen los formularios activos del servidor ¿qué es esto?, son como los formularios tradicionales de HTML, pero se le agrega en el TAG FORM el atributo runat=server, y dentro del formulario se agregan los controles activos de servidor. Cuando ocurre un evento, la página se auto envía a sí misma y se procesa el controlador para ese evento.

Por ejemplo:

```
<%@ Page language="VB"%>

<%@ Import Namespace="System.data.oledb" %>

<script runat="server">

Sub Nombre_change(sender as object, e as EventArgs)

Mensaje.text = "Buenos días " + Nombre.text

End sub

</script>

<html><body>

<font face="verdana" size=2>Esta es la página de ejemplo</font><br><br>

<form runat="server">

Tu nombre: <asp:TextBox id="Nombre" OnTextChanged="Nombre_change"

        runat="server" autopostback="true"/>

<asp:label id="Mensaje" runat="server" />

</form>

</body></html>
```

Tal vez parezca un poco complicado al principio. Pero analicemos el código. Primero si quieren experimentar, copien y peguen el código en el bloc de notas y guárdenlo en su WWWROOT como bienvenida.aspx, tomen nota de que para correr ASP.net deben tener el .NET Framework instalado en su PC, de otra manera no funcionará. Vayan a http://localhost/bienvenida.aspx. Tecleen su nombre y sitúen el cursor fuera del casillero de texto, como por arte de magia aparece un mensaje que nos da buenos días. Pero no es magia en realidad, veamos cómo funciona realmente.

En el código hay tres colores marcados: verde, rojo y azul.

- En las declaraciones del color verde podemos ver que le dan dos instrucciones, la primera, dice al compilador que el lenguaje utilizado es VB.net, de otra manera diría "C#", nótese la ausencia del ".net". En la segunda instrucción, vemos que importa el espacio de nombre System.Data.OleDB, que sirve para trabajar con bases de datos, innecesario en esta página claro, pero lo incluí para que vean como se incluye un espacio de nombre. Un espacio de nombre es algo que te permite realizar determinada función, por ejemplo, si queremos utilizar las poderosas características que trae ASP.net para trabajar con XML, importaríamos el espacio de nombre "System.XML".

- En el color rojo podemos observar sólo el método Nombre_change, correspondiente al evento del campo de texto Nombre, o sea, se ejecuta cuando el campo de texto Nombre cambia. Lo único que hace es poner en la etiqueta Mensaje el valor "Buenos días " más el valor que contenga el campo de texto Nombre.

- El color azul es el conocido HTML, pero con un formulario con RunAt="Server", o sea, estos controles se ejecutan en el servidor y luego procesados traen código HTML común, o sea por ejemplo: <input name="Nombre" type="text" id="Nombre">

Una vez procesado todo el código, envía el siguiente resultado al navegador:

```
<html><body>
<font face="verdana" size=2> Esta es la página de ejemplo </font><br><br>
<form name="_ctl0" method="post" action="bienvenida.aspx" id="_ctl0">
<input type="hidden" name="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE"
    value="dDwxMTU3NzQ3OTc0Ozs+jxrUecWhDY6AanZbrFANP9MYypQ=" />
<script language="javascript">
<!--
function __doPostBack(eventTarget, eventArgument) {
var theform = document._ctl0;
theform.__EVENTTARGET.value = eventTarget;
theform.__EVENTARGUMENT.value = eventArgument;
theform.submit();
}
// -->
</script>

Tu nombre: <input name="Nombre" type="text" id="Nombre"
    onchange="__doPostBack('Nombre','')" language="javascript" />
<span id="Mensaje"></span>
</form>
</body></html>
```

Code-Behind

Microsoft recomienda que para realizar programación dinámica se use el modelo code-behind, o de respaldo, que coloca el código en un archivo separado o en una etiqueta de script especialmente diseñada. Los nombres de los archivos code-behind están basados en el nombre del archivo ASPX tales como MiPagina.aspx.cs o MiPagina.aspx.vb (esta práctica se realiza automáticamente en Microsoft Visual Studio y otras interfaces de desarrollo). Cuando se usa este estilo de programación, el desarrollador escribe el código correspondiente a diferentes eventos, como la carga de la página, o el clic en un control, en vez de un recorrido lineal a través del documento.

El modelo code-behind de ASP.NET marca la separación del ASP clásico y alienta a los desarrolladores a construir aplicaciones con la idea de presentación y contenido separados en mente. En teoría, esto permite a un diseñador web, por ejemplo, enfocarse en la creación del diseño con menos posibilidades de alterar el código de programación mientras lo hace. Esto es similar a la separación en el Modelo Vista Controlador

Estructura de directorios de ASP.NET

En general, la estructura de directorios de ASP.NET puede ser determinada por las preferencias del desarrollador. Aparte de unos pocos nombres de directorios reservados, el sitio puede expandirse a cualquier número de directorios. La estructura es típicamente reflejada directamente en las urls.

Los nombres de directorios especiales (a partir de ASP.NET 2.0 son):

App_Browsers

Contiene archivos de definición específicos para navegadores.

App_Code

Es un directorio para códigos. El servidor ASP.NET automáticamente compilara los archivos (y subdirectorios) en esta carpeta en un ensamblado que es accesible desde cualquier página del sitio. App_Code es típicamente usada para código de acceso a datos, código de modelo o código de negocios. También cualquier manejador http especifico para el sitio e implementación de módulos y servicios web van este directorio. Como alternativa a utilizar App_Code el desarrollador puede optar por proporcionar un ensamblado independiente con código pre compilado.

App_Data

Directorio por defecto para la base de datos, tales como archivos mdb de Microsoft Access y archivos mdf de Microsoft SQL Server. Este directorio es usualmente el único con permisos de escritura en la aplicación.

App_LocalResources

Contiene archivos de recursos localizados para páginas individuales del sitio.

App_GlobalResources

Contiene archivos resx con recursos localizados disponibles para cada página del sitio. Este es donde el desarrollador ASP.NET típicamente almacenara mensajes que serán usados en más de una página.

App_Themes

Usado para temas alternativos del sitio.

App_WebReferences

Usado para archivos de descubrimiento y archivos WSDL para referencias a servicios web para ser consumidos en el sitio.

Bin

Contiene código compilado (archivos .dll) para controles, componentes, y otro código que pueda ser referenciado por la aplicación. Cualquier clase representada por código en la carpeta Bin es automáticamente referenciada en la aplicación.

Uso Actual de ASP.NET

En la actualidad una aplicación ASP.NET puede ejecutarse de dos formas distintas:

Aplicaciones cliente/servidor: estas aplicaciones están típicamente en formato de ejecutables compilados. Estos pueden integrar toda la riqueza de una interfaz de usuario, tal es el caso de las aplicaciones de desempeño y productividad, pero no se reúne la lógica de negocio como un recurso que se pueda reutilizar. Además acostumbran ser menos gestionables y escalables que las demás aplicaciones.

Aplicaciones que utilizan el navegador: dichas aplicaciones están caracterizadas por contar con una interfaz de web rica y muy útil. La interfaz gráfica integra varias tecnologías, las cuales son el HTML, XHTML, scripting, etc.; siempre y cuando el navegador que se esté utilizando soporte estas tecnologías.

Versiones ASP.NET

- ASP.NET versión 1.0 .
- ASP.NET versión 1.1.
- ASP.NET versión 2.0.
- ASP.NET versión 3.0
- ASP.NET versión 3.5.

Compilador de ASP.NET

Compila todo el código de ASP.NET, lo que permite el establecimiento inflexible de tipos, las optimizaciones de rendimiento y el enlace en tiempo de compilación, entre otras ventajas. Una vez que se ha compilado el código, el Common Language Runtime compila una vez más código de ASP.NET en código nativo, lo que permite un mayor rendimiento.

ASP.NET incluye un compilador que compilará todas las componentes de la aplicación, incluidas las páginas y los controles, en un ensamblado que el entorno de host de ASP.NET puede utilizar a continuación para atender las solicitudes del usuario.

Infraestructura de seguridad

Además de las características de seguridad de .NET, ASP.NET proporciona una infraestructura de seguridad avanzada para autenticar y autorizar el acceso de los usuarios y realizar otras tareas relacionadas con la seguridad. Puede autenticar usuarios con la autenticación de Windows suministrada por IIS o puede administrar la autenticación con su propia base de datos de usuario utilizando la autenticación mediante formularios ASP.NET y la suscripción ASP.NET. Además, puede administrar la autorización a las capacidades e información de su aplicación Web mediante los grupos de Windows o su propia base de datos de funciones personalizada utilizando las funciones de ASP.NET. Resulta fácil quitar, agregar o reemplazar estos esquemas dependiendo de las necesidades de la aplicación.

ASP.NET siempre se ejecuta con una identidad particular de Windows de modo que puede asegurar su aplicación utilizando las capacidades de Windows como, por ejemplo, las listas de control de acceso (ACL) de NTFS, permisos de la base de datos, etc. Para obtener más información sobre la identidad de ASP.NET, vea Configurar la identidad de procesos en ASP.NET y Suplantación de ASP.NET.

Funciones de administración de estado

ASP.NET proporciona funcionalidad de administración de estado intrínseca que permite almacenar información entre las solicitudes de página, como la información de clientes o el contenido del carro de la compra. Puede guardar y administrar información específica de la aplicación, específica de la sesión, específica de la página, específica del usuario y definida por el desarrollador. Esta información puede ser independiente de cualquier control de la página,

ASP.NET ofrece funciones de estado distribuidas, lo que le permite administrar información de estado en múltiples instancias de la misma aplicación en un equipo o en varios.

Configuración de ASP.NET

Las aplicaciones ASP.NET utilizan un sistema de configuración que le permite definir valores de configuración para su servidor Web, para un sitio Web o para aplicaciones individuales. Puede crear valores de configuración cuando se implementan las aplicaciones ASP.NET y puede agregar o revisar los valores de configuración en cualquier momento con un impacto mínimo en aplicaciones y servidores Web de operaciones. Los valores de configuración de ASP.NET se almacenan en archivos basados en la tecnología XML. Dado que estos archivos XML son archivos de texto ASCII, es fácil realizar cambios de configuración a sus aplicaciones Web. Puede extender el esquema de configuración para satisfacer sus requisitos.

Supervisión de estado y características de rendimiento

ASP.NET incluye características que le permiten supervisar el estado y el rendimiento de su aplicación ASP.NET. La supervisión del estado de ASP.NET permite proporcionar información sobre eventos clave que proporcionan información sobre el estado de una aplicación y sobre las condiciones de error. Estos eventos muestran una combinación de diagnósticos y características de supervisión, a la vez que proporcionan un elevado grado de flexibilidad en lo que respecta a lo que se registra y cómo. Para obtener más información, vea Información general sobre la supervisión de estado en ASP.NET.

ASP.NET admite dos grupos de contadores de rendimiento a los que pueden obtener acceso las aplicaciones:

- El grupo de contadores de rendimiento del sistema ASP.NET
- El grupo de contadores de rendimiento de la aplicación ASP.NET

Capacidad de depuración

ASP.NET aprovecha la infraestructura de depuración en tiempo de ejecución para permitir la depuración entre lenguajes y equipos. Se pueden depurar tanto objetos administrados como no administrados, así como todos los lenguajes compatibles con el Common Language Runtime y los lenguajes de script. Para obtener información detallada, vea Depuración en ASP.NET.

Además, el marco de trabajo de páginas ASP.NET proporciona un modo de seguimiento que permite insertar mensajes de instrumentalización en las páginas Web ASP.NET.

Marco de trabajo de servicios Web XML

ASP.NET es compatible con los servicios Web XML. Un servicio Web XML es un componente que incluye funcionalidad de empresa que permite a las aplicaciones intercambiar información entre firewalls utilizando estándares como los servicios de mensajería HTTP y XML. Los servicios Web XML no están relacionados con ninguna tecnología de componentes ni con ninguna convención de llamada a objetos en concreto. Como resultado, pueden obtener acceso a los servicios Web XML los programas escritos en cualquier lenguaje, que usen cualquier modelo de componentes y se ejecuten en cualquier sistema operativo.

Entorno de host extensible y administración del ciclo de vida de las aplicaciones

ASP.NET incluye un entorno de host extensible que controla el ciclo de vida de una aplicación desde el momento en que un usuario cualquiera tiene acceso a un recurso (como una página) en la aplicación hasta el momento en que se cierra la aplicación. Aunque ASP.NET se basa en un servidor Web (IIS) como un host de la aplicación, ASP.NET proporciona gran parte de la propia funcionalidad de host. La arquitectura de ASP.NET permite responder a los eventos de aplicación y crear controladores y módulos HTTP personalizados.

Entorno de diseñador extensible

ASP.NET incluye la compatibilidad mejorada para crear diseñadores de controles de servidor Web para utilizarlos con una herramienta de diseño visual como Visual Studio. Los diseñadores permiten crear una interfaz de usuario en tiempo de diseño para un control; de este modo, los desarrolladores pueden configurar las propiedades y el contenido del control en una herramienta de diseño visual

Mejoras de ASP.NET

Las secciones siguientes proporcionan información sobre las mejoras y las nuevas características en Visual Studio 2008 y Visual Web Developer Express.

Datos dinámicos

Los datos dinámicos de ASP.NET son un marco de trabajo que permite crear fácilmente aplicaciones web ASP.NET controladas por datos. Para ello, detecta automáticamente el modelo de datos en tiempo de ejecución y determina el comportamiento de la interfaz de usuario a partir de dicho modelo. Un marco con una estructura pre construida que proporciona al instante un sitio web funcional para ver y editar datos. Dicha técnica podrá personalizarse con facilidad posteriormente mediante el uso de metadatos y plantillas, o mediante la creación de páginas ASP.NET estándar para invalidar el comportamiento predeterminado

Enrutamiento de direcciones URL

El enrutamiento de direcciones URL en ASP.NET permite usar direcciones URL que no es necesario asignar a archivos determinados en un sitio web. Dado que la dirección URL no tiene que asignarse a un archivo, en una aplicación web puede usar direcciones URL descriptivas de la acción del usuario, y por tanto, que los usuarios puedan identificar mejor. En el enrutamiento de direcciones URL, se definen modelos de dirección URL que contienen marcadores de posición

para los valores utilizados al administrar solicitudes URL. En tiempo de ejecución, las partes de la dirección URL que siguen al nombre de aplicación se analizan como valores discretos, tomando como base un modelo de dirección URL que haya definido.

Control EntityDataSource

El control EntityDataSource admite escenarios de enlace de datos basados en el Entity Data Model (EDM). La especificación EDM representa datos como conjuntos de entidades y relaciones. Entity Framework usa el EDM en asignaciones relacionales de objetos y otros escenarios como Servicios de datos de ADO.NET. Los usuarios acostumbrados al modelo en tiempo de diseño de los controles de enlace de datos de ASP.NET encontrarán la superficie de programación del control EntityDataSource similar a la de otros controles de origen de datos.

El control EntityDataSource administra las operaciones de creación, lectura, actualización y eliminación con un origen de datos procedentes de controles con enlace de datos en la página. El control EntityDataSource usa cuadrículas que se pueden editar, formularios con ordenación y filtrado controlados por el usuario, controles de lista desplegable con doble enlace y páginas principal-detalle. El control EntityDataSource puede obtener valores de parámetros de consulta a partir de controles de páginas, parámetros de consulta anexados al URI de la página, cookies, así como otros objetos de parámetros de ASP.NET.

Nuevas extensiones de ASP.NET AJAX

Las nuevas extensiones de ASP.NET AJAX proporcionan un mayor control del historial del explorador al usar el botón Atrás. Así mismo, permiten combinar automáticamente varios scripts de cliente en un script compuesto. De esta forma se reduce el tiempo de carga de scripts, ya que disminuye enormemente el número necesario de viajes de ida y vuelta (round trips) al servidor.

Formato JScript

Se ha extendido la funcionalidad de formato de código para proporcionar compatibilidad con JScript en Visual Studio y Visual Web Developer Express. Puede elegir dar formato al código manualmente en un documento o selección, o aplicar el formato automáticamente mientras escribe.

Mejoras de ASP.NET en la versión 3.5

.NET Framework versión 3.5 incluye mejoras para ASP.NET en las áreas siguientes:

- Nuevos controles de servidor, tipos y una biblioteca de scripts de cliente que funcionan juntos para permitir el desarrollo de aplicaciones web con estilo AJAX.
- Extensión de la autenticación de formularios basada en servidor, administración de funciones y servicios de perfil como servicios web que pueden usar las aplicaciones basadas en web.
- Un nuevo control EntityDataSource que expone el Entity Data Model a través de la arquitectura de controles de origen de datos ASP.NET.
- Un nuevo control de datos ListView que muestra datos y proporciona una interfaz de usuario con un alto grado de personalización.
- Un nuevo control LinqDataSource que expone Language-Integrated Query (LINQ) a través de la arquitectura de controles de origen de datos ASP.NET.
- Una nueva herramienta de combinación (Aspnet_merge.exe) que combina los ensamblados pre compilados para admitir la implementación flexible y la administración de lanzamientos. Esta característica no está disponible en Visual Web Developer Express.

.NET Framework versión 3.5 también se integra con IIS 7.0. Ahora puede usar servicios ASP.NET como la autenticación de formularios y el almacenamiento en caché para todos los tipos de contenido, no sólo páginas web ASP.NET (archivos .aspx). Esto se debe a que ASP.NET e IIS 7.0 usan la misma canalización de solicitudes. La canalización de procesamiento de solicitudes unificada implica que puede usar código administrado para desarrollar módulos de canalización HTTP que trabajen con todas las solicitudes en IIS. Además, los módulos y controladores IIS y ASP.NET admiten ahora la configuración unificada. Para obtener más información, consulte Información general sobre el ciclo de vida de una aplicación ASP.NET para IIS 7.0.

Desarrollo de AJAX

.NET Framework versión 3.5 permite crear aplicaciones web que representan interfaces de usuario de próxima generación con componentes de cliente reutilizables. Puede desarrollar las páginas web aplicando un enfoque basado en servidor, basado en cliente o una combinación de ambos, según sus requisitos. Los modelos de programación basados en cliente y en servidor AJAX incluyen:

Controles de servidor compatibles con desarrollo de AJAX basado en servidor. Esto incluye los controles ScriptManager, UpdatePanel, UpdateProgress y Timer. Con estos controles se puede crear un comportamiento de cliente enriquecido con un breve script de cliente o sin script, como la representación parcial de páginas y la presentación del progreso de actualización durante las devoluciones de datos asincrónicas.

Microsoft AJAX Library, que es compatible con el desarrollo basado en cliente y orientado a objetos que es independiente del explorador. Además de ser compatible con los controles de servidor habilitados para AJAX, con la biblioteca de clientes podrá desarrollar componentes de cliente personalizados que amplían los elementos DOM o que representan un elemento DOM.

Clases de servidor que le permiten desarrollar controles de servidor que se asignan a los componentes de cliente personalizados cuyos eventos y propiedades se establecen mediante declaración. Los tipos de servidor que son compatibles con esta funcionalidad incluyen los controles que se derivan de las clases base ExtenderControl o ScriptControl, o bien que implementan las interfaces IExtenderControl o IScriptControl.

Compatibilidad para la globalización y localización del script con script de cliente. Con la globalización es posible mostrar fechas y números basados en un valor de referencia cultural (configuración regional). Con la localización puede especificar el contenido localizado (texto, imágenes, etc.) en los componentes de cliente de los elementos de la interfaz de usuario o de los mensajes de excepción.

Acceso a los servicios web y a los servicios de autenticación de ASP.NET, de administración de funciones y de aplicación de perfiles.

.NET Framework versión 3.5 permite habilitar en una página, de forma sencilla, las actualizaciones parciales asincrónicas de la misma, lo que evita la sobrecarga de las devoluciones de datos de página completa. Sólo tiene que colocar el marcado y los controles existentes dentro de los controles UpdatePanel. Las devoluciones de datos dentro de un control UpdatePanel se convierten en devoluciones de datos asincrónicas y actualizan sólo la parte de la página incluida dentro del

panel, lo cual hace que la utilización por parte del usuario sea más fluida. Puede mostrar el progreso de la actualización parcial de la página mediante la utilización de controles UpdateProgress.

Obtener información sobre el desarrollo de AJAX en ASP.NET

La documentación proporciona abundante información para ayudarle a obtener información sobre cómo desarrollar aplicaciones web con estilo AJAX en ASP.NET. Para comenzar, siga la secuencia de temas descritos en Agregar funcionalidad AJAX y de cliente.

