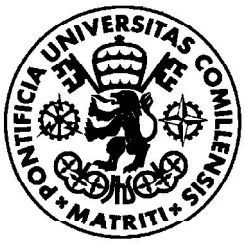


UNIVERSIDAD PONTIFICIA COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

ANÁLISIS DE LOS ESTÁNDARES HTML5 Y CSS3: IMPLEMENTACIÓN DE UN EJEMPLO PARA UN SITIO EN INTERNET

AUTOR: SARA COUCEIRO DE JUAN

MADRID, Septiembre de 2010

Autorizada la entrega del proyecto del alumno:

SARA COUCEIRO DE JUAN

El Director del Proyecto

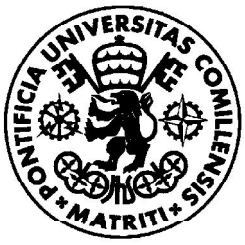
ANTONIO MACIAS VECINO

Fdo.: ..... Fecha: / /

Vº Bº del Coordinador de Proyectos

DAVID CONTRERAS BÁRCENA

Fdo.: ..... Fecha: / /



UNIVERSIDAD PONTIFICIA COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

ANÁLISIS DE LOS ESTÁNDARES HTML5 Y CSS3: IMPLEMENTACIÓN DE UN EJEMPLO PARA UN SITIO EN INTERNET

AUTOR: SARA COUCEIRO DE JUAN
DIRECTOR: ANTONIO MACIAS VECINO

MADRID, Septiembre de 2010

RESUMEN

Al ser Internet una red abierta, con el objetivo primordial de ser independiente de las plataformas que generación y acceso al contenido, se necesitan distintos estándares para poder intercambiar la información. Este proceso de estandarización lo realiza el consorcio W3C partiendo de las propuestas de las empresas implicadas. El procedimiento que han seguido hasta el momento ha sido basado en la práctica, primero, las empresas establecen una solución y cuando funciona se crea un estándar. La situación actual es que mientras se estandariza una tecnología es muy habitual que los distintos navegadores resuelvan el problema de distintas formas o mediante añadidos como Flash. Por compatibilidad con las aplicaciones preexistentes se acaban manteniendo las soluciones previas y por tanto aunque exista un estándar no siempre se utiliza. Por eso la velocidad de adopción de un estándar es muy importante.

El proyecto va a analizar distintas propuestas mejora a los estándares de Internet: HTML Versión 5 y CSS Versión 3. Las propuestas son elaboradas por grupos de trabajo distintos y se encuentran en distintas fases, por tanto los estándares no están cerrados y actualmente siguen evolucionando. Hay que tener en cuenta que un estándar es un ejercicio teórico pero la tecnología no será viable hasta que los motores de navegación comerciales los implementen de una manera eficiente.

A grandes rasgos las novedades facilitan la vida a los desarrolladores porque muchas funcionalidades, que se implementan con librerías complejas en java script o tecnologías de servidor como ASP.NET, PHP y JSP, pasan a poder realizarse en el navegador de una manera mucho más sencilla. No deja de ser una forma de unificar la metodología de desarrollo. Como consecuencia dedicando los mismos recursos se podrán conseguir aplicaciones Web más potentes y fáciles de utilizar.

Desde el punto de vista de los usuarios, estos percibirán que con las últimas versiones de los navegadores pueden tener a su disposición nuevas funcionalidades y la interfaz les es más agradable. Cada vez existen menos diferencias con las aplicaciones de escritorio y la tendencia es que los navegadores funcionen como pequeñas máquinas virtuales en las que se realiza gestión de procesos, tienen librerías gráficas, se guardan ficheros, se tienen bases de datos para las aplicaciones y se comunican entre ellas. Es decir que se van transformando en el entorno de ejecución universal para todo tipo de aplicaciones orientadas a ser utilizadas por un usuario.