Servicios web y servicios de aplicación

.NET Framework versión 3.5 permite crear servicios web basados en ASP.NET (.aspx) y WCF a los que puede llamar desde las páginas web en script de cliente con Microsoft AJAX Library. También puede llamar a los servicios de aplicación basados en servidor que se exponen como servicios web, lo que incluye la autenticación de formularios, la administración de funciones y los perfiles. Estos servicios de aplicación se pueden usar en aplicaciones compatibles con WCF, lo que incluye páginas web habilitadas para AJAX y clientes de formularios Windows Forms. Por consiguiente, las aplicaciones que se generan con estas tecnologías ASP.NET o WCF pueden compartir información que facilitan los servicios de aplicación. Para obtener más información, vea Servicios web de AJAX en ASP.NET y Usar servicios web ASP.NET.

Control de datos ListView

El control ListView combina diferentes aspectos de controles de datos existentes. El control ListView resulta útil para mostrar datos de cualquier estructura de repetición, de forma similar a los controles DataList y Repeater. Sin embargo, a diferencia de estos controles, el control ListView admite las operaciones de edición, inserción y eliminación, así como la ordenación y la paginación. El nuevo control DataPager proporciona la funcionalidad de paginación para ListView.

El control ListView es un control con alto grado de personalización que permite usar plantillas y estilos para definir la interfaz de usuario del control. Al igual que en los controles Repeater, DataListy FormView, las plantillas del control ListView no se predefinen para representar una interfaz de usuario concreta en el explorador. Para obtener más información, vea Información general sobre el control de servidor web ListView.

Control DataPager

El control DataPager se usa para recorrer página a página los datos mostrados por un control que implementa la interfaz IPageableItemContainer, como el control ListView. El control DataPager admite la interfaz de usuario de paginación integrada. Puede especificara interfaz de usuario de paginación con el objeto NumericPagerField, que permite a los usuarios seleccionar una página por número de página. También puede usar el objeto NextPreviousPagerField, que permite a los usuarios navegar por las páginas una a una, o saltar a la primera o la última página. También puede crear una interfaz de usuario de paginación personalizada con el objeto TemplatePagerField.

Control LinqDataSource

El control LinqDataSource expone Language-Integrated Query (LINQ) a través de la arquitectura de controles de origen de datos ASP.NET. El control LinqDataSource se usa cuando se crea una página web que recupera o modifica datos y se desea usar el modelo de programación que proporciona LINQ. Puede simplificar el código de una página web permitiendo que el control LinqDataSource cree automáticamente los comandos para interactuar con los datos. Si usa el control LinqDataSource, puede reducir la cantidad de código que debe escribir para realizar operaciones de datos en comparación con las mismas operaciones en el control SqlDataSource o el control ObjectDataSource. Asimismo, cuando se utiliza el control LinqDataSource, sólo es necesario conocer un modelo de programación para interactuar con tipos diferentes de orígenes de datos.

Puede usar el marcado declarativo para crear un control LinqDataSource que conecte con los datos de una base de datos o de una recolección de datos como una colección. En el marcado, puede especificar los criterios para mostrar, filtrar, ordenar y agrupar los datos. Cuando el origen de datos es una tabla de base de datos SQL, también puede configurar un control LinqDataSource para actualizar, insertar y eliminar datos. Para realizar estas tareas, no necesita escribir los comandos SQL. La clase LinqDataSource proporciona un modelo de eventos que permite personalizar el comportamiento de visualización y actualización. Para obtener más información, consulte Información general sobre el control de servidor web LinqDataSource.

Herramienta de combinación de ASP.NET

La herramienta de combinación de ASP.NET (Aspnet_merge.exe) permite combinar y administrar ensamblados creados por la herramienta de pre compilación de ASP.NET (Aspnet_compiler.exe). (La herramienta de combinación se lanzó anteriormente como un complemento para Visual Studio 2005.) La herramienta de combinación crea ensamblados únicos para el sitio. Puede crear un ensamblado para el sitio web entero, para cada carpeta del sitio web o sólo para los archivos que constituyen la interfaz de usuario del sitio web (páginas y controles).

Microsoft Visual Studio 2008 

Microsoft® Visual Studio® 2008 cumple con la visión de Microsoft sobre aplicaciones inteligentes, al permitir que los desarrolladores creen rápidamente aplicaciones conectadas con la más alta calidad y con atractivas experiencias de usuario.

Con Visual Studio 2008, las organizaciones encontrarán que ahora es más fácil capturar y analizar información, y por lo tanto tomar decisiones de negocio más efectivas. Gracias a Visual Studio 2008, las organizaciones de todo tamaño podrán crear rápidamente aplicaciones más seguras, confiables y administrables, capaces de aprovechar mejor las características de Windows Vista™ y de Office 2007.

Visual Studio 2008 ofrece herramientas de desarrollo avanzadas, funciones de debugging, funciones para bases de datos y funciones innovadoras que permiten crear rápidamente aplicaciones futuras para distintas plataformas. Visual Studio 2008 incluye mejoras como diseñadores visuales para un desarrollo más rápido con .NET Framework 3.5, mejoras sustanciales en las herramientas de desarrollo Web y mejoras de programación que aceleran el desarrollo a partir de todo tipo de datos.

Visual Studio 2008 ofrece todo el soporte requerido para marcos y herramientas, y necesario para crear aplicaciones AJAX para Web, completas y expresivas. Los desarrolladores podrán aprovechar estos marcos con una faceta para clientes y otra faceta para servidores, marcos que permiten construir fácilmente aplicaciones Web concentradas en los clientes, que se integran con cualquier proveedor de datos de back-end, que se ejecutan dentro de cualquier explorador moderno, y que tienen acceso total a los servicios de aplicaciones ASP.NET y de la plataforma Microsoft.

Visual Studio 2008 ofrece avances clave para desarrolladores en función de los siguientes tres pilares:

- Desarrollo rápido de aplicaciones
- Colaboración eficiente entre equipos
- Innovación en experiencias de usuario

Desarrollo rápido de aplicaciones

Para que los desarrolladores puedan crear rápidamente software moderno, **Visual Studio 2008** ofrece funciones de programación y de datos mejoradas, como **LINQ** *(Language Integrated Query)*, que facilita el armado de soluciones capaces de analizar información y de actuar en consecuencia. **Visual Studio 2008** también brinda la posibilidad de apuntar a distintas versiones de **.NET Framework** desde el mismo entorno de desarrollo. Por lo tanto, los desarrolladores podrán construir aplicaciones que apunten a **.NET Framework 2.0**, **3.0** o **3.5**, y así podrán admitir una amplia variedad de proyectos en un mismo entorno.

Colaboración eficiente entre equipos

Visual Studio 2008 propone ofertas expandidas y mejoradas que ayudan a mejorar la colaboración entre equipos de desarrollo, incluidas herramientas que colaboran con la integración entre profesionales especializados en bases de datos y diseñadores gráficos.

Innovación en experiencias de usuario

Visual Studio 2008 ofrece nuevas herramientas que aceleran la creación de aplicaciones conectadas con las últimas plataformas, incluidas la Web, Windows Vista, Office 2007, SQL Server 2008 y Windows Server 2008. Para la Web, ASP.NET AJAX y otras nuevas tecnologías permitirán que los desarrolladores creen rápidamente una nueva generación de experiencias más eficientes, interactivas y personalizadas.

Uso de Microsoft .NET Framework 3.5

.NET Framework permite la construcción rápida de aplicaciones conectadas que ofrecen experiencias de usuario increíbles, ya que ofrecen bloques de construcción (software pre-fabricado) que resuelven las tareas de programación más frecuentes. Las aplicaciones conectadas construidas sobre los modelos de negocio de **.**NET Framework procesan de manera efectiva y facilitan la integración de sistemas en entornos heterogéneos.

Juntos, Visual Studio y **.**NET Framework reducen la necesidad de código en común, disminuyen los tiempos de desarrollo, y permiten que los desarrolladores se concentren en resolver problemas comerciales.

.NET Framework 3.5 construye más que .NET Framework 3.0. Las mejoras fueron aplicadas a áreas fundamentales como la biblioteca básica de clases, Windows Workflow Foundation, Windows Communication Foundation, Windows Presentation Foundation **y** Windows CardSpace**.**

Herramientas RAD

La primera metodología para la programación de aplicaciones fue la utilización directa de llamadas a las funciones del API (Application Programming Interface) de Windows. Este mecanismo suponía una pesada tarea de codificación para crear aplicaciones con simples elementos tales como menús, ventanas, cuadros de diálogo, etc

A medida que el entorno Windows fue extendiéndose y ofrecía nuevas posibilidades tales como DDE y OLE, era necesario un conjunto de herramientas que permitiese un desarrollo de aplicaciones más productivo y que facilitase la creación de aplicaciones más complejas. Es cuando aparecen los denominados marcos de aplicaciones (application frameworks) tales como MFC (Microsoft Fundation Classes). Con ellas, surge el término de encapsulación de las funciones del API de Windows. Estas librerías permitían un rápido diseño de, por ejemplo, un cuadro de diálogo modal con estilo en tres dimensiones y dos botones, que podía ser llamado con una única sentencia.

Pero, además, se incorporaba también la potencia de la programación orientada a objetos y todo lo que ello suponía. Los proyectos podían ser divididos en mini proyectos de diseño de ciertos componentes visuales y algorítmicos que luego podían ser fácilmente enlazados en la aplicación final. Además se permitía el prototipado creando simples elementos de presentación que serían unidos al código diseñado con posterioridad. Y una importante característica de la programación orientada a objetos como es la herencia, permitía la creación de nuevos componentes dentro de la jerarquía de clases prediseñada.

Empiezan a aparecer por entonces nuevos términos como el de RAD (Rapid Application Development), pero sin duda el más conocido es el de "Programación Visual". Este fue acuñado por Microsoft Corporation(r) en 1991 con la aparición de la primera herramienta de este tipo, Visual BasicTM.

La programación visual orientada a objetos es ya una realidad. Las posibilidades que se ofrecen son insospechadas. Crear un objeto o instancia de una clase (control) es tan fácil como arrastrar y soltar un control en el formulario. La particularización de un determinado objeto se consigue asignando valores a sus propiedades. El código asociado a un determinado objeto es automáticamente encapsulado a él.

Los actuales entornos RAD permiten no solo el desarrollo de aplicaciones finales sino también el diseño de prototipos eficientes, y ofrecen:

- Entorno de desarrollo visual, que incluye, entre otros, el menú propio del entorno, la paleta de componentes, la ventana de propiedades (que refleja las propiedades asociadas al control activo) y el editor de código.

- Controles de alto nivel, que suelen ser de dos tipos: los incorporados en la propia herramienta y aquellos externos (VBX y OCX principalmente) que pueden ser añadidos. Estos controles se disponen en forma de iconos en una paleta de componentes.

- Acceso directo a las distintas partes del código asociado a cada uno de los objetos, y que se encuentra en la ventana del editor de código.

Las aplicaciones diseñadas se basan en entornos gráficos de usuario (GUI) fáciles de manejar y con el siguiente proceso de desarrollo:

- Creación de un formulario vacío que contendrá los componentes del interfaz de la aplicación (controles).

- Selección de los componentes apropiados del conjunto de los disponibles en la paleta de componentes en forma de iconos. Estos pueden luego ser redimensionados una vez colocados en el formulario.

- Particularización de las propiedades de los controles en base a los requisitos de la aplicación.

- Escritura de código para los distintos eventos significativos asociados al control. Este código se estructura dentro de cada procedimiento del control según el lenguaje de programación utilizado, y se pueden incluir sentencias propias del lenguaje y referencias a métodos y procedimientos del propio control o de otros controles, estén o no en el mismo formulario.

- Ejecuciones de prueba dentro del entorno de desarrollo.

- Modificación de la aplicación durante el proceso de desarrollo hasta la generación de un fichero ejecutable (.EXE) definitivo.

- Utilización de herramientas de apoyo para depurar y refinar el código.

La reutilización de componentes es uno de los avances más significativos dentro de esta tecnología. Es posible realizar aplicaciones muy potentes enlazando controles de alto nivel ya diseñados y en donde el desarrollador sólo debe preocuparse de organizar el enlace entre ellos dentro de su aplicación. Estos componentes pueden generarse a partir de una DLL creada con un compilador de C/C++, como es el caso de los VBX, o pueden generarse dentro de la propia herramienta, como ocurre con los servidores OLE de Visual Basic. Estos componentes suelen diseñarse utilizando características de programación orientada a objetos, pues ofrece ventajas tales como: ciclos de desarrollo más cortos, facilidad en el mantenimiento y en el control de versiones, facilidad en la compartición de código con otros componentes, etc..

Existe la posibilidad de que varios eventos de un mismo control o de diferentes controles compartan un mismo procedimiento o función codificada en la aplicación. Además es posible utilizar funciones externas incluidas en una DLL, para ello basta con definir en el código de la aplicación el prototipo de dicha función. Estas DLL pueden haber sido diseñadas con cualquier compilador, y en ellas se incluyen las propias de Windows (GDI, KERNEL y USER) que permiten hacer uso de las funciones API del entorno Windows.

Las herramientas RAD permiten crear aplicaciones con bases de datos. La conexión con una base de datos se puede efectuar de dos formas:

- Directamente, utilizando el motor de dicha base de datos.

- A través de enlaces remotos (vía ODBC).

En la aplicación se pueden utilizar dos alternativas:

- Controles específicos en el propio formulario.

- Objetos en el código.

Los controles de bases de datos incluidos en un formulario disponen de propiedades que permiten especificar la tabla o la base de datos con la que se enlazará. Los objetos incluidos en el código disponen de métodos que enlazan de igual forma con bases de datos. En ambos casos es posible indicar si dicho enlace es directo (a través del motor de la base de datos) o es remoto. Ambas

alternativas también contemplan el acceso a los datos a través de los métodos que implementan o utilizando búsquedas con sentencias SQL.

Además de los controles propios de bases de datos, adicionalmente se utilizan otros como cuadros de texto, etiquetas, listas desplegables, controles de tipo tabla (grids), etc. para la visualización de los resultados de lectura o para permitir la introducción de los datos a escribir en la base de datos.

AJAX

AJAX son las siglas de **A**synchronous **J**avaScript **A**nd **X**ML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que nos permiten hacer páginas de internet más interactivas.

La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor.

La complejidad se encuentra en que debemos dominar varias tecnologías:

- HTML o XHTML
- CSS
- JavaScript
- DHTML Básicamente debemos dominar todos los objetos que proporciona el DOM.
- XML Para el envío y recepción de los datos entre el cliente y el servidor.
- PHP o algún otro lenguaje que se ejecute en el servidor (ASP.Net/JSP)

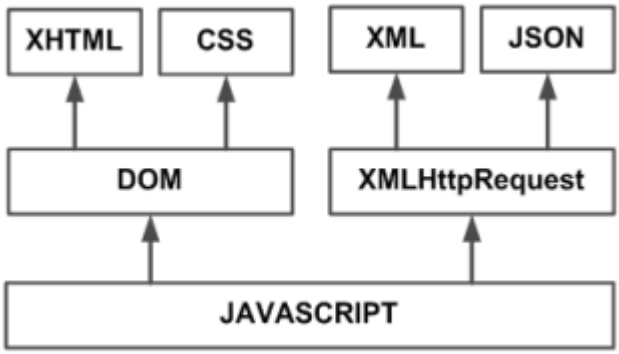

Figura 18.- Ejemplo de las tecnologías web.

En las aplicaciones web tradicionales, las acciones del usuario desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:

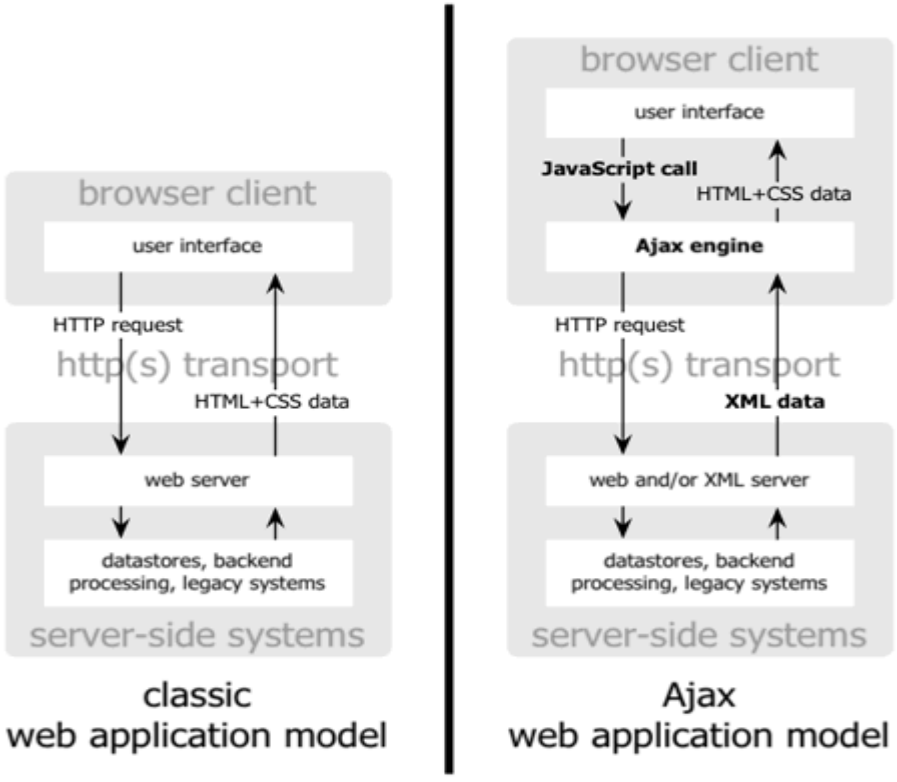

Figura 19.- Modelo clásico de aplicaciones web contra el modelo de Ajax.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interacción síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX.

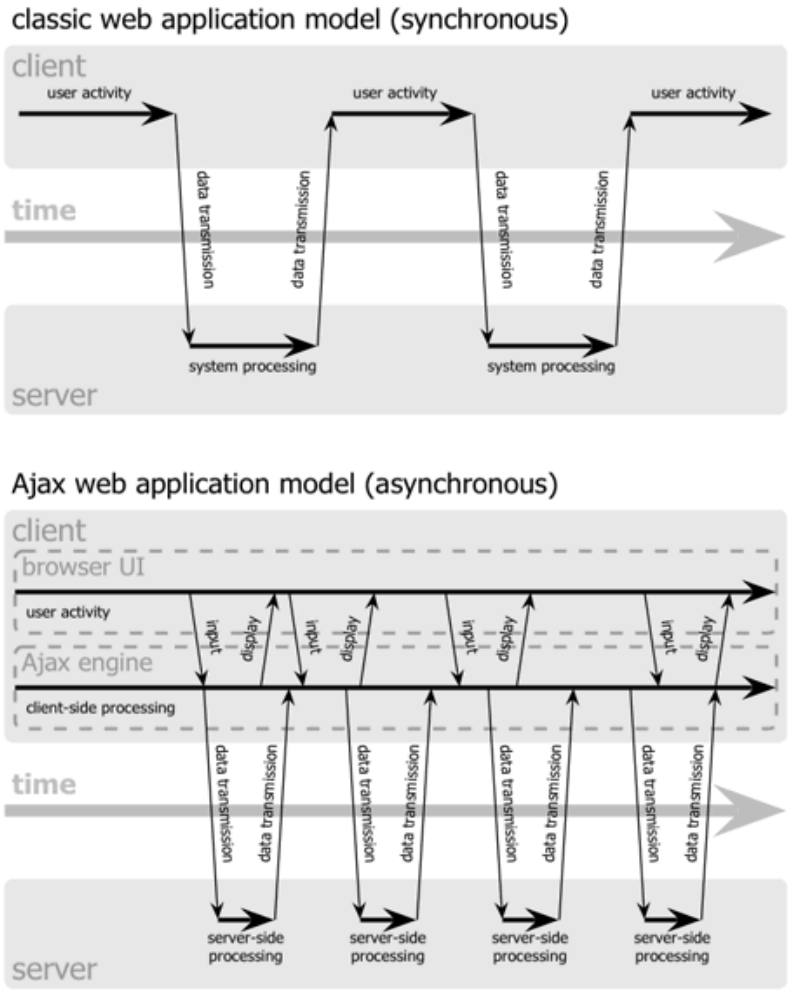

Figura 20.- Representación de la comunicación del modelo clásico contra el modelo de AJAX.

Comparación entre las comunicaciones síncronas de las aplicaciones web tradicionales y las comunicaciones asíncronas de las aplicaciones AJAX (Imagen original creada por Adaptive Path y utilizada con su permiso)

Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio.

Ventajas y desventajas de AJAX

Ventajas

1. Utiliza tecnologías ya existentes.
2. Soportada por la mayoría de los navegadores modernos.
3. Interactividad. El usuario no tiene que esperar hasta que lleguen los datos del servidor.
4. Portabilidad (no requiere plug-in como Flash y Applet de Java)
5. Mayor velocidad, esto debido que no hay que retornar toda la página nuevamente.
6. La página se asemeja a una aplicación de escritorio.

Desventajas

1. Se pierde el concepto de volver a la página anterior.
2. Si se guarda en favoritos no necesariamente al visitar nuevamente el sitio se ubique donde nos encontrábamos al grabarla.
3. La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.
4. Problemas con navegadores antiguos que no implementan esta tecnología.

5. No funciona si el usuario tiene desactivado el JavaScript en su navegador.

6. Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.

7. Dependiendo de la carga del servidor podemos experimentar tiempos tardíos de respuesta que desconciertan al visitante.

Framework de JavaScript

Hemos pasado de páginas estáticas basadas en solamente HTML y CSS a páginas dinámicas que utilizan motores externos como PHP, ASP y Coldfusion para dar este dinamismo. Aunque Javascript no es una tecnologí-a reciente ha tomado mayor popularidad porque enriquece la experiencia de los usuarios de los sitios web.

La evolución nos ha traído el Cloud Computing o “Computación en Nube”, los Sistemas Operativos Web y las Aplicaciones basadas en Web. Todo esto gracias, en parte, a la implementación de JavaScript y los frameworks. Entre dichos frameworks, se encuentran jQuery, MooTools, ExtJs, Prototype, Scriptaculous, Dojo Toolkit, YUI, y más.

Características de los Frameworks

Es necesario entender que un framework es una abstracción de código común que provee funcionalidades genéricas que pueden ser utilizadas para desarrollar aplicaciones de manera rápida, fácil, modular y sencilla, ahorrando tiempo y esfuerzo. Entonces, un framework es concreto y también “incompleto”. Concreto porque es, desde un punto de vista simple, un conjunto de componentes; incompleto, porque por sí mismos no pueden ser utilizados, ya que guían a la solución de problemas de programación recurrentes, empero, por lo general, no son la solución específica completa.

En su mayoría, los frameworks JavaScript proveen componentes para:

- Compatibilidad. Agregan la posibilidad de escribir código JavaScript totalmente compatible con todos los navegadores y motores Javascript más utilizados. Esto aumenta la portabilidad y eliminan el “gran dolor de cabeza” de incompatibilidad entre navegadores y sus motores intérpretes JavaScript.

- Comunicación asíncrona (Ajax). Usando este acercamiento, es fácil utilizar XMLHttpRequest para manejar y manipular los datos en los elementos de un sitio bien, aumentando la interactividad y experiencia del usuario.
- DOM. Maximizan la capacidad de agregar, editar, cambiar, eliminar elementos de manera dinámica agregando librerías que facilitan usar DOM.
- Validación de Formularios. Permiten de una manera relativamente fácil validar campos dentro de uno o varios formularios. Esto, desde el punto de vista del desarrollador, simplifica y reduce el código para procesar dichos formularios, ya que los datos llegan previamente validados, reduciendo los errores de tipos de datos.
- Efectos visuales. Utilizando la manipulación de los elementos, se pueden crear efectos visuales y animaciones. Entre los efectos se encuentran: aparecer y desaparecer, redimensionamiento, mover, aparecer y más.
- Almacenamiento Client-side. En adición provee funciones para leer y escribir cookies. También proveen una abstracción de almacenamiento que permite a las aplicaciones web guardar datos del lado del cliente, persistente y de manera segura.
- Manejo JSON. Incrementa al máximo el manejo de datos, que pueden ser utilizados para presentar informaciones de manera dinámica y en tiempo de ejecución.
- Manejo de Eventos. Esta característica agregada, permite reaccionar de una manera u otra dependiendo de las acciones del usuario.
- Recibidores de Datos. Permiten utilizar diferentes formatos de datos como XML, HTML, Texto, JSON, ATOM, entre otros.
- "Arrastra y Suelta". Mejor conocido como Drag and Drop. Es una funcionalidad que brinda la posibilidad de arrastrar elementos dentro de una misma página que interactúe con el resto de los elementos.

Aplicaciones Web vs. Sitios Web

Suele confundirse los conceptos de Aplicación Web con Sitio Web, lo cual lleva a la eterna discusión sobre cuál es el mejor framework, independientemente de la eficiencia de los mismos. Una aplicación Web intenta portar las clásicas aplicaciones de Escritorio hacia entornos Web, que son utilizadas a través de los distintos navegadores, agregando portabilidad y capacidad de acceder desde diferentes dispositivos.

Un sitio web es simplemente eso, un portal, una página, un grupo de páginas, una serie de documentos dinámicos o no. En ambos casos se puede mejorar la experiencia del usuario utilizando JavaScript y en efecto, frameworks. Aparte de la eficiencia, simplicidad, rendimiento, documentación, forma de uso, entre otros factores más, la finalidad de lo que se quiere crear con estos frameworks es uno de los más importantes a la hora de elegir entre uno u otro.

Otro factor importante que se debe considerar es la curva de aprendizaje y la facilidad de uso cuando se crean Proyectos donde intervienen más de una persona. Una variable a considerar es la buena práctica y los patrones de programación aplicados.

Datos importantes y ejemplos de frameworks

Cada framework tiene sus características que los diferencian de los demás. Podemos destacar lo siguiente:

Dojo Toolkit

Está compuesto por Widgets que son componentes de código en Javascript pre-empaquetados que puede ser utilizados para enriquecer sitios web con varias caracterí-sticas que trabajan a través de la mayoría de los navegadores, tales como: menús, tabs, tooltips y tablas ordenables.

Aunque la funcionalidad tal vez única de Dojo es la capacidad para el Dibujo de Vectores en 3-D. También es posible utilizar temas para mejorar el look and feel de la aplicación. En 2007 Dojo incluyó ejemplos de almacenamiento del lado del servidor que son implementaciones del namespace dojo.data.

IBM y Sun Microsystems han anunciado el soporte oficial a Dojo, incluyendo contribuciones de código. De la misma forma Zend Technologies, la compañía detrás del motor de PHP, lo ha incorporado en Zend Framework. Dojo es utilizado por Lucid Desktop, un WebOs open source. También puede ser integrado en aplicaciones creadas con AdobeAIR.

Una galería de Imágenes a estilo del iPod Touch, un Juego de Matemáticas, un Reproductor de Videos y una Lista con efecto FishEye son unos de los ejemplos que podemos encontrar en la web de Dojo Toolkit.

ExtJS

Originalmente fue construido como una extensión de YUI. Incluye interporalidad con JQuery y Prototype. Posee controles para Campos de Textos, incluyendo Áreas de Texto. Controladores selectores de fecha, Campos Numéricos, para Radio box y Checkt box.

También componentes para crear y manipular DataGrids donde goza de cierta ventaja sobre otros frameworks. Es posible crear "ventanas" con Barras de Herramientas y Menús con estilo de aplicaciones de Escritorio, Diálogos modales y eventos.

ExtJS puede ser adquirido bajo licencias Libres y Comerciales. La compañía detrás de este framework ofrece cursos de capacitación y un extenso soporte. Entre los ejemplos ofrecidos en su sitio web encontramos ejemplos de Tabs, Ventanas, Árboles, Menús y más.

jQuery

Es una librería liviana que enfatiza la interacción entre Javascript y HTML. Microsoft integra jQuery Visual Studio para el uso en aplicaciones desarrolladas en ASP.NET. Nokia usa jQuery en sus aplicaciones web. Contiene selectores de elementos DOM usando el motor Sizzle y permite la modificación de DOM (incluyendo soporte para CSS 3 y Xpath).

Actualmente su última versión estable, la 1.3.2, viene un solo archivo de 19KB, y su funcionalidad puede ser extendida utilizando plugins. Algunos plugins útiles son: jQueyUI, un set de plugins, componentes para interfaces de usuario y efectos visuales y jExpand, un ultra-liviano plugin que permite crear tablas expandibles que ayuda a organizar mejor los datos. Entre otros más.

Mootools

Liviano, modular y orientado a objetos, la meta es ser un intermediador para los desarrolladores ayudándolos a crear código JavaScript en una manera elegante, flexible y eficiente. Contiene un gran número de componentes, pero no todos necesitan ser cargados en cada aplicación. Consta de un Core, que es una colección de librerías que el resto de sus componentes necesitan, Class, que es la librería básica.

También provee unas componentes que enriquecen a los objetos nativos de JavaScript, para agregar compatibilidad y simplificación de código. Fx es su API avanzado de efectos para animar Elementos. Algunas aplicaciones utilizando MooTools son: ape, bing, joomla, vimeo, palm, Nintendo, phpMyAdmin y netvibes.

Prototype y Scriptaculous 

Prototype es una simple implementación de un solo archivo de código en Javascript que provee un framework para Ajax y otras herramientas. Contiene varias funciones para programar en Javascript que van desde accesos directos a funciones, elementos y objetos Javascript, hasta funciones para lidiar con XMLHttpRequest.

Scriptaculous es una librería JavaScript basada en Prototype que agrega efectos visuales dinámicos y una interface para elementos a través de DOM. Viene incluido en Seaside y Ruby on Rails.

¿Cuál es el mejor framework?

Determinar cuál framework es mejor es prácticamente imposible. Es importante recordar que estos frameworks son simples herramientas que nos ayudan a realizar diferentes tareas de diferentes tipos, todos basados en el mismo lenguaje, JavaScript. Lo correcto es seleccionar uno u otro dependiendo la utilidad y la capacidad de saber cómo utilizarlo.

jQuery ha tenido gran éxito, ya que es fácil de usar y aprender, además de que existe en internet una gran cantidad de documentación y plugins que extienden su funcionalidad. Ha sido muy utilizado por Diseñadores con pocos conocimientos en programación.

Scriptaculous, basado en Prototype, se integra bastante bien con Ruby On Rails, lo cual ha beneficiado a desarrolladores independientes y empresas al momento de crear aplicaciones de vanguardia.

Dojo Toolkit, en su incorporación con Zend Framework promete mucho, pero no goza de la popularidad de MooTools, que aunque es liviano y orientado a objetos, en algunas ocasiones

resulta complicado realizar simples acciones, que utilizando Prototype suelen ser bastantes sencillas, como Peticiones Periódicas y acceso a diferentes elementos.

No cabe duda de que si lo que se quiere crear es una aplicación web con estilo de aplicación de Escritorio, ExtJs es la mejor opción, inclusive se podría crear un Sistema Operativo Web utilizando este framework de forma relativamente fácil. Lo importante es saber utilizar las herramientas con las que se cuentan y sacarle el máximo provecho

Según nuestras necesidades y gustos hemos decidido utilizar ExtJS

Crisis del software Punto que deseamos erradicar

La crisis del software es el hecho de que el software que se construye no solamente no satisface los requerimientos ni las necesidades pedidos por el cliente, sino que además excede los presupuestos y los horarios de tiempos. La industria del software no ha podido satisfacer la demanda, la complejidad del software producido y demandado se incrementa constantemente, el software es solicitado para ejecutar las tareas demandantes de hoy y está presente en todos los sistemas que van desde los más sencillos hasta los de misión crítica. Las aplicaciones de software son complejas porque modelan la complejidad del mundo real. En estos días, las aplicaciones típicas son muy grandes y complejas para que un individuo las entienda y, por ello, lleva gran tiempo implementar software.

Síntomas

Uno de los principales problemas en el desarrollo de software de hoy en día es que muchos proyectos empiezan la programación tan pronto se definen y concentran mucho de su esfuerzo en la escritura de código. Últimamente el desarrollo de software se ralentizado. El estudio de este fenómeno es importante porque la existencia de software científico libre facilita que cualquier laboratorio del mundo pueda desarrollar ciencia libre usando este software como herramienta de trabajo.

Algunos "síntomas" que indican que el software se encuentra en un periodo de crisis son:

- Baja Calidad del Software.
- Tiempo y Presupuesto Excedido.
- Confiabilidad Cuestionable.
- Altos Requerimientos de Personal para desarrollo y mantenimiento.

Factores de Influencia

Para poder llevar el estado del proceso de software como un estado de crisis, los críticos han destacado ciertas características que han permitido esta postura del software respecto a otras etapas de su corta historia. Algunos de esos factores son:

- Aumento del poder computacional.
- Reducción del costo del hardware.
- Rápida obsolescencia de hardware y software.
- Aceptación de la computarización en las empresas.
- Incremento en el número de usuarios de los sistemas de software.
- Tipo de usuario no homogéneo aun en sistemas hechos a la medida.
- Personal de desarrollado y mantenimiento diferente.
- La magnitud del proyecto impacta en:
    - Tiempo costo y número de desarrolladores,
    - Control administrativo y detalles técnicos
    - Aumento en el conocimiento del problema.

Cambios en el entorno

- Tecnológicos (Internet, redes, ERP, CRM, SCM).
- Económicos (crisis económicas, globalización, etcétera).
- Sociales (nuevas necesidades, costumbres nuevas, etcétera).

Posibles causas de la crisis del software

Hay varias razones que pueden ser propuestas como causa de la crisis. No son mutuamente excluyentes; de hecho, es posible que la verdadera causa sea una mezcla de todas ellas. Sin embargo, todas tienen en común que son causadas por el método de valorar los avances científicos y el mecanismo actual de financiación de la actividad científica. Las causas de la crisis del software fueron vinculadas a la complejidad en general del proceso de software y a la relativa inmadurez de la ingeniería de software como una profesión. La crisis se manifestó a sí misma en varias maneras:

- Proyectos gestionados con un sobre-presupuesto.
- Proyectos gestionados con sobre tiempo.
- Software de baja calidad.
- El software a menudo no satisfacía los requerimientos deseados.
- Los proyectos fueron inmanejables, con un código difícil de mantener.

Puntos que buscamos en el desarrollo de este proyecto

BAX Framework ASP.NET 3.5.

BAX Framework se enfoca en resolver los problemas más comunes que se presentan en el desarrollo y es la base de las aplicaciones web que la empresa desarrolla. Como base de dichas aplicaciones debe proveer características:

- Rápidas.
- Confiables.
- Seguras.
- Fáciles.

Para que los analistas, que no son expertos programadores, puedan construir fácilmente aplicaciones de alta calidad con poco conocimiento técnico y se enfoquen en el negocio y en la interacción con el usuario.

Adicionalmente podemos hacerlo de tal forma que la tecnología del lado del servidor sea intercambiable a fin de atender las 2 plataformas de desarrollo estándar de facto actuales: .NET y Java. Llamamos intercambiable al hecho de poder cambiar todo el lado del servidor a otra tecnología, haciendo pequeños cambios o ninguno al lado del cliente y la aplicación tendrá que funcionar perfectamente en cualquiera de los casos.

Para ello buscaremos 3 características importantes:

Alto performance para el usuario final.

Alta mantenibilidad.

Server Side Transparente.

Alto performance

Dejar de usar las facilidades que proporciona el framework ASP.NET para la creación de aplicaciones web.

¿Por qué si ASP.NET 3.5 tiene tantas facilidades dejar de usarlas?

- No es eficiente. No cumple con las expectativas del cliente o termina siendo más complicado manejarlo de esa manera.
- Depende de cookies ó de información que puede verse a simple vista.

- El overhead para incrementar la productividad del programador genera problemas potenciales de performance, por ejemplo, el código del CS generado por los componentes visuales es excesivo lo que ocasiona más tráfico en la red lo mismo el mantenimiento del estado de cada componente.
- Hay operaciones que necesitan más codificación que hacerse a mano, por ejemplo los verbos AJAX y las operaciones sobre bases de datos.
- Las herramientas visuales necesitan muchos recursos y en una máquina estándar es poco práctico usarlos.

Algunas prácticas que apoyan lo anterior

Mucho tiempo que el usuario final espera para la carga de una aplicación web, se invierte en esperar que los componentes como imágenes, hojas de estilo, scripts, flash, etc. se terminen de descargar para que pueda ser utilizada la aplicación. Por lo que existen prácticas que ayudan a reducir peticiones HTTP, mientras el diseño sigue enriqueciendo a las páginas web.

- Archivos combinados
- Combinación de todas las hojas de estilo.
- Cargar una sola imagen y mediante CSS recortar esta imagen en otras según su utilidad.
- Utilizar una colección de servidores distribuidos en múltiples localizaciones.
- Agregar expiraciones o cabeceras para el control de la cache. Es decir para los componentes estáticos aplicar una política de "never-expire" y para los componentes dinámicos utilizar "Cache-Control" para apoyar al navegador con peticiones condicionales.
- Poner las hojas de estilo en la parte superior da la página. (Investigaciones que realizo Yahoo dieron como resultado que si se ponen de eta manera las hojas de estilo la pagina carga más rápido).
- Poner los Scripts al final de la página (La especificación HTTP/1.1 así lo sugiere)
- Evitar las expresiones CSS dentro del código HTML.
- Crear el archivos JavaScript y CSS externos para que el navegador guarde en cache estos archivos.
- Reducir la búsqueda de DNS
- Minimizar practicas innecesarias de JavaScript y CSS, es decir no abusar de las hojas de estilo ni de de js que no son de utilidad.
- Evitar las redirecciones. Esto sucede cuando uno olvida colocar la barra diagonal (/) al final de una url por ejemplo: http://astrology.yahoo.com/astrology en respuesta es una redirección a http://astrology.yahoo.com/astrology/.
- Eliminar Scripts duplicados.
- Optimizar la respuesta del AJAX.

- Utilizar el método GET para las respuestas de AJAX.
- Post-Carga de componentes.
- Pre-Carga de componentes.
- Reduce el número de elementos DOM (Document Object Model [Modelo de Objetos del Documento]).
- Minimizar el uso de iframes.
- Es importante tener un tamaño pequeño de las cookies para minimizar el impacto en el tiempo de respuesta a los usuarios.
- Optimización de CSS mediante Sprites (de una imagen solo mostrar la parte que nos interesa).
- No escalar imágenes en HTML
- Guardar componentes debajo de los 25K
- Etc.

En cuanto al código del lado del servidor

No utilice excepciones en el código Las excepciones pueden afectar al rendimiento de manera significativa, por lo que debería evitarlas para controlar el flujo normal del programa. Si es posible detectar en el código una condición que podría provocar una excepción, hágalo en lugar de almacenar la excepción en la caché y controlar la condición. Algunos escenarios comunes que se pueden detectar en el código son comprobar si un valor es **null**, asignar un valor a String que se va a analizar como valor numérico o comprobar valores específicos antes de aplicar operaciones matemáticas. En el ejemplo siguiente se incluye código que causaría una excepción y código que comprueba una condición. Ambos fragmentos producen el mismo resultado.

C#

```
// This is not recommended.
try {
   result = 100 / num;
}
catch (Exception e) {
  result = 0;
}

// This is preferred.
if (num != 0)
   result = 100 / num;
else
  result = 0;
```

Si utilizamos el operador <%= para mostrar un valor, estamos llamando de una u otra forma a una función Response.Write lo que aumenta el tiempo de proceso de la página ASP así que lo mejor es siempre utilizar el Response.Write(). Ahora veamos este caso...

```
<%

Response.Write("<head>")
Response.Write("<title>")
Response.Write("Pagina                                                    ....")
Response.Write("</title>")
Response.Write("</head>")

%>
```

Este ejemplo utiliza varias llamadas al Response.Write() lo mejor que se pude hacer en este caso es juntar todas en un solo Response.Write()...

```
Response.Write("<head><title>Página ....</title></head>")
```

Utilizar comentarios

Lo comentarios tiene un mínimo impacto en la velocidad de la pagina ASP, entonces siempre que sean moderados los comentarios no se notara en el rendimiento de la pagina. Sin embargo, los comentarios que se generan de manera automática cuando agregamos un componente desde el Visual Studio suele ser redundante y eventualmente innecesario.

Alta mantenibilidad.

Seguimos la siguiente tendencia

Privilegiar a la mantenibilidad sobre el performance y el uso de memoria.