Los estándares Web, HTML v5 y CSS v3, son tan amplios que no se han podido abarcar todas las novedades y se ha reducido a una muestra de las que actualmente se pueden utilizar. Es decir, existen implementaciones reales de las mismas y tienen utilidad para numerosas aplicaciones. En ese aspecto me he centrado en tratamiento de información en la parte del cliente (navegador) descargando al servidor de esa carga de trabajo.

ABSTRACT

As Internet is an open web, being its main aim to be independent of the generation platforms and content access, different standards are needed for exchanging information. This standardization process is done by the W3C consortium, starting by the recommendations of the implicated companies. They do as follows: based on their experience, the companies establish a solution and when it works, they create a standard. Presently, while a technology is standardized, it is very frequent that the different browsers resolve the problem in different ways or by additions, such as Flash. For compatibility with the existent applications, the previous solutions are maintained. Therefore, even though a standard may exist, it may not always be used. That is the reason why the speed with which a standard is adopted, is so important.

This thesis is going o analyse the different ways to improve Internet standards: HTML Version 5 y CSS Version 3. The proposals are elaborated by different working teams and are in different phases of development. Therefore, the Standard are not definite and presently they are in development. It has to be taken into account that a standard is a theoretical analysis but that the technology will not be viable until the commercial browsers will implement them in an efficient way.

Living a broad outline, the innovations make life easier for those developing. A great quantity of the functions, which are implemented with complex libraries in java script or Server technologies such as ASP.NET, PHP y JSP, become to be done by the browser in a much easier way. After all it is a way of unifying de development methodologies. Therefore, with the same resources much more potent and easy to use Web applications can be obtained.

From the users point of view, users will get a browser with different functions and a much more pleasant interface. Each time there are less differences between the desktop applications. The tendency is to make the browsers work as small virtual machines which manage the different processes. They have graphic libraries, they save files, they have data bases for the applications and they communicate between them. That is, they are transforming into a universal execution setting for all type of applications positioned for being used by users.

The Web, HTML v5 and CSS v3 standards, are so broad that all the innovations could not be covered in this thesis. A reduced sample of the innovations that actually can be used is described in this thesis. That is, there are real implementations for this innovations that are of great use for a great number of applications. In this aspect, I have focused on the treatment of information in the client's side (browser) reliving the server of that task.

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN

1.1. NECESIDAD DE LOS ESTÁNDARES EN INTERNET

Al ser Internet una red contenido abierto, independiente de la plataforma que genera y accede a ese contenido, se necesitan distintos estándares para poder funcionar. La evolución que han seguido hasta el momento ha sido basada en la práctica, primero se establece una solución y cuando funciona se crea un estándar.

HTML Versión 5 y CSS Versión 3 intentan aglutinar necesidades que existen en el mercado y que se están resolviendo de distintas maneras. Por ejemplo en el campo de los gráficos vectoriales Flash se ha convertido en un estándar de facto. En cualquier caso los estándares están en continua revisión, no obligan, y son prácticamente recomendaciones y soluciones de compromiso.

1.2. FINALIDAD DE LOS ESTÁNDARES HTML Y CSS

En sus comienzos el objetivo de HTML es mostrar y enlazar información. Posteriormente se introdujo la posibilidad de enviar información, mediante formularios, y modificar el contenido de la página de manera dinámica, mediante JavaScript.

La finalidad del estándar CSS es uniformar la presentación de varias páginas HTML de forma que un sitio Web presentara una apariencia consistente y fácil de modificar. Las CSS modifican el estilo de etiquetas y atributos HTML de manera que solo hay que definir la forma de presentar cada elemento en un único sitio.

La situación ideal, a la que tiende HTML 5, es separar totalmente la presentación (CSS) del contenido (HTML) de forma que se definan y trabajen

de manera independiente. Esto también permitiría separar perfiles profesionales y obtener una mayor calidad en el resultado final. La situación actual es que se está muy lejos de alcanzar esta separación y solo se está imponiendo en proyectos grandes, prohibición estricta del atributo *style* en HTML y de otros atributos de estilo.

1.3. IMPLEMENTACIÓN DE LOS ESTÁNDARES EN LOS NAVEGADORES

En el momento de escribir el proyecto existen varios motores de navegación diferenciados en los cuales se basan los distintos navegadores comerciales. Estos motores soportan con diferencias notables, y con errores, los distintos estándares referentes a la Web: HTML, CSS, JavaScript, XML ...

Actualmente el motor menos evolucionado es el Trident que da soporte a las diferentes versiones del Internet Explorer. De hecho la gama de Internet Explorer es la que menos funcionalidades referentes a HTML 5 y CSS 3 soporta. La futura versión 9, actualmente en Beta, se espera que mejore este aspecto. La política que parece seguir Microsoft es incorporar soporte a funcionalidades de HTML 5 cuando ya son habituales en otros navegadores (Opera, Firefox, Safari, Chrome ...).

Este soporte desigual de los estándares provoca que haya que validar las páginas Web en los distintos navegadores. Normalmente haciendo correcciones en el código y con un coste más elevado que desarrollar exclusivamente para un navegador. Es decir implementar los estándares de distintas maneras está costando dinero a la hora de desarrollar.

Por último hay que tener en cuenta que cada vez existen más dispositivos móviles con pantallas reducidas y acceso a Internet (iPhone, Blackberry,

móviles, netbooks…). En consecuencia los navegadores y los estándares están evolucionando para mostrar contenidos en pantallas reducidas. Hay navegadores como Safari, utilizado en el iPhone, que cada vez se utilizan más y por tanto cada vez más páginas Web los contemplan.

CAPÍTULO 2

ANÁLISIS DE HTML VERSIÓN 5

2.1. LA ESPECIFICACIÓN XHTML Y HTML 5

XHTML es una especificación que se populariza a partir del HTML versión 4.

Como ya se comentó en apartados anteriores el HTML que admiten los navegadores no tiene porque estar bien construido, con etiquetas cerradas y una sintaxis estricta. Uno de los grandes éxitos de Internet Explorer fue que empezó a mostrar páginas con errores en su programación pero que el usuario no percibía ninguna diferencia en cómo se le mostraban. Es decir los navegadores para ganarse a los programadores comenzaron a aceptar cualquier código y lo adaptaban para mostrar algo decente al usuario.

El XHTML implica ponerles restricciones al HTML y obliga entre otras cosas a:

- Cerrar todas las etiquetas o si es una única etiqueta indicar que se cierra.

  Ejemplos:

- Distinguir entre mayúsculas minúsculas en atributos y etiquetas.

- Todos los valores de los atributos de las etiquetas deben ir entrecomillados. ejemplo: id="valor_id"

- Tener un esquema XSD sobre el que se pueda validar el documento entero.

Todo esto hace que sea mucho más fácil y rápido tratar, transformar y validar los documentos XHTML, además de que sean documentos más correctos. En

realidad HTML como tal debería ser un lenguaje XML con unas etiquetas y atributos específicos. En la práctica eso no es posible y cualquier navegador comercial que quiera tener un mínimo de éxito debe permitir ciertas licencias a los programadores y mostrar código HTML erróneo.

HTML Versión 5 propone volver a los orígenes y respetar las normas generales que impone XML a los documentos HTML. Por tanto estos documentos serán más fáciles de tratar de manera automatizada. Lo cual viene muy bien para los robots indexadores de contenido, es uno de los parámetros que Google tiene en cuenta, y para el futuro desarrollo de la Web semántica, que se trata más adelante en este mismo documento.

2.2. ACCESIBILIDAD WEB Y NUEVOS ESTÁNDARES

La accesibilidad no es un concepto exclusivo de los navegadores, la idea general es que cualquier producto tenga en cuenta las necesidades especiales de todos los posibles usuarios finales. Como a nadie le gusta perder cuota de clientes lo extraño es porque ha tardado tanto en imponerse tanto en al ámbito público como en el privado. La razón hay que buscarla principalmente en el coste económico adicional que implica en el desarrollo de página Web y el desconocimiento de sus beneficios.