Uno de los problemas a los que nos enfrentábamos al desarrollar software es que de una manera u otra siempre nos veíamos limitados por los siguientes aspectos.

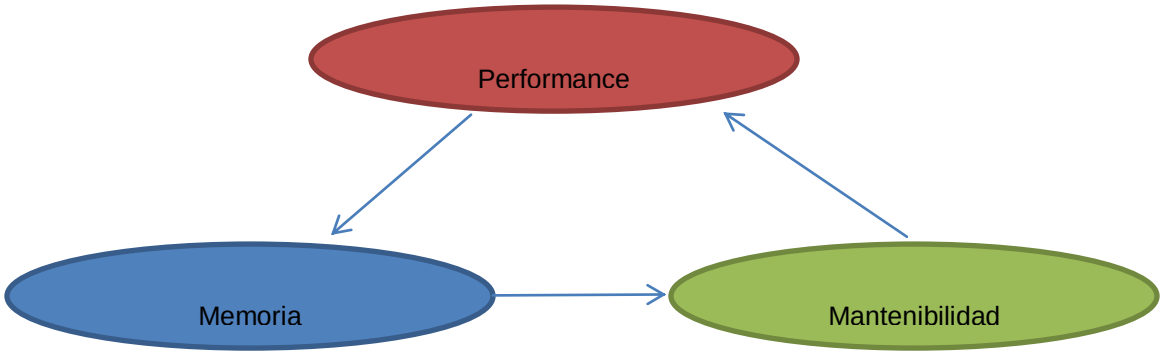

Figura 21.- Diagrama de la relación Performance - Memoria - Mantenibilidad.

Se tenía una relación estrecha entre el Performance-La Memoria-y la Mantenibilidad del código. Y muchas veces se prefería que la aplicación tuviera un buen performance y tratar de ser razonable con el uso de la memoria antes de preocuparnos por la mantenibilidad, sin embargo esto traía problemas cuando se requería hacer un cambio tanto para la misma persona que desarrollo el programa, como para otro programador que retomara, agregara o actualizara la aplicación; por lo que esto de sacrificar la mantenibilidad no era del todo una gran idea puesto que después se perdía mucho tiempo haciendo esas actividades de mantenimiento.

Sin embargo al día de hoy existen maquinas muy poderosas, y no nos tenemos que preocupar por la memoria ni por el performance ya que la evolución de estos dos términos ha sido exponencial y significativamente.

"Las aplicaciones web se prestan para ejecutarse en servidores enormes y en clientes pequeños".

Por ello especialmente para la elaboración de este framework se consideraron aspectos importantes que rigieron la programación de este framework, como:

- Nombramiento apropiado para atributos y métodos.
- Utilizar comentarios.
- Buena documentación.
- Utilizar y explotar los beneficios que da la Programación Orientada a Objetos.
- Utilizar un buen mecanismo de manejo de errores.

Al enfocarnos en estos aspectos logramos hacer más mantenible el código, se podrá expandir o integrar nuevas funciones si así lo requiere en un futuro, si alguna persona requiere modificar el código será fácil y se encontraran rápidamente errores que tenga el sistema.

Se considerará que se cumple el objetivo de ser mantenible si:

- Se pueden encontrar y corregir fácilmente los bugs (errores).
- Se pueden agregar características nuevas fácilmente.
- Nuestros ingenieros pueden entenderlo con poca documentación.

Para ello aplicaremos los siguientes principios.

Nombrar correctamente y hacer mucho refactoring (refactorización).

En ingeniería del software, el término refactorización se usa a menudo para describir la modificación del código fuente sin cambiar su comportamiento, lo que se conoce informalmente por limpiar el código. La refactorización se realiza a menudo como parte del proceso de desarrollo del software: los desarrolladores alternan la inserción de nuevas funcionalidades y casos de prueba con la refactorización del código para mejorar su consistencia interna y su claridad. Los test aseguran que la refactorización no cambia el comportamiento del código.

La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. El objetivo, por el contrario, es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto, para facilitar el mantenimiento en el futuro. Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede refactorizarlo primero para facilitar esta tarea y luego añadir el nuevo comportamiento.

La propia Orientación a Objetos tiende a ser auto documentativa. La estructura de Objeto->OrdenAlObjeto() hace que el código sea entendible por medio de la simple lectura, pero para que esto funcione es necesario nombrar correctamente los elementos de codificación: clases, instancias, propiedades, métodos y variables.

Las variables deben ser sustantivos, nombres cortos de preferencia y muy descriptivos.

Los nombres de los métodos deben ser verbos, de ser necesario con complementos que describan muy bien la acción.

Se sabe que al resolver un problema no es fácil encontrar los nombres adecuados, así que procederemos de la siguiente forma:

- Resolver los algoritmos como se pueda.
- Una vez que se tenga el algoritmo con el resultado que se espera se revisa para buscar oportunidades de mejora en su estructura. Buscamos reducir el código y nombrar correctamente los elementos de programación usados.

Para esto el refactoring, en su faceta de cambiar nombres y simplificar el código, será una práctica común durante el desarrollo.

Aplicar el principio DRY (Don't Repeat Yourself)

El principio no repitas, conocido también como **Una vez y sólo una** es una filosofía de definición de procesos que promueve la reducción de la duplicación especialmente en computación. Según este principio ninguna pieza de información debería estar duplicada nunca debido a que la duplicación incrementa la dificultad en los cambios y evolución posterior, puede perjudicar la claridad y crea un espacio para posibles inconsistencias.

Cuando el principio DRY se aplica de forma eficiente los cambios en cualquier parte del proceso requieren cambios en un único lugar. Por el contrario, si algunas partes del proceso están repetidas por varios sitios, los cambios pueden provocar fallos con mayor facilidad si todos los sitios en los que aparece no se encuentran sincronizados

Al aplicar DRY se obtiene cohesión fuerte y acoplamiento débil, principio básico y todavía válido de la programación estructurada.

Un solo lugar donde esté el código implica un solo lugar donde buscar:

- Bugs ,
- Donde corregir y

- Donde ampliar características.

DRY es la parte más importante a cuidar en la programación, pero ¿cómo aplicar DRY?

El objetivo a buscar es centralizar el código y escribir una sola versión reutilizable. Para ello empleamos las siguientes técnicas para centralizar código. Para ello usamos la Inversión de Control y la Inyección de Dependencia.

Inversión de Control y la Inyección de Dependencia

En los comienzos de la programación, los programas eran lineales y monolíticos. El flujo de ejecución era simple y predecible, ejecutándose línea tras línea.

Aparecieron dos conceptos que revolucionaron la programación: la modularidad y la reutilización de los componentes: se crean bibliotecas de componentes reutilizables. El flujo se complica, saltando de componente a componente, y aparece un nuevo problema: la dependencia (acoplamiento) entre nuestros componentes.

El problema se empieza a considerar lo suficientemente importante como para definir nuevos conceptos en el diseño:

- Inversión de Control (IoC)
- Inyección de Dependencias (Dependency Injection, DI)

La inversión de control es una aplicación práctica del principio de inversión de dependencia que se usa en aquellas situaciones donde se desea que el código genérico controle la ejecución de componentes específicos.

Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (procedure calls) o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.
A día de hoy la inversión de control a menudo se asocia a un conjunto de frameworks que ofrecen características avanzadas para inyectar código en tiempo de ejecución.

Algunos frameworks conocidos para inversión de control son los siguientes :

- Castle Windsor
- Ninject
- Spring .NET
- StructureMap

La mayoría de los frameworks de inversión de control están implementados sobre un objeto contenedor que resuelve las dependencias a través de un fichero de configuración, el procedimiento suele ser en la mayoría de los casos el mismo:

- Se instancia el contenedor
- Se pasa la interfaz como argumento
- El framework devuelve un objeto que implementa la interfaz

Inyección de Dependencias (en inglés Dependency Injection, DI) es un patrón de diseño orientado a objetos, en el que se suplen objetos a una clase en lugar de ser la propia clase quien cree el objeto. El término fue acuñado por primera vez por Martin Fowler.

Ambos patrones se encuentran muy asociados, tanto que es normal confundirlos, para hacerlo claro, inversión de control es el modo en que funcionan generalmente los frameworks, en lugar de nosotros instanciar un objeto Database en el data access application block de Enterprise Library le decimos a la clase DatabaseFactory que lo haga por nosotros, es inversión de control, en lugar de tener nosotros el control sobre la creación de objetos le pedimos alguien más que lo haga. Inyección de dependencias es cuando un objeto A depende de uno B, esta dependencia es inyectada al objeto A por el constructor por ejemplo, es decir, el objeto A tiene un constructor que acepta un parámetro del tipo objeto B, esto es inyectar una dependencia.

Para poder entender el por qué de todo esto vamos a remitirnos a un ejemplo:

El problema:

```
public interface IProcessor
{    void Process();
}
public class ProcessorImp : IProcessor
{
   IDbConnection _connection;
   public ProcessorImp()
   {
      _connection = new System.Data.OleDb.OleDbConnection();
   }
   public void Process()
   {
      //llama a _connection y hace algo...
      _connection.Open();
   }
}
```

Si miramos con detenimiento el método Process vemos algo interesante, imaginemos que queremos probar nuestro ProcessorImp y nos encontramos que en el constructor se instancia _connection, por lo tanto, para correr una prueba vamos a necesitar acceder a la base de datos, eso definitivamente no es deseable y mucho menos práctico para una prueba unitaria, pero ¿cuál es la solución?

La solución, inyectar dependencias:

La forma de solucionar esto es de alguna manera evitar que se instancie _connection dentro de ProcessorImp y utilizarlo en forma transparente de modo que podamos, en caso de probarlo, utilizar una implementación de _connection que no acceda a la base de datos.

Primera aproximación, desacoplar con interfaces

La inyección de dependencias es algo tan simple como evitar que nuestros objetos de negocios creen objetos que necesitan utilizar, proveyéndoselos ya listos para operar. ¿Y cómo hacemos esto? Inyectándolos.

Hay un par de formas de inyectar dependencias, pero nosotros vamos a elegir la inyección de dependencias por constructor (ya que no tiene sentido que exista la clase Processor sin un IDbConnection), que no es más que pasarle a nuestro objeto de negocios ese objeto que necesita ya listo a través de su constructor, más aún, vamos a pasarle una interfaz lo cual nos va a traer otras ventajas adicionales, vamos a ver cómo queda el código con esta modificación:

```
public class ProcessorImp : IProcessor
{
    IDbConnection _connection;
    public ProcessorImp(IDbConnection connection)
    {
        _connection = connection;
    }
    public void Process()
    {
        //llama a _connection y hace algo...
    }
}
```

Un poco mejor, vemos que estamos pasando el IDbConnection en el constructor, lo cual es bueno porque nos va permitir pasarle cualquier cosa que implemente esta interfaz y simule el comportamiento de la capa de datos en caso de querer probarlo en una prueba unitaria. Ahora bien, para construir un objeto IProcessor antes hacíamos:

```
IProcessor processor = new ProcessorImp();
```

Pero ahora no podemos porque el constructor de ProcessorImp (la única clase que tenemos que implementa IProcessor) en su lugar tenemos que hacer esto:

```
IDbConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["default"] .ConnectionString);
```

```
IProcessor processor = new ProcessorImp(connection);
```

Para simplemente después hacer

```
processor.Process();
```

O sea, 1 línea útil y 2 inútiles (para la finalidad de la lógica de negocios), esto puede agravarse en caso de tener más de una dependencia de inyectar o que la dependencia que inyectamos a su vez tenga otras dependencias.

Entonces, ¿cómo se resuelve?, fácil, echando mano de otro patrón de diseño Factory para realizar la inversión de control.

El patrón Factory

Nos va a permitir recuperar objetos listos para ser utilizados, sin necesidad de saber nada de ellos, ni de qué dependen, si necesitan información de configuración, etc.

Mágico, si pero ¿cómo hacemos?, veamos un poco de código:

```
public class ProcessorFactory
{
    public static IProcessor BuildProcessor()
    {
        return new ProcessorImp(
            new System.Data.SqlClient.SqlConnection(
ConfigurationManager.ConnectionStrings["default"] .ConnectionString);
    }
}
```

A esto le llamamos inversión de control (IoC) y es justamente eso, en lugar de decidir nosotros cómo crear un objeto hacemos que alguien más decida por nosotros, o sea, invertimos el control, se usa así:

```
IProcessor processor = ProcessorFactory.BuildProcessor();

processor.Process();
```

Lindo, ahora la creación del objeto no interfiere con la lógica de negocios y no necesitamos saber nada del objeto que vamos a utilizar, no limitamos a saber de la interfaz. Mejorando las pruebas unitarias Esta forma de hacer las cosas nos permite mejorar las pruebas unitarias ya que para probar el ProcessorImp, simplemente creamos un Factory que nos dé un ProcessorImp y le inyecte por constructor una IDbConnection que no haga nada o que haga lo que necesitamos para la prueba más o menos así:

```
public class ProcessorFakeFactory
{
    public static IProcessor BuildProcessor()
    {
        return new ProcessorImp(new FakeDbConnection());
    }
}
```

Otras ventajas de la inversión de control

Otra de las ventajas que tenemos utilizando este diseño es que podemos obtener diferentes implementaciones de IProcessor sin hacer mucho y lo más importante, sin impactar en el código de lógica de negocios. Si recuerdan algo que dije antes, en la interfaz no se encuentra el constructor y esto nos permite con la misma interfaz utilizar indistintamente diferentes implementaciones de IProcessor que pueden tener otras dependencias

Por ejemplo:

```
public class ProcessorFakeFactory
{
    public static IProcessor BuildProcessor()
    {
        return BuildProcessor(ProcessorType.Sql);
    }
    public static IProcessor BuildProcessor(ProcessorType type)
    {
        switch (type)
        {
            case ProcessorType.Sql:
                return new ProcessorImp(
                    new System.Data.SqlClient.SqlConnection(
                    ConfigurationManager.ConnectionStrings["default"].ConnectionString));
                break;
            case ProcessorType.OleDb:
                return new ProcessorImp(
                    new System.Data.OleDb.OleDbConnection(
                    ConfigurationManager.ConnectionStrings["oledb"].ConnectionString));
                break;
            default:
                throw new ArgumentException("type"); break;
        }
    }
    public enum ProcessorType
    {
```

```
        Sql,
        OleDb
    }
}
```

Esta implementación es más o menos lo que hace DataAccess Application Block cuando hacemos DataBaseFactory.CreateDataBase() con una agregado que veremos más adelante.

Se ve claramente las ventajas de éste modelo, disminuimos el acoplamiento, mejoramos las pruebas unitarias, obtenemos mayor flexibilidad y se queda más mantenible.

Paso de parámetros tipo función (delegados)

Un delegado es un tipo de objeto que encapsula un método de forma segura, similar a un puntero a función de C y C++. A diferencia de los punteros a función de C, los delegados están orientados a objetos, proporcionan seguridad de tipos y son seguros. El tipo de un delegado se define por su nombre. El ejemplo siguiente declara un delegado denominado Del que puede encapsular un método que toma un valor String como argumento y devuelve un valor void.

```
public delegate void Del(string message);
```

Generalmente, un objeto de delegado se crea proporcionando el nombre del método que el delegado contendrá o con un método anónimo. Una vez que se crean instancias de un delegado, el delegado pasará al método una llamada realizada por éste al delegado. Los parámetros que el llamador pasó al delegado se pasan al método y el delegado devuelve al llamador el valor devuelto del método, si hay alguno. Esto se conoce como invocar al delegado. Se puede invocar un delegado con instancias como si fuera el propio método contenido.

Por ejemplo:

```
// Create a method for a delegate.
public static void DelegateMethod(string message)
{
    System.Console.WriteLine(message);
}
// Instantiate the delegate.
Del handler = DelegateMethod;
// Call the delegate.
handler("Hello World");
```

Los tipos de delegados se derivan de la clase Delegate en .NET Framework. Los tipos de delegados son sealed (no se pueden derivar) y no es posible derivar clases personalizadas de Delegate. Puesto que el delegado con instancias es un objeto, puede pasarse como parámetro o asignarse a una propiedad. Esto permite que un método acepte un delegado como parámetro y llame al delegado posteriormente. Esto se conoce como devolución de llamada asincrónica y constituye un método común de notificación de un llamador cuando ha finalizado un proceso largo. Cuando se utiliza un delegado de esta forma, no es necesario que el código que utiliza el delegado conozca la implementación del método que se está utilizando. La funcionalidad es similar a la encapsulación que proporcionan las interfaces.

Otro uso común de devoluciones de llamada es definir un método de comparación personalizado y pasar ese delegado a un método de ordenación. Esto permite al código del llamador ser parte del algoritmo de ordenación. El método del ejemplo siguiente utiliza el tipo Del como parámetro:

```
public void MethodWithCallback(int param1, int param2, Del callback)
{
    callback("The number is: " + (param1 + param2).ToString());
}
```

A continuación, se puede pasar el delegado creado anteriormente a ese método.

MethodWithCallback(1, 2, handler);

y recibir el resultado siguiente en la consola.

The number is: 3

Al utilizar el delegado como abstracción, MethodWithCallback no es necesario llamar a la consola directamente; es decir, el delegado no se tiene que diseñar pensando en una consola. Lo que MethodWithCallback hace es simplemente preparar una cadena y pasarla a otro método. Esto es especialmente eficaz, puesto que un método delegado puede utilizar cualquier número de parámetros.

Cuando se crea un delegado para contener un método de instancia, el delegado hace referencia tanto a la instancia como al método. Un delegado no conoce el tipo de instancia a parte del método que éste contiene, de modo que un delegado puede hacer referencia a cualquier tipo de objeto siempre que exista un método en dicho objeto que coincida con la firma del delegado. Cuando se crea un delegado para contener un método estático, éste sólo hace referencia al método. Consideré las siguientes declaraciones.

```
public class MethodClass
{
    public void Method1(string message) { }
    public void Method2(string message) { }
}
```

Junto con el método estático DelegateMethod que se mostró previamente, ahora tenemos tres métodos que la instancia de Del puede contener.

Un delegado puede llamar a más de un método cuando se invoca. Esto se denomina multidifusión. Para agregar un método adicional a la lista de métodos del delegado (lista de invocación), simplemente es necesario agregar dos delegados mediante los operadores de suma o de asignación de suma ('+' o '+=').

Por ejemplo:

```
MethodClass obj = new MethodClass();

Del d1 = obj.Method1;

Del d2 = obj.Method2;

Del d3 = DelegateMethod;

//Both types of assignment are valid.

Del allMethodsDelegate = d1 + d2;

allMethodsDelegate += d3;
```

En este momento, allMethodsDelegate contiene tres métodos en su lista de invocación: Method1, Method2 y DelegateMethod. Los tres delegados originales, d1, d2 y d3 no cambian. Cuando se invoca a allMethodsDelegate, se llama a los tres métodos por orden. Si el delegado utiliza parámetros de referencia, ésta a su vez se pasa secuencialmente a cada uno de los tres métodos y todos los cambios efectuados por un método son visibles para el siguiente método. Cuando alguno de los métodos produce una excepción que no se detecta dentro del método, esa excepción se pasa al llamador del delegado y no se llama a ninguno de los métodos siguientes de la lista de invocación. Si el delegado tiene un valor devuelto y/o fuera de los parámetros, devuelve el valor devuelto y los parámetros del último método invocado. Para quitar un método de la lista de invocación, utilice el operador de resta o de asignación de resta ('-' o '-=').

Por ejemplo:

```
//remove Method1

allMethodsDelegate -= d1;

// copy AllMethodsDelegate while removing d2

Del oneMethodDelegate = allMethodsDelegate - d2;
```

Puesto que los tipos de delegados se derivan de System.Delegate, los métodos y las propiedades definidos por esa clase se pueden llamar en el delegado. Por ejemplo, para buscar el número de métodos en la lista de invocación de un delegado, puede escribir.

```
int invocationCount = d1.GetInvocationList().GetLength(0);
```

Los delegados con más de un método en su lista de invocación derivan de MulticastDelegate, que es una subclase de System.Delegate. El código anterior funciona en ambos casos porque las dos clases admiten GetInvocationList.

Los delegados de multidifusión se utilizan ampliamente en el control de eventos. Los objetos de origen de eventos envían notificaciones de eventos a objetos de destinatario registrados para recibir ese evento. Para registrar un evento, el destinatario crea un método diseñado para controlar el evento, a continuación crea un delegado para dicho método y pasa al delegado al origen de eventos. El origen llama al delegado cuando se produce el evento. Luego el delegado llama al método de control de eventos del destinatario y entrega los datos del evento. El origen de eventos define el tipo de delegado para un evento dado.