A grandes rasgos son necesarias varias consideraciones a la hora de hacer desarrollos Web accesibles:

- Personal más cualificado, especializado y con disciplina a lo hora de escribir el código. Separar la presentación (HTML) del negocio (.NET, J2EE, PHP ...).

- Pasar baterías de pruebas y modificar el código para solucionar errores.

- En algunos casos presentar alternativas a contenidos con Flash, JavaScript … es decir a veces se trabaja el doble.

Páginas para validar la accesibilidad de páginas Web. Estas páginas presentan un informe de los incumplimientos y las correcciones necesarias

- www.tawdis.net
- http://validator.w3.org/
- http://jigsaw.w3.org/css-validator/

Hay numerosa legislación española en este aspecto, es especialmente relevante el Real Decreto 1494/2007 de Acceso de los discapacitados a la sociedad de la información. Según el artículo 5 y la disposición transitoria única de ese decreto. Todas las páginas existentes o de nueva creación financiadas por la administración pública además de entidades o empresas que gestionen servicios públicos, deben que cumplir el nivel 2 (normativa WAI) a partir del 31 de Diciembre de 2008. Excepto cuando una funcionalidad no tiene alternativa tecnológica accesible (esto es importante por ejemplo en casos de firma electrónica donde no existen alternativas accesibles).

HTML5 y CSS3 pretenden desarrollar aspectos que inciden directamente en la accesibilidad, por ejemplo la representación sonora del texto o como se comentó en al apartado anterior documentos más estrictos y mejor formados. Estos nuevos estándares se están creando pensando las normativas de accesibilidad existentes, por tanto ayudan a etiquetar el contenido indicando que semántica tiene.

2.3. FILOSOFÍA GENERAL DE HTML 5

Lo primero es que el nuevo estándar surge a partir de un conjunto de especificaciones que resuelven problemas reales. La idea es que ya se tiene el problema resuelto, de alguna forma, y ahora de lo que se trata es de hacerlo bien y consensuar una solución.

HTML 5 tiene un horizonte muy lejano y todavía quedan años para estar completamente implementado. Mientras tanto sigue creciendo e incorporando especificaciones para facilitar el acceso a Internet a dispositivos móviles y decodificadores de televisores digitales.

Cada vez se pasa mayor funcionalidad a los navegadores, por tanto estos tienden a ser pequeños sistemas operativos. El estándar recoge estos cambios y comienza a incorporar más ampliaciones: Bases de datos en los navegadores, acceso al sistema de ficheros y dispositivos de E/S, reproductores de gráficos 2D / 3D, reproductores de video y audio, hilos de ejecución, mecanismos de comunicación ...

La eficiencia que proporcionan los navegadores a la hora de gestionar recursos no suele ser muy buena, se está mejorando. Sin embargo ese es un aspecto secundario frente al hecho de que puedas poner una aplicación a disposición de miles de clientes simplemente con subirla a producción en un servidor Web. Es una forma de gestionar la información muy atractiva e independiente de la plataforma, solo se ve limitada por la versión de navegador que tenga el cliente.

2.4. LA WEB SEMÁNTICA Y EL NUEVO ESTÁNDAR

La Web semántica se pretende que sea una evolución de la Web actual. La idea es no solo mostrar información sino permitir que las máquinas analicen esa información y nos muestren resultado o conclusiones respecto a ella. Sería como tener un buscador personalizado que aprende lo que te interesa y lo localiza analizando sitios Web. Para esto hay que etiquetar bien el contenido y sus relaciones mediante metadatos expresados en el lenguaje adecuado.

Las fases de evolución de la Web serían:

WEB 1.0 → Acceder a contenidos.
WEB 2.0 → Interactuar con el contenido y aportar información: redes sociales, Wikis y herramientas colaborativas.
WEB 3.0 → Las máquinas / navegadores extraen el contenido de la red y nos muestran lo que nos interesa. Son capaces de hacer deducciones y llegar a conclusiones.

Para favorecer la Web semántica cada vez más navegadores incorporan nuevos lenguajes para poder describir el contenido y asociarlos a páginas HTML 5 que lo visualizan. El HTML 5 al llevar una estructura similar al XML es más fácil de procesar. También organiza mejor la semántica del contenido y permite determinar cuál será la cabecera de la página o el pie de página entre otras cosas.

El lenguaje con más éxito es el RDF que permite establecer un conjunto de términos, sus relaciones y sus normas de combinación. Es válido en cualquier ámbito no solo en su aplicación para la Web y su implementación para la Web, basada en XML, es OWL y permite describir conocimiento en un ámbito

concreto. Los creadores de páginas Web deben indicar la semántica de los contenidos que muestran para que los procese un agente semántico. Esto todavía no se ha desarrollado, entre otras cosas, por la publicidad. Si se puede acceder al contenido de las páginas sin que las tenga que visualizar una persona se perderían ingresos publicitarios.

Cuando la Web semántica se expanda se podrán hacer preguntas genéricas, del estilo "comprar coche Madrid por menos de 2.000 euros", y los agentes semánticos / buscadores devolverán en lugar de un conjunto de páginas un conjunto de coches que se ajustan a esa petición.

2.5. CAMBIOS QUE INTRODUCE EL ESTÁNDAR HTML5

2.5.1. NUEVAS ETIQUETAS

HTML 5 proporciona nuevas etiquetas para indicar el tipo de contenido que se espera en cada zona de la pantalla de forma que los buscadores puedan indexarlo mejor.

2.5.1.1. Etiquetas nuevas para estructurar el contenido

- **hgroup:** Representa el índice o cabecera de una sección.

- **section** (Sección Independiente, se divide en artículos)**:** Representa una sección "general" dentro de un documento o aplicación, como un capítulo de un libro. Puede contener subsecciones y si lo acompañamos de h1-h6, etiquetas para destacar la importancia del texto.

- **article** (Contenido Independiente dentro de una sección)**:** Representa un contenido independiente en un documento, el caso más claro son las entradas de un blog o las noticias de un periódico online. Así, dentro de la portada podremos tener varios artículos demarcados semánticamente, por lo que una herramienta puede extraerlos fácilmente.

- **aside** (Menú de Navegación)**:** Se representa como una barra lateral para separar el contenido de navegación o auxiliar de una zona concreta. Esencial para delimitar el contenido "importante" del contenido "de apoyo"

- **header** (Encabezado): Representa la cabecera de una sección.

- **footer** (Pie de Página): Representa el pié de una sección, con información acerca de la página/sección, como el autor, el copyright o el año.

- **nav:** Representa una sección dedicada a la navegación entre el sitio.

La adaptación entre el HTML 4 y el HTML 5 se puede ver en el siguiente gráfico.