La comparación de delegados de dos tipos distintos asignados en tiempo de compilación producirá un error de compilación. Si las instancias de delegado son estáticamente del tipo System.Delegate, se permite la comparación, pero se devolverá false en tiempo de ejecución.

Por ejemplo:

```
delegate void Delegate1();

delegate void Delegate2();

static void method(Delegate1 d, Delegate2 e, System.Delegate f)

{

    // Compile-time error.

    //Console.WriteLine(d == e);


    // OK at compile-time. False if the run-time type of f

    //is not the same as that of d.

    System.Console.WriteLine(d == f);

}
```

Interfaces

Las interfaces se definen mediante la palabra clave interface, como se muestra en el ejemplo siguiente:

```
interface IEquatable<T>

{

    bool Equals(T obj);

}
```

Las interfaces describen un grupo de funcionalidades relacionadas que pueden pertenecer a cualquier elemento class o struct. Las interfaces pueden estar compuestas de métodos, propiedades, eventos, indizadores o cualquier combinación de estos cuatro tipos de miembros. Una interfaz no puede contener campos. Los miembros de interfaz son automáticamente públicos.

Cuando se dice que una clase o estructura hereda una interfaz, significa que la clase o la estructura proporcionan una implementación para todos los miembros que define la interfaz. La propia interfaz no proporciona ninguna funcionalidad que una clase o estructura pueda heredar de la manera en que se puede heredar la funcionalidad de una clase base. Sin embargo, si una clase base implementa una interfaz, la clase derivada hereda esta implementación.

Al igual que las clases se pueden heredar de una clase base o estructura, las clases y estructuras se pueden heredar de interfaces, con dos excepciones:

- Una clase o estructura puede heredar más de una interfaz.
- Cuando una clase o la estructura hereda una interfaz, hereda sólo los nombres de método y las firmas, ya que la propia interfaz no contiene ninguna implementación.

Por ejemplo:

```
public class Car : IEquatable<Car>
{
    public string Make {get; set;}
    public string Model { get; set; }
    public string Year { get; set; }

    // Implementation of IEquatable<T> interface
    public bool Equals(Car car)
    {
        if (this.Make == car.Make &&
            this.Model == car.Model &&
            this.Year == car.Year)
        {
            return true;
        }
        else
            return false;
    }
}
```

Para implementar un miembro de interfaz, el miembro correspondiente de la clase debe ser público, no estático y tener el mismo nombre y la misma firma que el miembro de interfaz. Las propiedades e indizadores de una clase pueden definir descriptores de acceso adicionales para una propiedad o indizador definidos en una interfaz.

Por ejemplo, una interfaz puede declarar una propiedad con un descriptor de acceso get, pero la clase que implementa la interfaz puede declarar la misma propiedad con descriptores de acceso get y set. Sin embargo, si la propiedad o el indizador utiliza una implementación explícita, los descriptores de acceso deben coincidir.

Las interfaces y los miembros de interfaz son abstractos; las interfaces no proporcionan una implementación predeterminada

La interfaz IEquatable<(Of <(T>)>) informa al usuario del objeto de que éste puede determinar si es igual que otros objetos del mismo tipo; el usuario de la interfaz no necesita saber cómo se implementa.

Las interfaces pueden heredar otras interfaces. Es posible que una clase herede una interfaz varias veces, a través de las clases base o interfaces que hereda. En ese caso, la clase sólo puede implementar la interfaz una vez, siempre que ésta se declare como parte de la nueva clase. Si la interfaz heredada no está declarada como parte de la nueva clase, la clase base que la declaró proporcionará su implementación. Es posible que una clase base implemente miembros de interfaz a través de miembros virtuales. En ese caso, la clase que hereda la interfaz puede cambiar el comportamiento de la interfaz reemplazando los miembros virtuales.

Información general sobre interfaces

Una interfaz tiene las siguientes propiedades:

- Una interfaz es como una clase base abstracta: cualquier tipo no abstracto que hereda la interfaz debe implementar todos sus miembros.
- No se pueden crear instancias directamente de una interfaz.
- Las interfaces pueden contener eventos, métodos, indizadores y propiedades.
- Las interfaces no contienen implementaciones de métodos.
- Las clases y estructuras se pueden heredar de más de una interfaz.
- Una interfaz se puede heredar de varias interfaces.

Expresiones Lambda

Una expresión lambda es una función anónima que puede contener expresiones e instrucciones y se puede utilizar para crear delegados o tipos de árboles de expresión.

Todas las expresiones lambda utilizan el operador lambda = >, que se lee como "va a". El lado izquierdo del operador lambda especifica los parámetros de entrada (si existe alguno), mientras que el lado derecho contiene el bloque de expresiones o instrucciones. La expresión lambda x => x * x se lee "x va a x veces x". Esta expresión se puede asignar a un tipo de delegado del siguiente modo:

```
delegate int del(int i);

del myDelegate = x => x * x;

int j = myDelegate(5); //j = 25
```

Para crear un tipo de árbol de expresión:

```
using System.Linq.Expressions;

// ...

Expression<del> = x => x * x;
```

El operador => tiene la misma prioridad que la asignación (=) y es asociativo por la derecha.

Las expresiones lambda se utilizan en consultas de LINQ basadas en métodos como argumentos para métodos de operador de consulta estándar, tales como Where y Where.

Cuando se utiliza la sintaxis de método para llamar al método Where en la clase Enumerable (como se hace en LINQ to Objects y LINQ to XML), el parámetro es un tipo delegado System..::.Func<(Of <(T, TResult>)>). Una expresión lambda constituye la manera más práctica de crear ese delegado. Cuando se llama al mismo método en, por ejemplo, la clase System.Linq..::.Queryable (como se hace en LINQ to SQL), el tipo de parámetro es System.Linq.Expressions..::.Expression<Func>, donde Func es cualquier delegado de Func que tenga hasta cinco parámetros de entrada. De nuevo, una expresión lambda constituye una manera muy concisa de construir ese árbol de expresión. Las expresiones lambda permiten que las llamadas a Where tengan un aspecto similar, aunque, de hecho, el tipo de objeto creado desde la expresión lambda sea diferente.

En el ejemplo anterior, observe que la firma de delegado tiene un parámetro de entrada con tipo implícito int y devuelve un int. La expresión lambda se puede convertir en un delegado de ese tipo porque también tiene un parámetro de entrada (x) y un valor devuelto que el compilador puede convertir implícitamente al tipo int. (La inferencia de tipo se analiza con más detalle en las siguientes secciones.) Cuando el delegado se invoca utilizando un parámetro de entrada de 5, devuelve un resultado de 25.

Las expresiones lambda no se permiten en el lado izquierdo del operador is o as.

Todas las restricciones que se aplican a los métodos anónimos también se aplican a las expresiones lambda.

Lambdas de expresión

Una expresión lambda con una expresión en el lado derecho se denomina lambda de expresión. Las lambdas de expresión se utilizan ampliamente en la construcción de Árboles de expresiones. Una lambda de expresión devuelve el resultado de la expresión y presenta la siguiente forma básica:

(input parameters) => expression

Los paréntesis sólo son opcionales si la lambda tiene un parámetro de entrada; de lo contrario, son obligatorios. Dos o más parámetros de entrada se separan por comas y se encierran entre paréntesis:

(x, y) => x == y

A veces, es difícil o imposible para el compilador deducir los tipos de entrada. Cuando esto ocurre, puede especificar los tipos explícitamente como se muestra en el ejemplo siguiente:

(int x, string s) => s.Length > x

Para especificar cero parámetros de entrada, utilice paréntesis vacíos:

() => SomeMethod()

Observe en el ejemplo anterior que el cuerpo de una lambda de expresión puede estar compuesto de una llamada a método. Sin embargo, si va a crear árboles de expresión que se utilizarán en otro dominio, como SQL Server, no debería utilizar llamadas a métodos en expresiones lambda. Los métodos no tendrán ningún significado fuera del contexto de .NET Common Language Runtime.

Lambdas de instrucciones

Una lambda de instrucciones es similar a un lambda de expresión, salvo que las instrucciones se encierran entre llaves:

(input parameters) => {statement;}

El cuerpo de una lambda de instrucciones puede estar compuesto de cualquier número de instrucciones; sin embargo, en la práctica, generalmente no hay más de dos o tres.

delegate void TestDelegate(string s);

…

```
TestDelegate myDel = n => { string s = n + " " + "World"; Console.WriteLine(s); };

myDel("Hello");
```

Las lambdas de instrucciones, como los métodos anónimos, no se pueden utilizar para crear árboles de expresión.

Lambdas con los operadores de consulta estándar

Muchos operadores de consulta estándar tienen un parámetro de entrada cuyo tipo es uno de la familia Func<(Of <(T, TResult>)>) de delegados genéricos. El delegado Func<(Of <(T, TResult>)>) usa parámetros de tipo para definir el número y tipo de parámetros de entrada, y el tipo de valor devuelto del delegado. Los delegados Func son muy útiles para encapsular expresiones definidas por el usuario que se aplican a cada elemento en un conjunto de datos de origen. Por ejemplo, considere el siguiente tipo delegado:

```
public delegate TResult Func<TArg0, TResult>(TArg0 arg0)
```

Se pueden crear instancias del delegado como Func<int,bool> myFunc, donde int es un parámetro de entrada y bool es el valor devuelto. El valor devuelto siempre se especifica en el último parámetro de tipo. Func<int, string, bool> define un delegado con dos parámetros de entrada, int y string, y un tipo de valor devuelto bool. El siguiente delegado Func, cuando se invoca, devolverá verdadero o falso para indicar si el parámetro de entrada es igual a 5:

```
Func<int, bool> myFunc = x => x == 5;

bool result = myFunc(4); // returns false of course
```

También puede proporcionar una expresión lambda cuando el tipo de argumento es Expression<Func>, por ejemplo, en los operadores de consulta estándar que se definen en System.Linq.Queryable. Al especificar un argumento Expression<Func>, la expresión lambda se compilará en un árbol de expresión.

A continuación, se muestra un operador de consulta estándar, el método Count:

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

int oddNumbers = numbers.Count(n => n % 2 == 1);
```

El compilador puede deducir el tipo del parámetro de entrada, o también se puede especificar explícitamente. Esta expresión lambda particular cuenta aquellos enteros (n) que divididos por dos dan como resto 1.

El método siguiente generará una secuencia que contiene todos los elementos de la matriz de números que aparecen antes del "9", ya que ése es el primer número de la secuencia que no cumple la condición:

```
var firstNumbersLessThan6 = numbers.TakeWhile(n => n < 6);
```

Este ejemplo muestra cómo especificar varios parámetros de entrada encerrándolos entre paréntesis. El método devuelve todos los elementos de la matriz de números hasta encontrar un número cuyo valor sea menor que su posición. No confunda el operador lambda (=>) con el operador mayor o igual que (>=).

```
var firstSmallNumbers = numbers.TakeWhile((n, index) => n >= index);
```

Deducción de tipos en las expresiones lambda

Al escribir expresiones lambda, no tiene por qué especificar un tipo para los parámetros de entrada, ya que el compilador puede deducir el tipo según el cuerpo de la lambda, el tipo de delegado subyacente y otros factores que se describen en la especificación del lenguaje C# 3.0. Para la mayoría de los operadores de consulta estándar, la primera entrada es el tipo de los elementos en la secuencia de origen. Así, si está realizando una consulta sobre un IEnumerable<Customer>, se deducirá que la variable de entrada será un objeto Customer, lo cual significa que se podrá tener acceso a sus métodos y propiedades:

```
customers.Where(c => c.City == "London");
```

Las reglas generales para las expresiones lambda son las siguientes:

- La lambda debe contener el mismo número de parámetros que el tipo delegado.
- Cada parámetro de entrada en la lambda se debe poder convertir implícitamente a su parámetro de delegado correspondiente.
- El valor devuelto de la lambda (si existe) se debe poder convertir implícitamente al tipo de valor devuelto del delegado.

Observe que las expresiones lambda, en sí mismas, no tienen tipo, ya que el sistema de tipos comunes no tiene ningún concepto intrínseco de "expresión lambda". Sin embargo, a veces resulta práctico hablar informalmente del "tipo" de una expresión lambda. En estos casos, el tipo hace referencia al tipo del delegado o el tipo de Expression en el que se convierte la expresión lambda.

Las reglas siguientes se aplican al ámbito de variables en expresiones lambda

- Una variable que se captura no será eliminada por el recolector de elementos no utilizados hasta que el delegado que hace referencia a ella quede fuera del ámbito.
- Las variables introducidas dentro de una expresión lambda no son visibles en el método externo.
- Una expresión lambda no puede capturar directamente un parámetro ref o out desde un método que la englobe.
- Una instrucción return en una expresión lambda no hace que el método que la engloba vuelva.
- Una expresión lambda no puede contener una instrucción goto, una instrucción break o una instrucción continue cuyo destino esté fuera del cuerpo o en el cuerpo de una función anónima contenida.

Server Side Transparente

Como objetivo adicional queremos aislar la implementación del lado del cliente de la implementación del lado del servidor para poder reutilizarlo.

Así la parte del cliente (que es la más difícil, compleja y variable) es única y estándar, mientras el lado del servidor puede ser intercambiable. Esto es para poder atender las dos plataformas de desarrollo estándar de facto actuales: .NET y Java con una nomenclatura unificada para nuestros desarrolladores.

El objetivo es lograr una sintaxis unificada que permita a nuestros desarrolladores construir aplicaciones usando un cuerpo de conocimiento unificado para ambas plataformas.

Se considera comenzar la versión para Java una vez que validemos la versión para .NET, por eso no se ahondará en este punto, sin embargo, el diseño de la arquitectura considera este punto.

Capítulo 1 Consolidación de framework y macros

Definición del framework

En el primer mes de desarrollo se ha definido la arquitectura del nuevo Framework, previendo, conceptualizando y pensando en próximas actualizaciones que va a tener esta nueva versión.

Se realizo una revisión a profundidad, evaluando todas aquellas rutinas, que son apropiadas para poder reutilizar o en su defecto para que sirvan de base para reestructurar la función.

La revisión del framework pasado, trajo como ventaja comparar los conocimientos que anteriormente se tenían contra los nuevos conocimientos aprendidos para desarrollar este nuevo framework.

Para empezar a servir las peticiones de todas las nuevas operaciones se creó una base de datos provisional.

Manejo de credenciales

Se diseño una página contenedora que será la encargada de cargar todas las partes de la página, donde se ha construido provisionalmente una página de login y se ha realizado las operaciones siguientes:

- Operación de validación de credenciales
    - Manejo de credenciales cuando estén bien
    - Manejo de credenciales cuando las credenciales estén erróneas.
- Si la validación es correcta sirve otra pagina HTML como respuesta
- Si la validación es incorrecta manda un mensaje de error.

Nota: esta operación de validación de credenciales no se conecta a la base de datos

Sesiones

Se ha diseñado el mecanismo de manejo de sesiones y elementos de seguridad que serán parte de la validación de sesiones y evitaran que se duplique una sesión.

Dentro de los elementos que se manejan para la seguridad de las sesiones se definieron las siguientes variables:

- SessionID.
- CHA.
- Folio.

Se toman estas variables por que tienen diferentes usos

SessionID: es la variable que guarda el id de la sesión, este se mantiene fijo en todo el tiempo que el usuario esta logueado en la aplicación.

CHA: esta variable cambia tanto del lado del servidor, como del lado del cliente en base de un algoritmo definido, con forme se realiza la comunicación entre ambos lados.

Folio: esta es otra variable de seguridad que tiene como fin almacenar un numero consecutivo en cada operación para validar que la secuencia de operaciones venga de la misma fuente.

En conjunto estas tres variables son la base de la seguridad en las sesiones, para hacer cada una de ellas irrepetibles, evitar que dos usuarios estén firmados con las mismas credenciales y evitar que algún agente extraño tenga el poder de hacer peticiones o solicitudes al servidor.

Y en cuanto se detecta alguna inconsistencia en estas variables la sesión de da por finalizada la sesión y automáticamente se bloque ese usuario.

Diseño del área de trabajo

Se diseño y se estableció un área de trabajo de cómo se van a generar las plantillas de las aplicaciones. De esta manera:

- Un área en la parte derecha donde se creara un menú dependiendo del usuario que se esté firmando.
- Un área de trabajo que cambiara de acuerdo a lo que el usuario requiera.
- Zona de prueba que servirá para cargar datos y presentar información.
- Un área de depuración (Que solo se presentara para el proceso de desarrollo), en esta área se mostraran todos los errores con su descripción que mande el servidor.

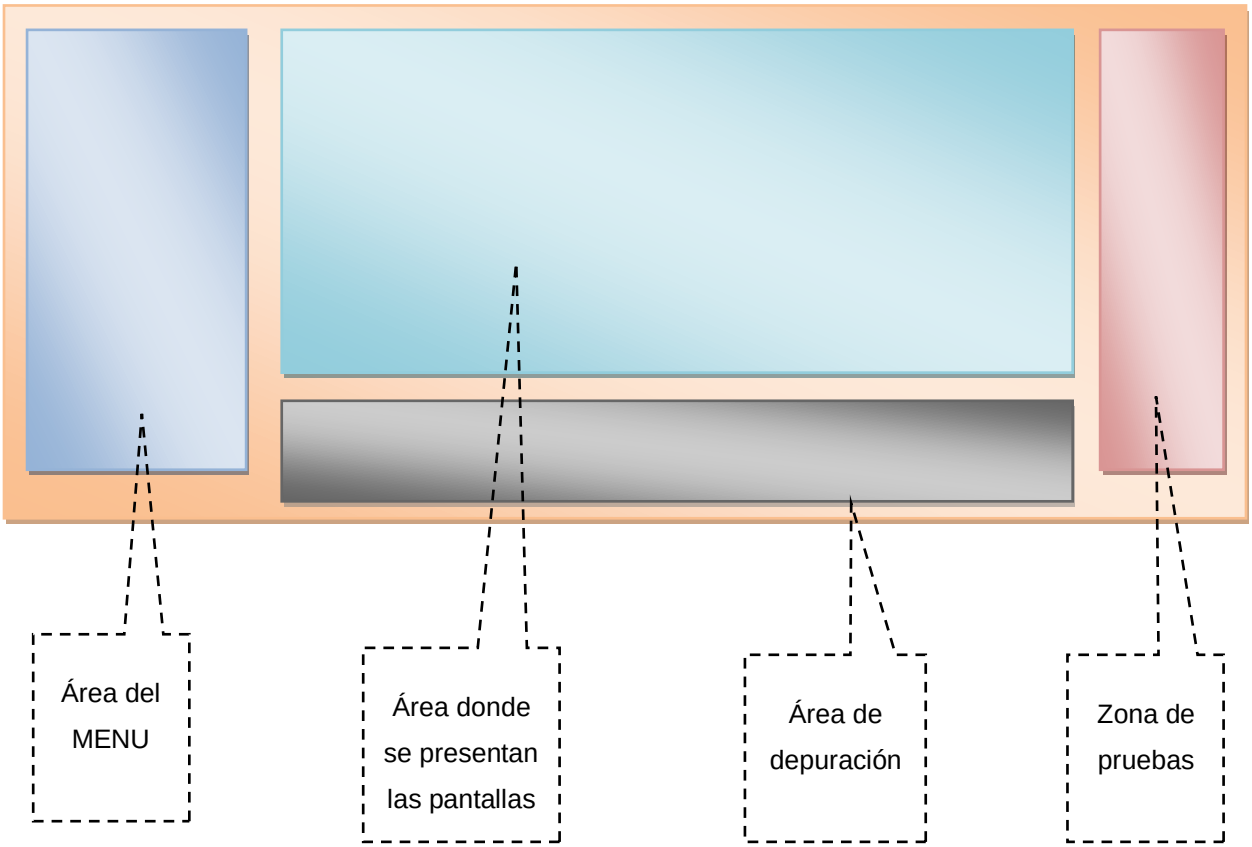

Figura 22.- Diagrama de distribución de la aplicación web.

Esa en esencia es la estructura base que se tomara en cuenta para la generación de todas las aplicaciones que sean generadas por nuestra herramienta.

Claro que este diseño queda sujeto a cambios y es posible que cada quien personalice de una manera sencilla las plantillas de cada aplicación, sin que se vea afectado el rendimiento o funcionalidad de la aplicación.

Consolidación del DAL (DATA ACCESS LAYER [Capa de acceso de datos])

DAL es una capa que se utiliza en los programas de computación para proveer simplicidad al acceso de los datos almacenados en un medio fijo, son utilizados en el modelo entidad-relación en base de datos.

Este método da la facilidad de poder tener un soporte para múltiples tipos de manejadores de base de datos y escoger la mejor opción según la necesidad o petición del cliente.

Con esto se definió el mecanismo y la lógica con la que se codifico esta capa de acceso de datos.

Las capas que se obtuvieron en la definición para resolver las operaciones son:

- SQLConnection.
- SQLPool.
- SQLBuilder.

Se estableció como se va a manejar el pool de conexiones de la base de datos.

Se programo esta capa de tal manera que las conexiones a la base datos se completamente independiente de las operaciones especificas de cada manejador de base de datos.

Como se muestra en la siguiente representación:

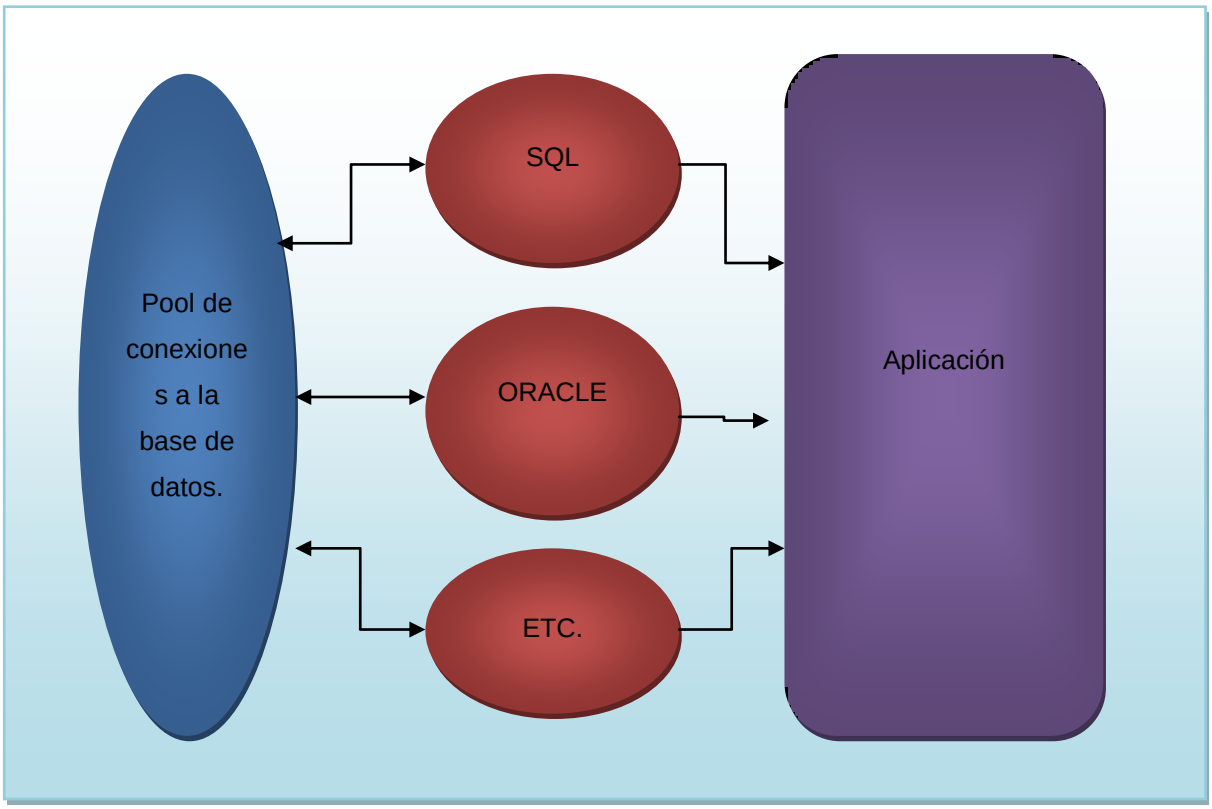

Figura 23.- Representación del pool de conexiones a diferentes bases de datos.

BMO (Business Meta Object [Meta Objetos de Negocio])

Son objetos que funcionan como plantillas de información que le sirven a objetos particulares de negocio para explotar información de forma genérica.

Este concepto es la aplicación del concepto de Metadata a la aplicación de reglas de negocio para formar un engine.

Se definieron de una manera muy básica tanto como para el servidor como para el lado del cliente el BMO.

Y se definieron los siguientes objetos.

- TbaxClass
- TbaxAttribute

UI(User Interfaz [Interfaz de Usuario])

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

Se empezó a crear el caso de uso de la ficha o forma.

Y se crearon unos componentes que se puedan utilizar para capturar los datos del tipo:

- Entero
- Número con decimales
- Moneda
- Fecha
- String
- Lógico
- String con Combo
- Texto

Se creó un mecanismo en JS que valide los campos de acuerdo al tipo de dato que se está capturando.

Se diseño una instancia de clase llamada Persona y se asociaron los controles con los campos de la instancia.

JS(Java Script)

Se creó TPersona.js que es la representación de la clase del lado del cliente.

Esta clase se cargo y se depuro.

Para TbaxClass.js se separo este archivo en 3 principales partes con el fin de encapsular el comportamiento específico.

Ordenar los JS para permitir intercambiar librerías de 3os

Remplazo de macros

Una macro es una abreviatura de macroinstrucción, es una serie de instrucciones que se almacenan para que se puedan ejecutar de forma secuencial mediante una sola llamada u orden de ejecución.

Se empezó a crear el mecanismo que remplazara macros dentro de la aplicación, con el fin de hacerla más dinámica.

Y se aplico este mecanismo para:

- Reemplazo de Macros para e ambiente de depuración y
- Reemplazo de Macros para un ambiente de producción.

Se definió la sintaxis con la que vamos a crear las macros la cual se vera de la siguiente manera:

%%Macro_a_reeplazar%% los "%%" indican la presencia de una macro.

Por ejemplo:

<html>

<head>

<title>%%Titulo%%</title>

</head>

<body>

%%Cuerpo%%

</body>

</html>

%%Titulo%% será substituido por el nombre de la aplicación definido por un archivo de configuración.

%%Cuerpo%% podrá ser reemplazada por un pagina html o en su defecto por texto dependiendo de la necesidad

Capítulo 2 Reglas de negocio

Reglas de negocio para los atributos.

Una regla de negocio es la que describe las políticas, normas, operaciones, definiciones y restricciones presentes en una organización y que son de vital importancia para alcanzar los objetivos misionales.

Por ejemplo: el cliente puede solicitar una ficha y esa ficha puede solicitar la siguiente información:

- Nombre
- Año de nacimiento
- Sexo
- Y un botón de Siguiente

Y nos puede especificar que para el campo nombre únicamente podamos capturar datos de tipo cadena, para el campo Año de nacimiento solo números, para sexo sea un combo donde solo se pueda seleccionar Masculino o Femenino, y por ultimo para que el usuario continúe valide que sea mayor de 18 años considerando el año actual contra el año capturado.