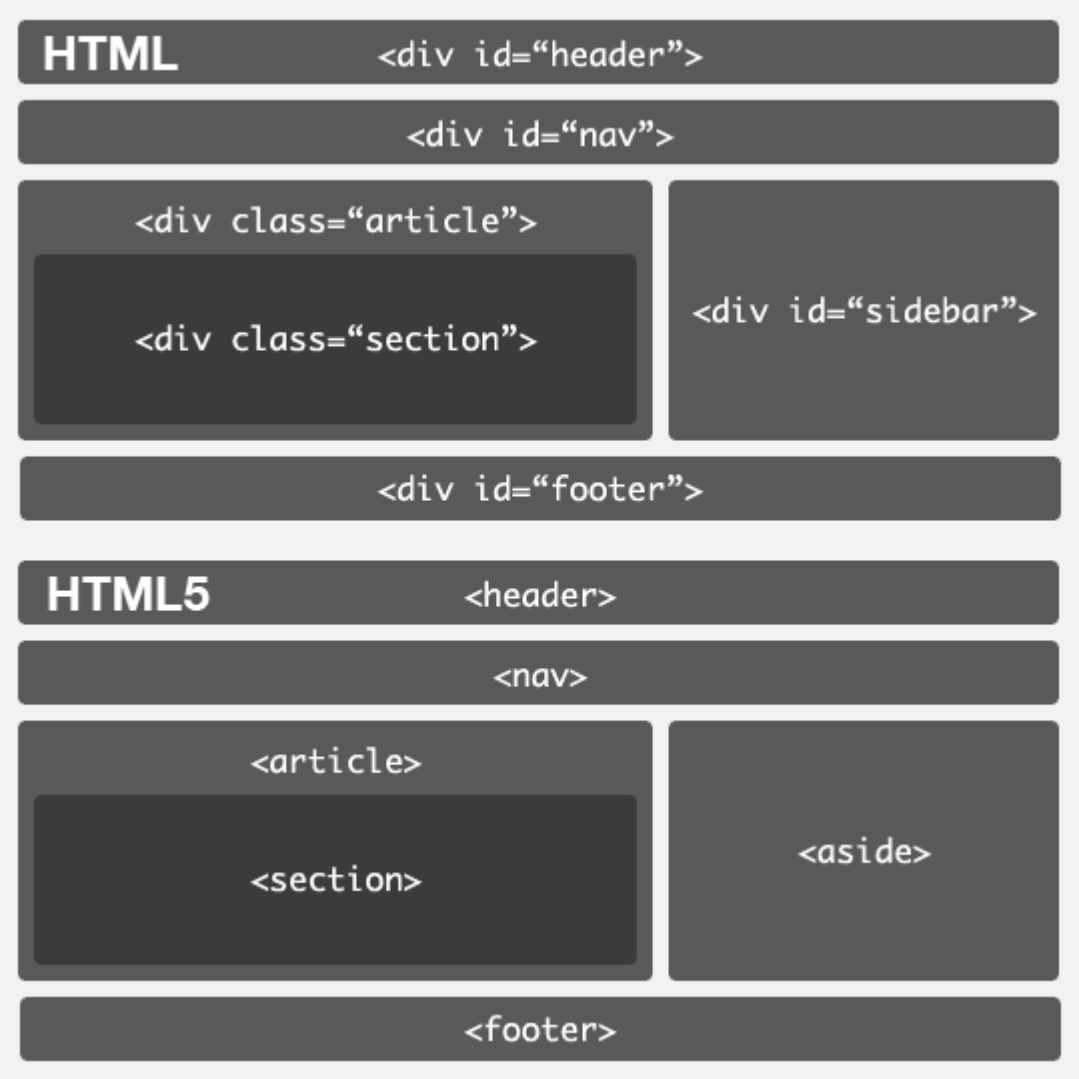


Figura 1 Adaptación del HTML 4 al HTML 5

2.5.1.2. Etiquetas Multimedia: Audio, Video

Se amplía el soporte a la reproducción de video y audio. Estableciendo unas etiquetas marco y unos codecs mínimos (h.264 para vídeo) que deben soportar los navegadores.

Sin duda uno de los añadidos más interesantes, ya que permite reproducir y manipular vídeos y audio sin necesidad de plugins como el de Flash. Se tratan de manera totalmente nativa como cualquier otro elemento, por ejemplo se pueden incluir enlaces o imágenes dentro de un vídeo. Aunque actualmente consumen muchos recursos se espera que se optimicen en un futuro.

Portales de vídeo como **Youtube** o **Dailymotion** ya están utilizando HTML5 para reproducir videos, eso sí en versiones Beta por el consumo de recursos y porque se necesitan los últimos navegadores para poder ver sus contenidos.