Todas estas especificaciones son reglas de negocio propias de esta ficha y solo pueden aplicar para este cliente en específico.

Por lo que abstrajimos este concepto y establecimos que estas reglas de negocio pueden ser configurables a través de archivos de configuración que aplican para cada caso de uso en específico.

Se hacen las primeras pruebas para los archivos de configuración que definen reglas de negocio sobre las clases.

Se especifica un formato unificado que llevaran los archivos de configuración que definirán las reglas de negocio.

Se realizo el mecanismo de cómo se va a cargar del lado del servidor las reglas de negocio al tiempo que la UI se carga de manera paralela.

Se definió cual va a ser el orden en que se van a cargar las reglas de negocio desde las más genéricas a las mas especificas sin mezclarlas.

Validación de los datos de manera asíncrona

La transmisión asíncrona se da lugar cuando el proceso de sincronización entre emisor y receptor se realiza en cada palabra de código transmitido.

Se hizo una ficha de captura provisional y se capturaron datos mismos que se validaban conforme se cambiaba de componente de diferentes maneras:

- Se seleccionaba con el mouse otro componente.
- Se cambiaba de componente con el botón de tabulador.
- Mediante la petición de un botón.

Del lado del Cliente.

Se crearon las primeras mascaras para los componentes de acuerdo al tipo de dato que estas iban a procesar:

- Numérico
- Fecha
- Texto
- Booleano

Estos de manera genérica

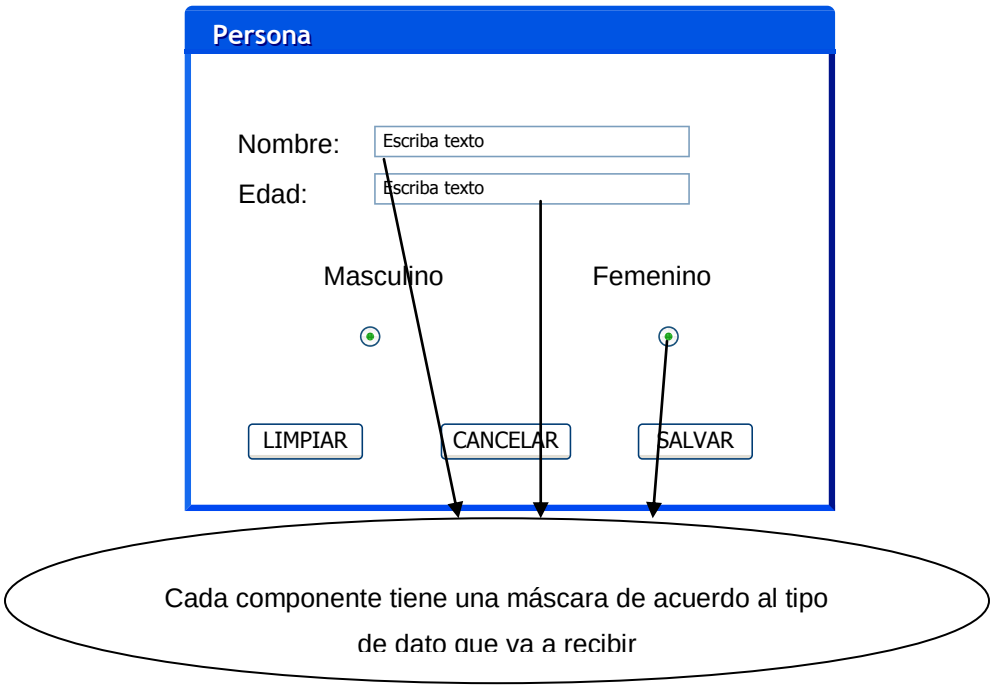

Figura 24.- Ejemplo de cómo aplican las mascaras de acuerdo al componente web.

Del lado del servidor

Se crea un Metadata de los atributos de la clase donde cada atributo tiene un tipo de dato y se valida que el dato que esté recibiendo el servidor.

También se realizo un manejo de errores si del lado del manejador de base datos a la hora de insertar un dato tiene problemas y la inserción no se lleva a cabo con éxito.

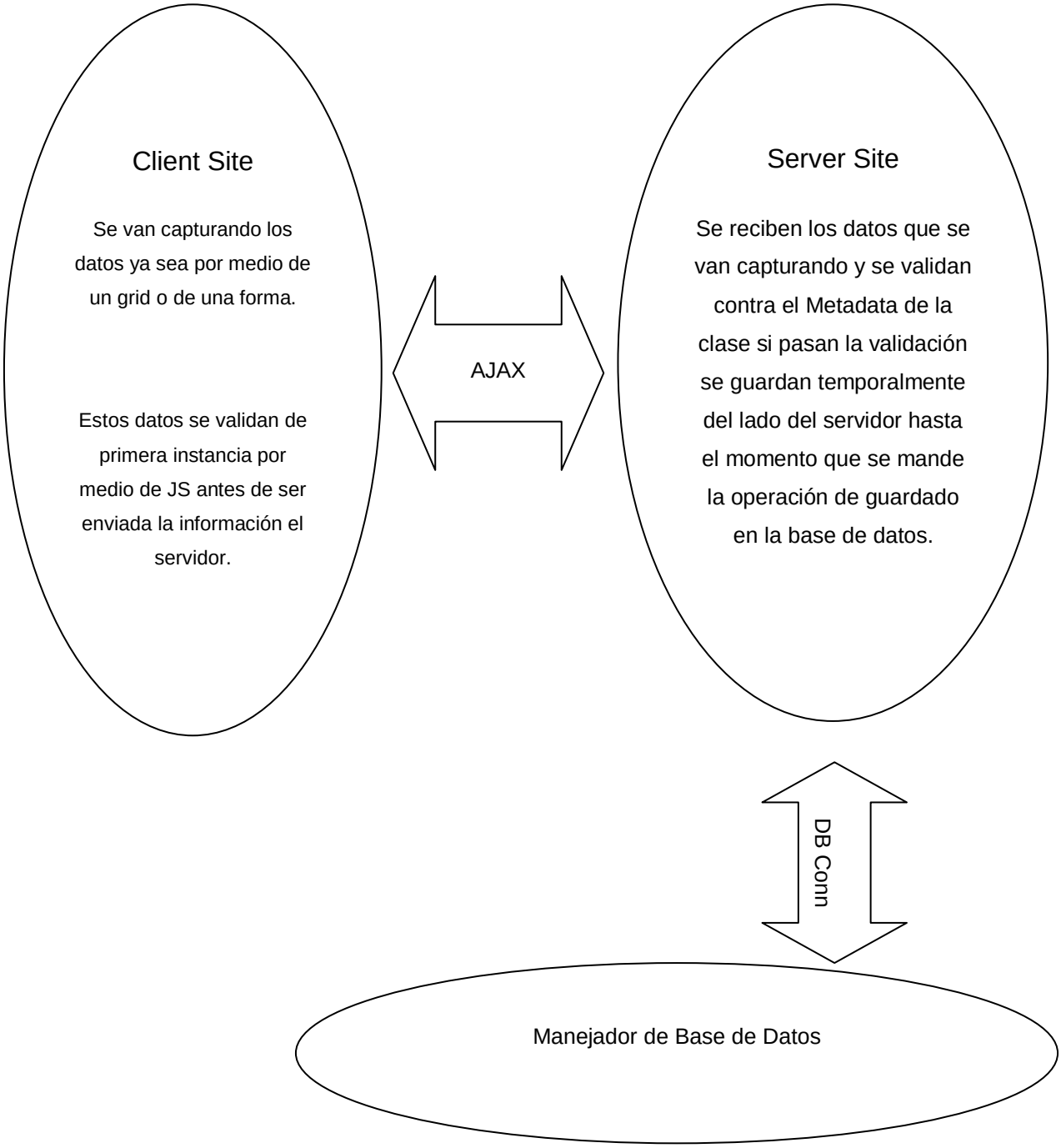

Figura 25.- Diagrama de operaciones del servidor.

Modelo de la validación y comunicación entre el CS, SS, y el MBD

La comunicación entre el lado del servidor y el lado del cliente se lleva a cabo por Ajax.

De igual manera ya definimos el método de comunicación del lado del servidor con el Manejador de Base de Datos.

Generalizando la comunicación y las operaciones con el fin de preparar el framework a diferentes Manejadores de Base de Datos por ejemplo: DB2, MySql, Oracle, etc.

Nota: por ahora toda la comunicación que se va a establecer a la base de datos será a un manejador SQL Server 2008

Todas las operaciones que se mandan del lado de cliente como guardar un registro, borrar, editar etc. Son atendidas por el lado del servidor, que dispara diferentes eventos para resolver las peticiones del cliente.

De esta manera se resuelve con éxito operaciones como:

- Recuperación un registro desde la base de datos para visualizarlo del lado del Cliente.
- Guardado de un registro.
- Edición de un registro.
- Borrado de un registro
- Cancelación de una edición.
- Cancelación de una inserción.
- Que persista un dato del lado del servidor para preparar el salvado de un registro.

Todas estas operaciones ya reflejan cambios directamente a la base de datos.

Y se ha creado código tanto del lado del Cliente en JS y HTML que manda las peticiones por Ajax, como del lado del Servidor en código C#.

Validaciones de los campos

Se resolvió una gran parte de las siguientes operaciones:

- Requerido.
- No duplicado.
- Ejecutar código C#.
- Inserción de 1 registro.
- Implementar código de inserción del CS.
- Editar los valores del registro, campo por campo, con validaciones del lado del cliente.

Las validaciones de los campos al igual que la de los datos se realizo de manera asíncrona con el fin de darle mejor performance al lado del cliente.

Información de la BD

Se ha obtenido el Metadata de la BD para decidir la construcción de queries.

Para cada clase obtuvimos la siguiente información:

- Índices y su composición.
- Número de renglones.

Disparar éste proceso en 2o plano.

Capítulo 3 Diseño y creación de la IU

Investigación del Framework ExtJs de java script

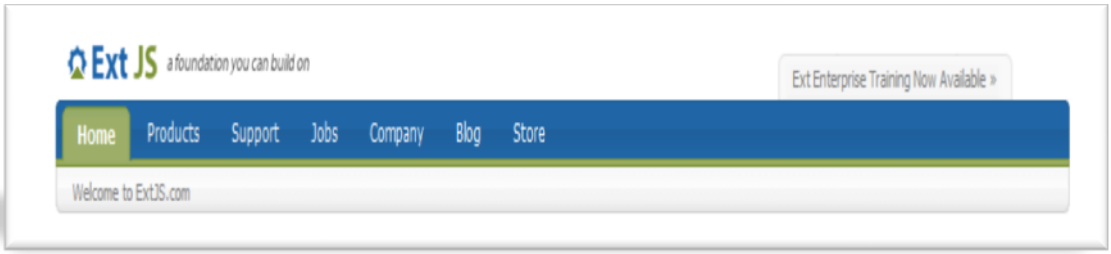

Figura 26.- Menú principal de la página de ExtJS.

ExtJs es un framework que permite hacer interfaces bastante poderosas y amigables para sistemas sobre internet, lo más interesante de esto es que es Cross-browser. En una palabra no ocupas hacer diferentes programaciones para los múltiples browsers o navegadores. ExtJs soporta:

- Internet Explorer
- FireFox
- Safari
- Opera

Algunas de las cualidades de ExtJs es:

- Alto rendimiento
- Interfaces personalizables
- Muy buena arquitectura que permite extender los componentes gracias a su buen modelado
- API intuitiva

- Licencia comercial y open source (gratis)

Se investigo cual era la última versión disponible y estable de ExtJS; se bajo su documentación con ejemplos.

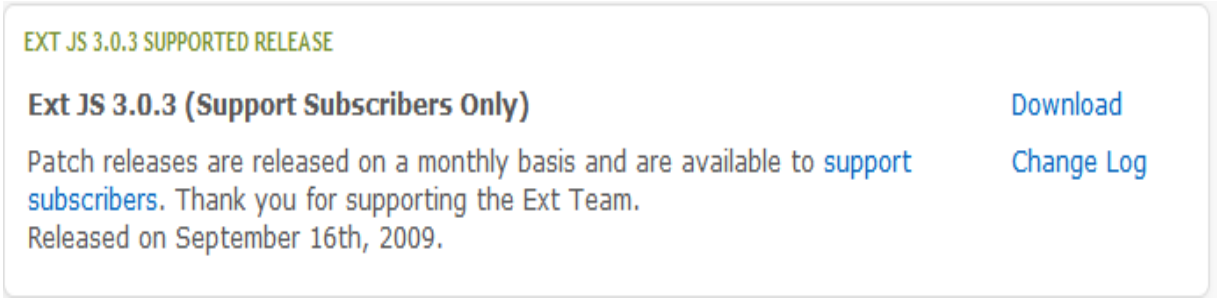

Figura 27.- Versión mas reciente de ExtJs.

Con esto se tomo un tiempo para entender los componentes y ver cuáles podrían ser utilizados para el desarrollo del Framework propio.

Se están haciendo prototipos funcionales respecto a una aplicación basada en la aplicación de pruebas para que estos componentes se empiecen a integrar como parte del Framework

Como por ejemplo:
Grids

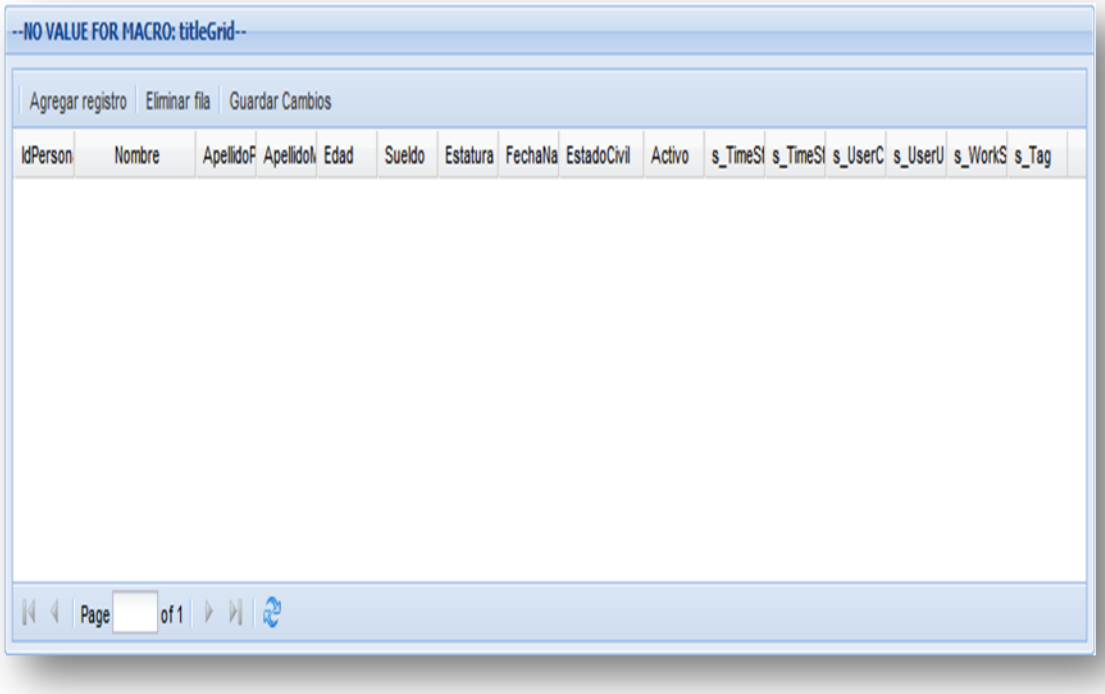

Figura 28.- Ejemplo de un grid.

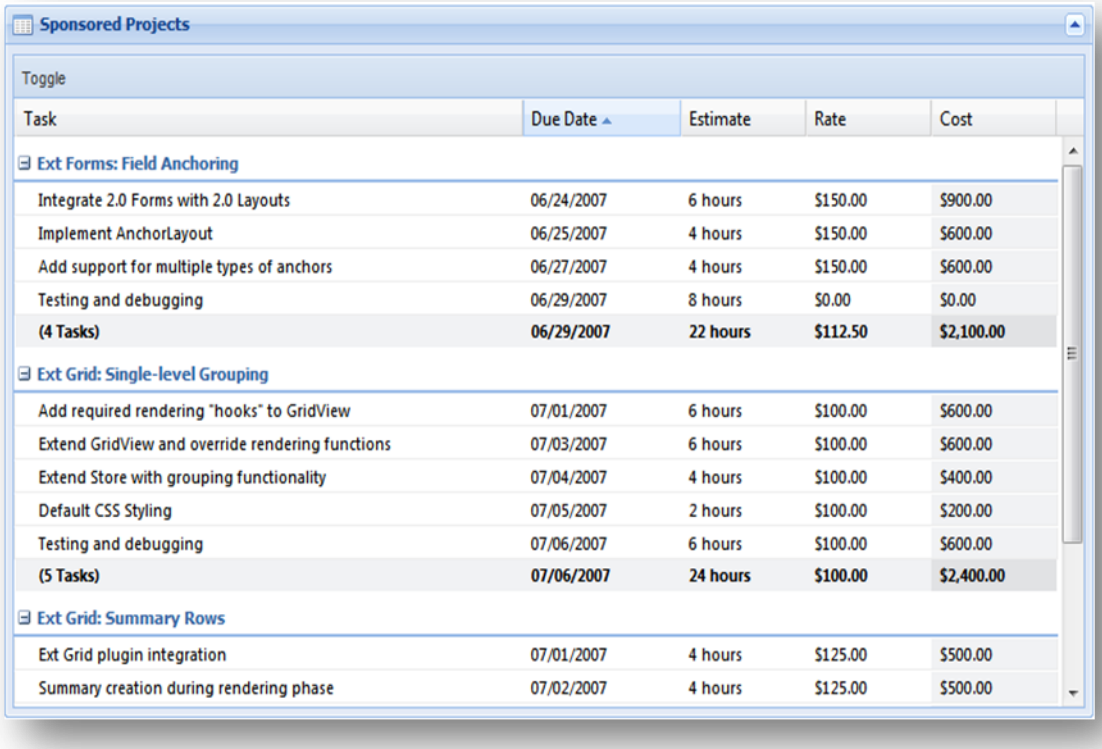

Figura 29.- Ejemplo de grid 2.

Capas para distribuir elementos en el navegador

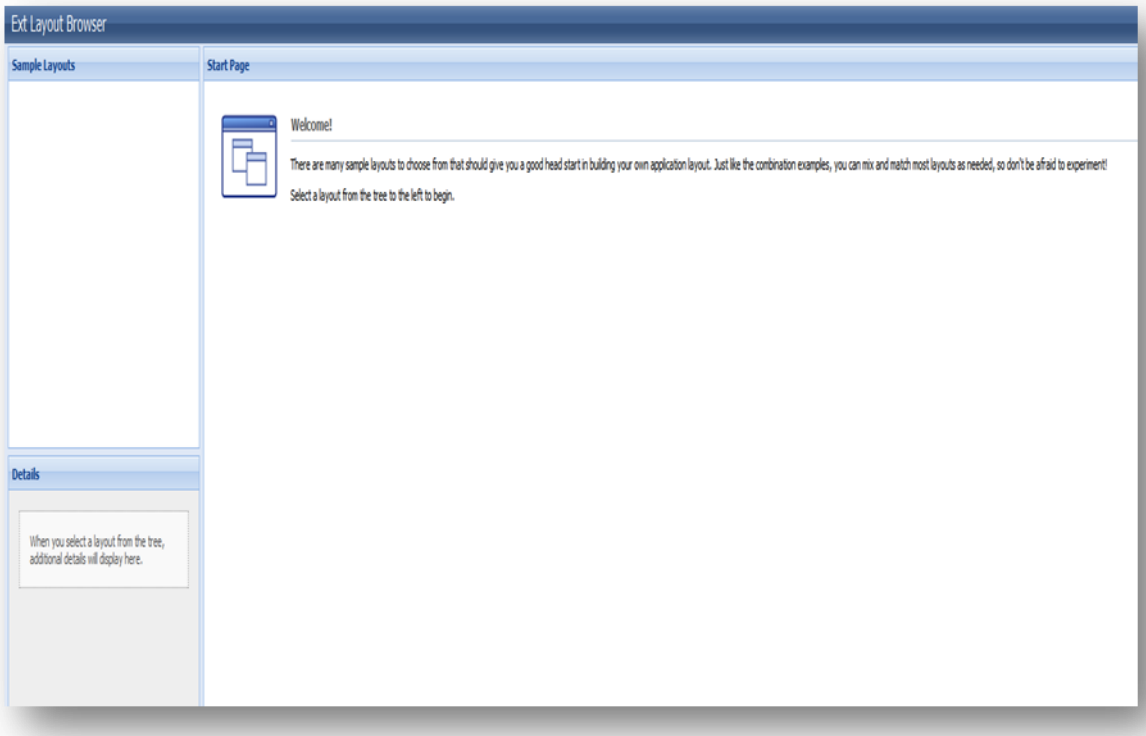

Figura 30.- Ejemplo de layout.

Calendarios

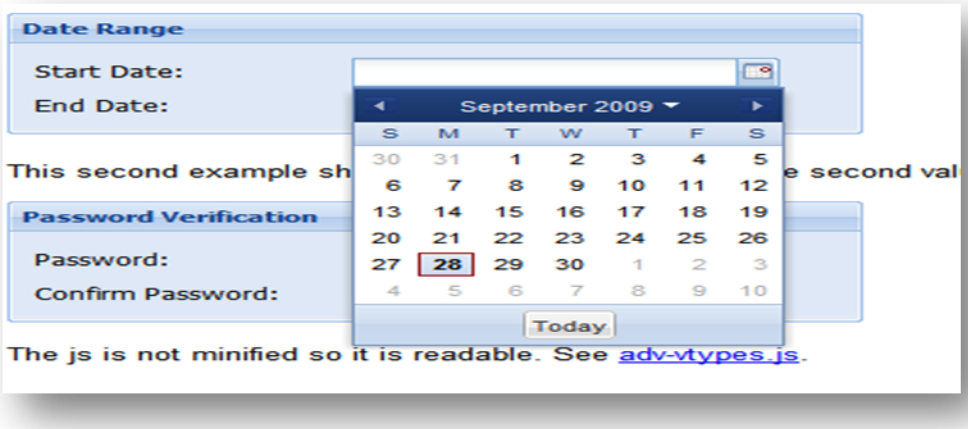

Figura 31.- Ejemplo de calendario.

Formas con varios componentes

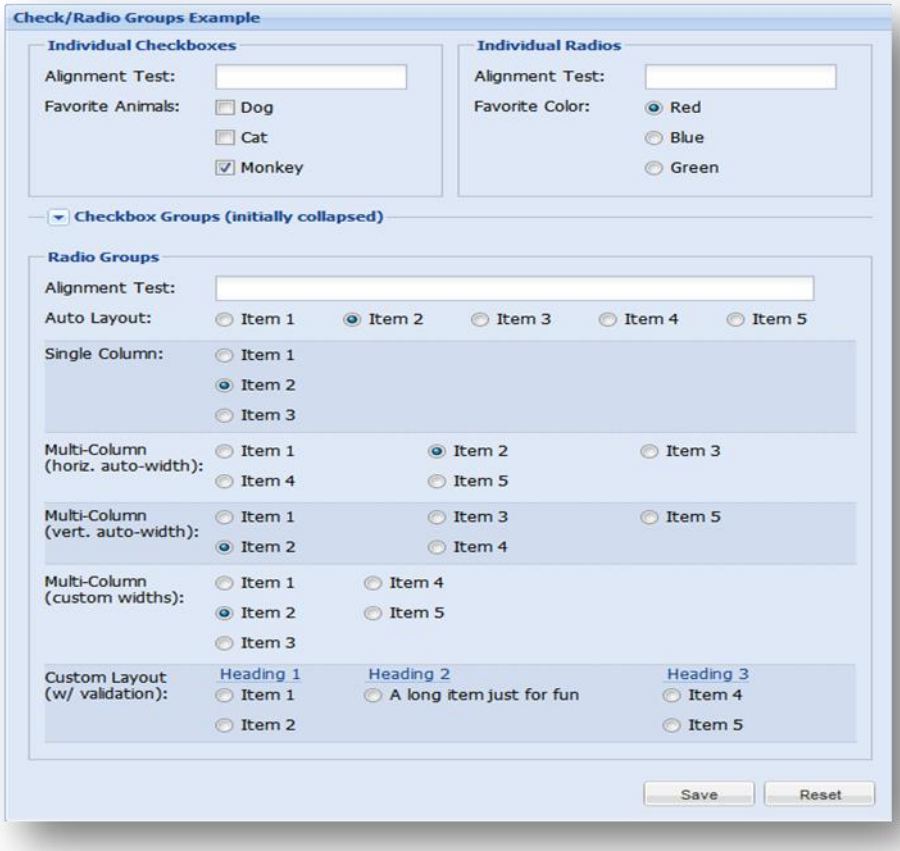

Figura 32.- Ejemplo de formas.

Menús

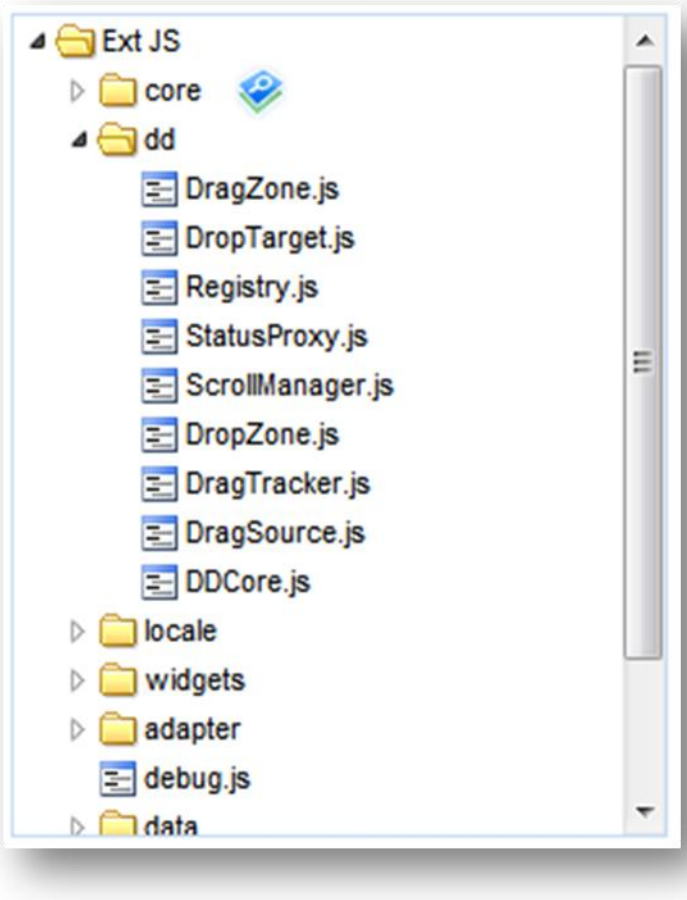

Figura 33.- Ejemplo de menú.

Entre otros componentes como botones, pestañas, arboles, etc. que formaran parte de las formas, de los grids, y de la aplicación y su entono.

Se está implementando la página principal con ExtJS y a su vez se está normalizando la aplicación para usar la librería propia de acciones de JavaScript especificada por nosotros.

Se realiza la investigación de cómo resuelve las siguientes operaciones ExtJS:

- Comunicación por AJAX con ExtJS.
- Como cargar contenido en div fijos usando AJAX.
- Valoración de que método utilizar para la comunicación ya sea el del código propietario o por ExtJS.

Se implemento el login en la página principal usando carga de contenido sin el uso de iframes y después de validar el login y tener las credenciales correctas se carga el menú de ExtJS; también se creó un caso de uso sencillo y se cargo en el WorkArea desde un link en el menú.

Se hacen la primeras pruebas con la construcción de los grids y la carga de información desde un catalogo.

Se diseñan distintos tipos de grids como por ejemplo:

- Grids de carga estática.
- Grids de carga dinámica.
- Grids editables.

También como agregarle compontes al grid como botones, barras de estado, pagina dores, limitar el número de registros por pagina, etc.

Así como las operaciones básicas invocadas desde el grid.

Se definió lo siguiente:

- Datos generales de la clase.
- Definición de atributos.
- Metadata del constructor.
- Creación de atributos PK.

Capítulo 4 Casos de uso

Carga de datos del usuario

Se realiza captura de información desde el lado del cliente, con el fin de probar y corroborar de que la función de captura de información este correcta.

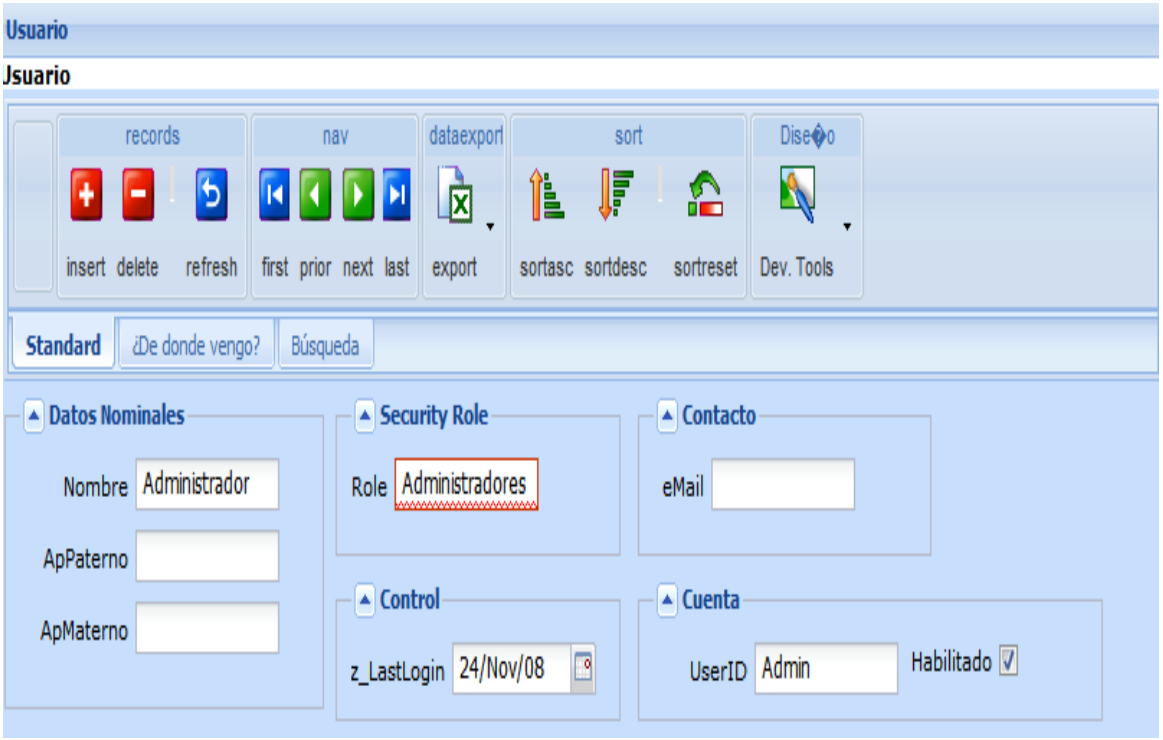

Figura 34.- Ejemplo de ficha de captura.

Eventos

Se procesan eventos que se pueden disparar de diferentes formas y en diferentes circunstancias, como por ejemplo:
Cuando se capturan datos en un grid:

- Función de guardado automático de los datos.
- Función de guardado mediante un botón.
- Función de validación del tipo de dato cuando se cambia de celda.
- Función de inserción.
- Etc.

Diseño de la IU (Interfaz de Usuario)

También se creó el mecanismo de diseño de la interfaz del usuario y no solo eso, sino que se creó un mecanismo donde el usuario puede acomodar los componentes de la manera que más le guste, se guarde esa configuración y se respete para cuando el usuario inicia su sesión dentro de la aplicación.

Los componentes también son drag and drop (arrastra y suelta)

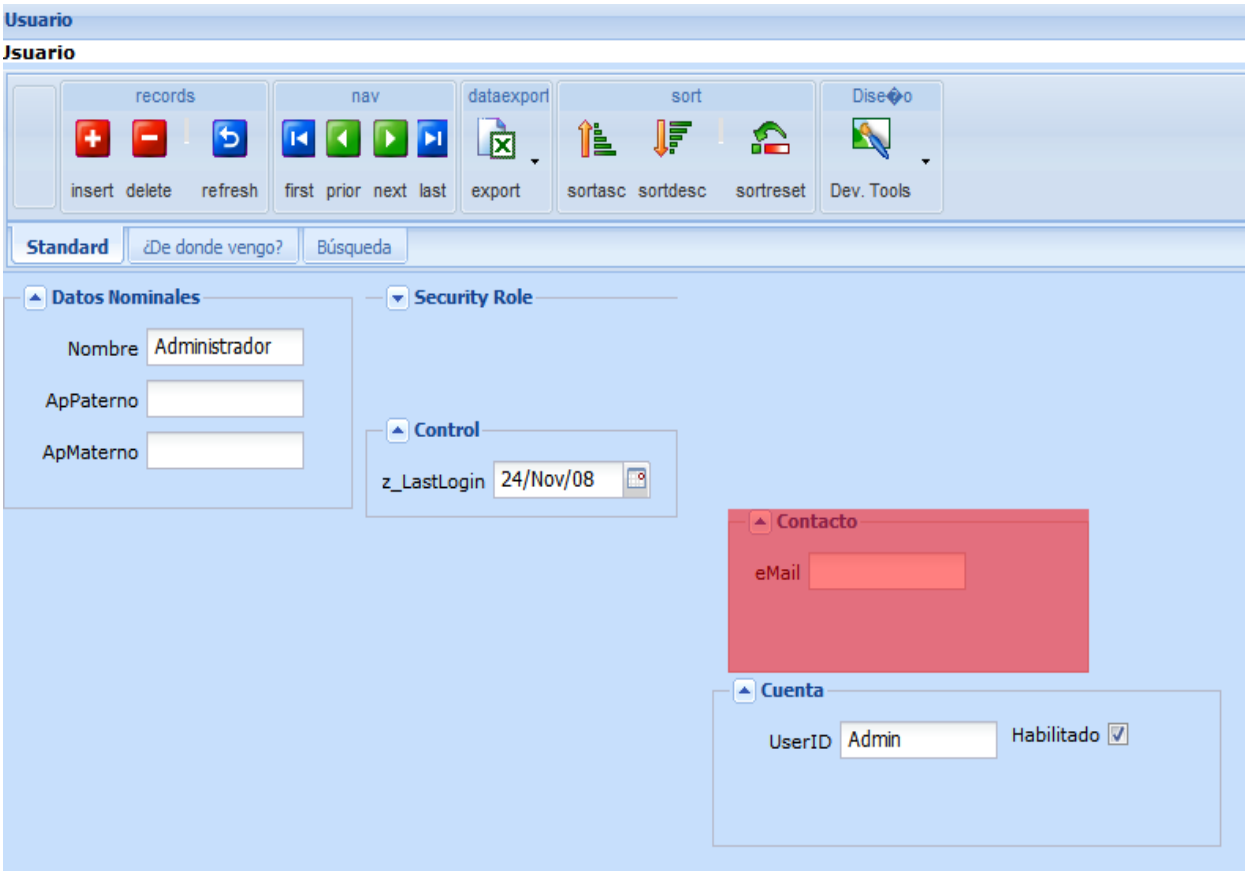

Figura 35.- Ejemplo de la interfaz de usuario.

En este ejemplo se muestra un componente de color rojo esto es porque se está moviendo de una posición a otra.

Creación del menú

Se elaboró el mecanismo de creación del menú, con la particularidad de que este puede ser modificado o cambiado mientras la aplicación esta en ejecución, mediante un archivo de configuración. Lo único que hay que hacer es actualizar la página de la aplicación.

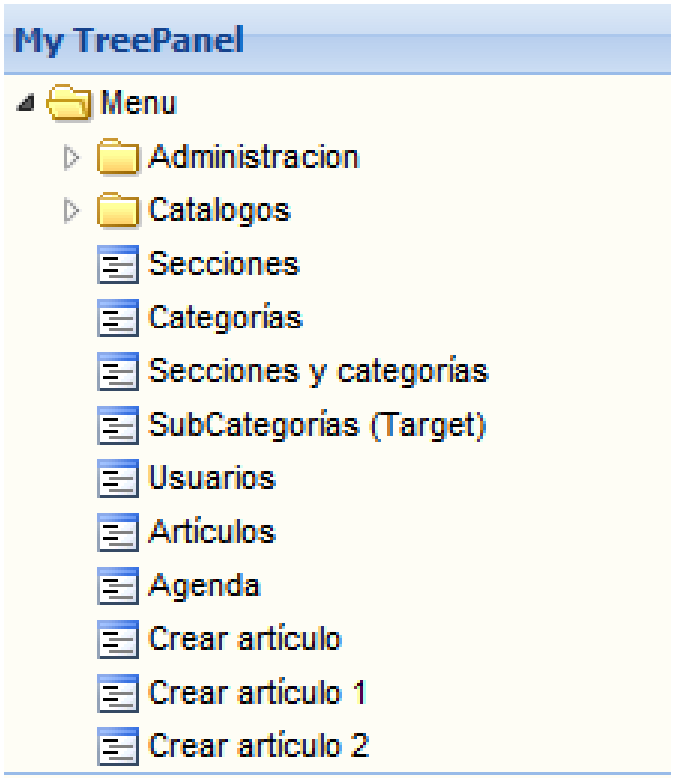

Figura 36.- Ejemplo de menú 2.

Maestro - Detalle

Se crearon ejemplos básicos de Maestro - Detalle y Maestro - Detalles

Que en esencia esto es dependiendo de un caso de uso padre o principal se harán operaciones especificas únicamente en una clase hija que afecte directamente a los **datos** que están presentes en la clase padre

Tanto en diseño por parte de cliente, como en la solución por parte del lado del servidor.

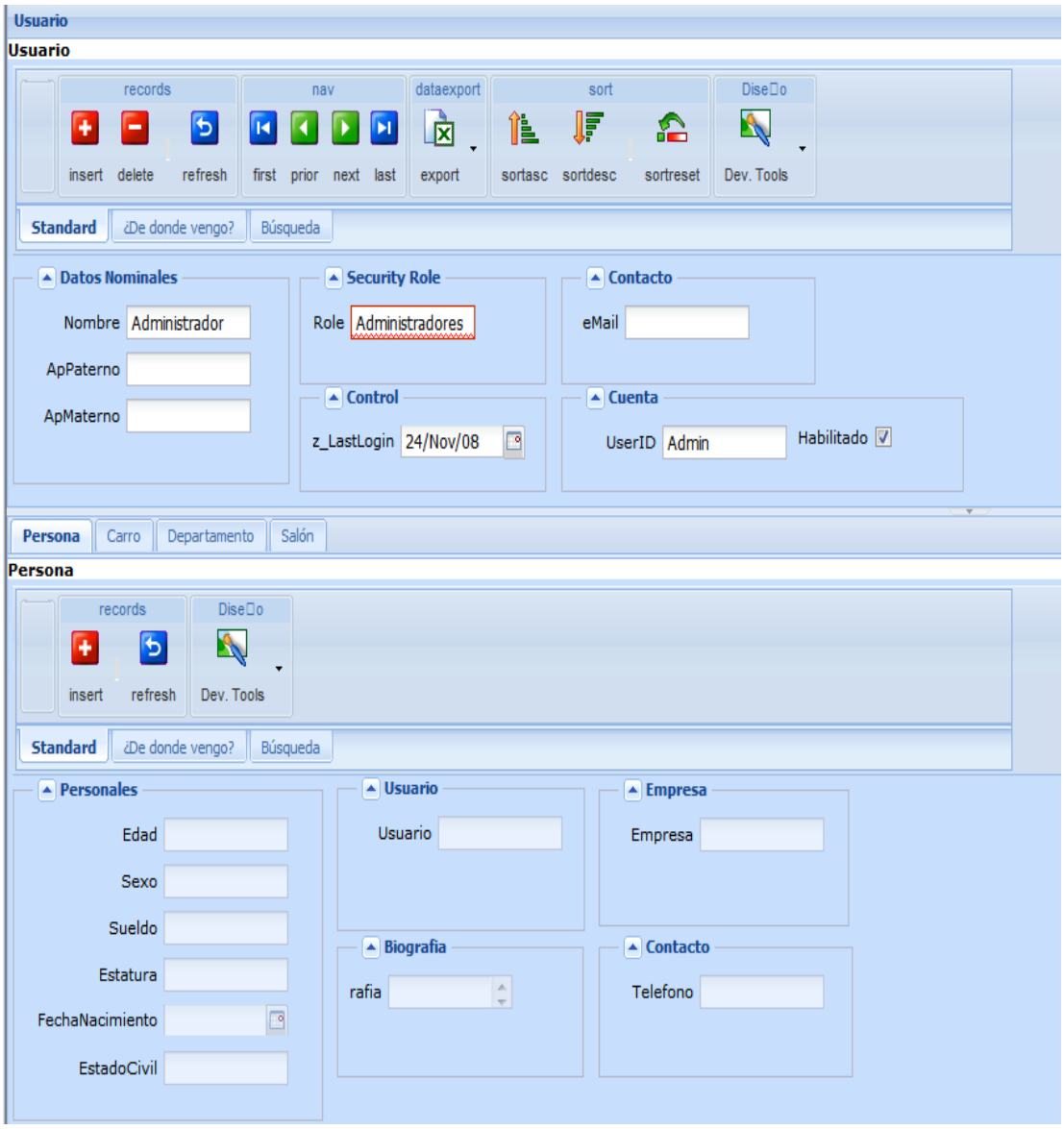

Figura 37.- Ejemplo de maestro - detalle.

En este ejemplo podemos observar que el caso de uso Usuario se captura un usuario y se le define el rol que va a tener en la aplicación, y en la parte de abajo en el caso de uso Persona se capturan datos específicos de este usuario en particular. Es decir si nosotros capturamos otro usuario diferente, se tendrá que capturar datos propios en el caso de uso Persona para este nuevo usuario, independiente mente de lo que le capturamos al usuario anterior.

Pestañas (Tab´s)

Es un elemento de la interfaz de un programa que permite cambiar rápidamente lo que se está viendo sin cambiar de ventana que se usa en un programa o menú.

Desempeñar una tarea a través de pestañas permite cargar varios elementos separados dentro de una misma ventana y así es posible alternar entre ellos con una mayor comodidad. Con las pestañas, además, es posible evitar tener multitud de ventanas abiertas en el escritorio, facilitando el uso del ordenador al usuario para trabajar en distintos aspectos de un mismo programa.

Se implementaron algunas pestañas del lado del cliente para solucionar operaciones específicas de cada caso de uso.

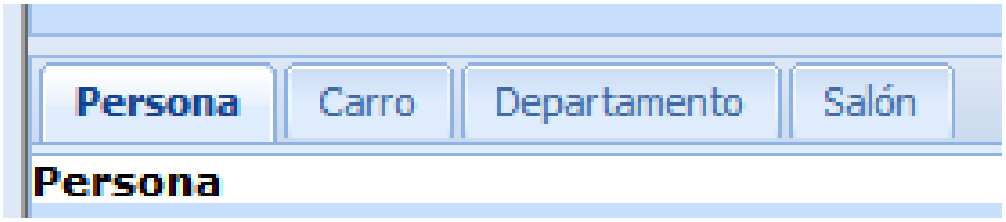

Figura 38.- Ejemplo de pestañas.

Registros (Log´s)

Se perfecciono el método de seguimiento o registro de acciones, para tener un rastreo de los errores.

Filtros

Que es una opción para encontrar los datos de un manera más rápida, sin tener que navegar entre toda la información de una manera manual.

Se creó un mecanismo de búsqueda de información mediante la aplicación de filtros de información.

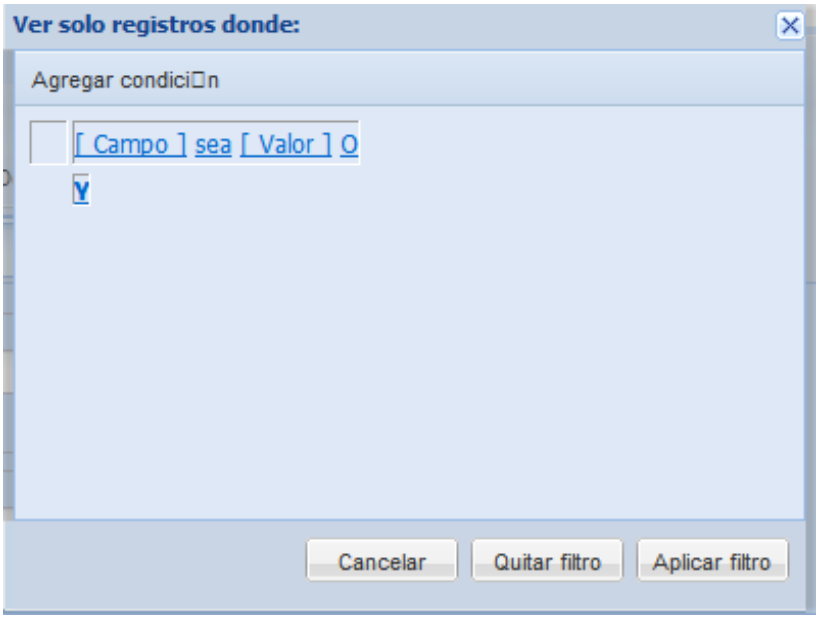

Figura 39.- Ejemplo de filtros.

Bitácora del trabajo realizado

- Investigación del ExtJS.
- Gate.
- Mecánica de templates.
- Pantalla principal.
- Login, validación de credenciales y asignación de directivas de seguridad.
- DAL.
- Reglas de negocio.
- Menú.
- Carga de ventanas.
- Ventana ficha.
- Ventana grid.
- Layouts de las ventanas.
- Tabs.
- Filtros.
- Logs.
- Caso de uso Maestro-Detalles.
- Manejo de Sesiones.

Bibliografía

http://www.mailxmail.com/curso-teoria-general-sistemas-informaticos/que-es-sistema

Cohen Karen, Daniel, *Sistemas de Información Gerencial, McGraw Hill.*

(BRIEN, 2006) O´ Briend, James. Bases de los Sistemas de Informacion, McGraw Hill.

http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx

Microsoft COM Technologies. url: http://www.microsoft.com/com/.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns : Elements of Reusable Object-Oriented Software.* Addison-Wesley Pub Co, 1st editi

http://es.kioskea.net/contents/bdd/bddintro.php3

http://technet.microsoft.com/es-es/

http://www.programacion.com/java/tutorial/servlets_jsp/1/

http://www.microsoft.com/spanish/msdn/latam/visualstudio2008/descripcion.aspx

http://support.microsoft.com/kb/550153/es

http://www.maestrosdelweb.com/editorial/comparacion-frameworks-javascript/

http://developer.yahoo.com/performance/rules.html#dns_lookups

http://es.wikipedia.org/wiki/No_te_repitas

http://leomicheloni.blogspot.com/2008/07/qu-es-la-inversin-de-control-inyeccin.html

http://msdn.microsoft.com/es-es/library/ms173172%28VS.80%29.aspx

http://msdn.microsoft.com/es-es/library/bb397687.aspx

Bill Evjen, Scott Hanselman Profesional ASP.NET 2.0 ANAYA multimedia

Conclusiones

Este proyecto se ha terminado hasta este punto y de manera general se han resuelto las especificaciones que se tenían planeadas.

Posteriormente se resolverán acciones específicas y la integración de este código con la herramienta con la que desarrolla la empresa; sin embargo el perfeccionamiento de todo este código elaborado y la integración de lo elaborado con la herramienta será parte de próximos proyectos.

Próximas acciones que se tienen que resolver para terminar la actualización e integración de este framework:

- Integración con BAX.
- Manejo de BLOBs.
- Servicios asíncronos de log y correo.
- Generación de reportes.
- Consultas no planeadas.
- Consola de administración

Anexos

Ajax

Siglas de Asynchronous Java Script and XML, es un término que describe un nuevo acercamiento a usar un conjunto de tecnologías existentes juntas, incluyendo las siguientes: HTML o XHTML, hojas de estilo (Cascading Style Sheets o css), Java Script, el DOM (Document Object Model), XML, XSLT, y el objeto XMLHttpRequest.

Cuando se combinan estas tecnologías en el modelo Ajax, las aplicaciones funcionan mucho más rápido, ya que las interfaces de usuario se pueden actualizar por partes sin tener que actualizar toda la página completa. Por ejemplo, al rellenar un formulario de una página web, con Ajax se puede actualizar la parte en la que se elige el país de residencia sin tener que actualizar todo el formulario o toda la página web completa.

ASP.net

El ASP.net de Microsoft es una tecnología de scripts que corren en el servidor y pueden ser utilizados para crear aplicaciones dinámicas e interactivas en el Web. Una página ASP.net es una página de HTML que contiene scripts que son procesados por un servidor Web antes de ser enviados al navegador del usuario. Usted puede combinar el ASP con el Lenguaje Extensible de Marcas (XML) y el Lenguaje de Marcas de Hipertexto (HTML) para crear poderosos sitios Web interactivos

En la actualidad una aplicación ASP.NET puede ejecutarse de dos formas distintas.

Aplicaciones cliente/servidor: estas aplicaciones están típicamente en formato de ejecutables compilados. Estos pueden integrar toda la riqueza de una interfaz de usuario, tal es el caso de las aplicaciones de desempeño y productividad, pero no se reúne la lógica de negocio como un recurso que se pueda reutilizar. Además acostumbran ser menos gestionables y escalables que las demás aplicaciones.

Aplicaciones que utilizan el navegador: Dichas aplicaciones están caracterizadas por contar con una interfaz de web rica y muy útil. La interfaz gráfica integra varias tecnologías, las cuales son el

HTML, XHTML, scripting, etc.; siempre y cuando el navegador que se esté utilizando soporte estas tecnologías.

Caso de uso

En otras palabras, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

Cliente – Servidor

La modalidad o arquitectura Cliente/Servidor es aquella en la que confluyen una serie de aplicaciones basadas en dos categorías que cumplen funciones diferentes (una requiere servicios y la otra los brinda) pero que a la vez, pueden realizar tanto actividades en forma conjunta como independientemente. Esas dos categorías son justamente cliente y servidor.

En el caso del cliente, es aquel que requiere un servicio del servidor. En esta categoría se realizan funciones de software basándose en el hardware pero en caso de no tener la capacidad de procesar los datos necesarios, recurre al servidor y espera a que este le brinde los servicios solicitados. El cliente es una estación de trabajo o computadora que está conectada a una red a través de la cual puede acceder al servidor.

Por el contrario, el servidor es la máquina desde la que se suministran servicios y que está a la espera del requerimiento del cliente. Una vez hecho, busca la información solicitada y le envía la respuesta al cliente; incluso puede enviar varios servicios a la vez, lo que es posible porque entre ellos están conectados mediante redes LAN o WAN.

Encriptación

Traducción de los datos originales de un mensaje a un código secreto, con el propósito de aumentar la seguridad de estos y evitar que sean vistos por personas no deseadas, sólo las autorizadas podrán, con una clave.

ExtJs

Es un framework de Java Script que encapsula muchas operaciones básicas en la programación, resolviendo estas de manera rápida y con un diseño vistoso.

Framework

Muchos de los que nos dedicamos al desarrollo de software utilizamos, conocemos o, como mínimo, nos hemos tropezado con el concepto de framework (cuya traducción aproximada sería "marco de trabajo").

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

¿Qué ventajas tiene utilizar un 'framework'?

Las que se derivan de utilizar un estándar; entre otras:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
- Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.

- Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.

Formas

Pantallas interactivas que permiten la presentación, introducción y respuesta de información, especialmente utilizada en entornos de ventanas y en páginas de internet.

Grid ó Grilla.

Cuadrícula para representar conjuntos de datos en forma de tabla.

GUI

(Graphical User Interface), interfaz gráfica de usuario. Programa software que gestiona la interacción con el usuario de manera gráfica.

Interfaz de usuario

Las interfaces básicas de usuario son aquellas que incluyen cosas como menús, ventanas, teclado, ratón y algunos otros sonidos que la computadora hace, en general, todos aquellos canales por los cuales se permite la comunicación entre el ser humano y la computadora. La mejor interacción humano-máquina a través de una adecuada interfaz (Interfaz de Usuario), que le brinde tanto comodidad, como eficiencia.

Java Script

Es un lenguaje basado en prototipos, pues las nuevas clases se generan clonando las clases base (prototipos) y extendiendo sus funcionalidades. Lenguaje de programación interpretado, o sea no requiere compilación. Es utilizado especialmente en páginas web embebido en el código HTML o similares. La mayoría de los navegadores pueden interpretar los códigos Java Script incluidos en las páginas web.

Log

Archivo que registra movimientos y actividades de un determinado programa (log file). En un servidor web, se encarga de guardar todos los requerimientos ("requests") y servicios entregados desde él, por lo que es la base del software de estadísticas de visitas.

Metadata

Se refiere a la descripción que se asigna a un objeto de aprendizaje, que sirve para catalogarlo y para solicitar su distribución

Microsoft SQL Server

Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL.

Navegador

Un navegador o navegador web (del inglés, *web browser*) es un programa que permite visualizar la información que contiene una página web (ya esté está alojada en un servidor dentro de la World Wide Web o en uno local).

El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

Programación orientada a objetos o POO (OOP según sus siglas en inglés)

Es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

Reglas del Negocio o Conjunto de Reglas de Negocio

Describe las políticas, normas, operaciones, definiciones y restricciones presentes en una organización y que son de vital importancia para alcanzar los objetivos misionales.

Sesión.

Espacio de tiempo comprendido entre el momento en que un usuario específico ingresa al sistema con su nombre de usuario y contraseña, y el momento en el que lo abandona.