EJEMPLO → http://www.youtube.com/html5

(Hay que activar la beta de HTML5 y tener un navegador potente)

2.5.1.3. Etiqueta para generar gráficos: Canvas

Es una de las grandes novedades de HTML 5 y permite dibujar dentro de la pantalla todo tipo de gráficos. La API en JavaScript puede modificar los gráficos y además se da soporte a la aceleración de gráficos en 3D por hardware especializado. Permite desarrollar juegos y potentes aplicaciones gráficas con HTML 5 y JavaScript eliminando la necesidad de utilizar tecnologías Flash (SWF) o similares.

Esta etiqueta podría sustituir en un futuro a tecnologías existentes como Adobe Flash y la alternativa que estaba apareciendo es Microsoft Silverlight.

CAPÍTULO 3

ANÁLISIS DE CSS VERSIÓN 3

3.1. FILOSOFÍA GENERAL DE CSS 3

Las CSS ayudan a mantener un estilo uniforme en todas las páginas de un sitio web. Son recomendadas y prácticamente obligatorias para cumplir con la accesibilidad. Permiten independizar la presentación de los contenidos de manera que es más fácil modificar en un único sitio como se verán determinadas etiquetas en la página.

3.2. CAMBIOS QUE INTRODUCE EL ESTÁNDAR CSS 3

- **Seleccionar elementos a tratar, Selectores:** CSS 3 aporta nuevas formas de seleccionar etiquetas admitiendo expresiones y pudiendo recorrer la jerarquía de objetos, aparecen nuevos pseudo-elementos para ayudar a esto.

- **Nuevas unidades de medidas**

- **Expresiones y consultas:** Se permiten cálculos entre distintas unidades de manera que se pueden mezclar porcentajes y unidades absolutas o relativas. Facilita enormemente adaptarse a tamaños de pantalla variables.

- **Mejoras en el posicionamiento:** Actualmente el contenido se posiciona mediante tablas o mediante capas. Las tablas cada vez se desaconsejan más por la accesibilidad y porque no se adaptan bien cuando cambia el tamaño de la pantalla. Las capas realmente no están pensadas para posicionar y son bastante complejas de manejar. En todo esto hay que tener en cuenta que se pueden desarrollar hojas de estilo específicas dependiendo de la dimensión o tipo de dispositivo, por ejemplo diferenciando entre móviles y PCs normales.

Se proponen 3 formas para estructurar el contenido:

- ***Posicionamiento en columnas***: no está pensado para tener distintas filas sino que son simplemente columnas en una única fila.
- ***Posicionamiento en rejilla (Grid):*** Es una mejora al posicionamiento en columnas que permite definir filas y por tanto dar la misma funcionalidad que una tabla. Se adapta perfectamente al tamaño del dispositivo de salida.
- ***Posicionamiento con CSS Layouts (Pantillas):*** Este último método de posicionamiento es el más complejo y mucho más potente, se basa en hace un mapa de posiciones donde irá el contenido y luego ir referenciando a cada celda donde se quiere depositar el contenido.

  Utiliza los atributos display (definir el mapa de contenidos) y posición (referenciar las celdas del mapa).

- **CSS Mobile para dispositivos móviles, CSS TV**

- **Transiciones y animaciones en 2D y 3D**
  - ***Transiciones:*** Se refieren a cambios en propiedades en intervalos de tiempo. Pueden ocasionar movimientos, cambios de color … .
  - ***Transformaciones 2D:*** Pueden ser en 2D y en 3D, los elementos pueden ser rotados, trasladados o escalados. Se establece un sistema de coordenadas, partiendo de la posición del objeto.

- ***CSS para reproducción de voz:*** Permite describir las fuentes del sonido, las pausas, los tipos de voces, la entonación… y asociarlo al contenido de etiquetas para que cuando lo lean los navegadores lo hagan de forma correcta.

- **Nuevas propiedades para bordes y fondos:** Permite mediante CSS visualizar efectos que hoy en día se emulan con imágenes

CAPÍTULO 4

IMPLEMENTACIÓN ACTUAL DE LOS NUEVOS ESTÁNDARES

4. IMPLEMENTACIÓN ACTUAL DE LOS NUEVOS ESTÁNDARES

El soporte, a HTML 5 y CCS 3, depende principalmente del motor que utilice cada familia de navegadores. Por tanto primero hay que tener claro que motor está utilizando cada navegador comercial.

A continuación se muestra una tabla con estos datos:

MOTOR	VERSIÓN	REVISIÓN	EMPLEADO EN
Amaya	11.3.1	11.3-pre	Amaya
Gecko	1.9.2.3	1.9.3a4	Mozilla software, **Firefox,** SeaMonkey, Galeon, Camino; K-Meleon; Flock; Epiphany gecko
iCab			iCab 1-3
KHTML			**Konqueror**
Presto	2.5.24		**Opera;** Opera Mobile, Nintendo DS & DSi Browser; Internet Channel
Prince	7.0		Prince XML
Tasman			Internet Explorer para MacOS
Trident	4.0 (IE 8)	5.0 (IE 9)	**Internet Explorer**
WebKit	525	528	Apple **Safari; Google Chrome;** Shiira; iCab 4 OmniWeb 5.5+; Epiphany; Adobe AIR; Midori; Adobe Dreamweaver CS4; Google **Android** browser; Palm WebOS browser; Symbian S60 browser; OWB; Steam;

Tabla 1 Motores que utiliza cada navegador

CAPÍTULO 5

MOTIVACIONES Y JUSTIFICACIÓN DEL PROYECTO

5. MOTIVACIONES Y JUSTIFICACIÓN DEL PROYECTO

Tras realizar un análisis de los nuevos estándares de tecnologías Web propuestos por W3C se detecta que los estándares no están cerrados y que cada navegador los va implementado parcialmente. Cada motor de navegación según considera interesantes ciertas tecnologías las va incorporando. Por tanto es necesario realizar ejemplos reales para ver cual es realmente el estado del arte.

El proyecto de implementación se organiza como una serie de ejemplos independientes que utilizan algunas de las posibilidades de HTML5 y CSS3. La idea es intentar abarcar cuantas más tecnologías mejor. Tiene una fuerte componente de investigación porque son tecnologías muy experimentales y en la práctica no se sabe cómo van a responder. Las implementaciones de los estándares generalmente no son completas ni precisas en cada motor de navegación, existiendo diferencias en todos ellos.

De hecho el estándar de HTML5 promueve la desaparición de todas las etiquetas que tengan que ver con los estilos que se aplican a los datos, sin embargo ningún navegador las ha suprimido sino que se mantienen por compatibilidad. Es de esperar que el cambio se vaya realizando paulatinamente a través de los años y siempre que los navegadores ofrezcan facilidades a los desarrolladores.

CAPÍTULO 6

OBJETIVO Y ALCANCE DEL PROYECTO

6. OBJETIVO Y ALCANCE DEL PROYECTO

El objetivo final es mostrar mediante ejemplos algunas de las tecnologías más interesantes de HTML5 que están actualmente soportadas por los navegadores actuales.

En concreto me voy a centrar en tecnologías de almacenamiento y manejo de datos en la parte cliente, el navegador. Como navegador de referencia se va a tomar Google Chrome 4.1+.

Se propone una aplicación Web de Informes basada en arquitectura cliente - servidor con ASP.NET .La parte servidora proporcionará datos a la parte cliente, el cliente haciendo uso de tecnologías HTML5 + CSS3, mostrará y procesará esos datos en local (en el navegador) y devolverá algunos al servidor para que actualice sus tablas.

Los datos que se utilizarán son públicos y se refieren a financiación territorial y datos estadísticos. La idea es tener un conjunto de datos amplio y complejo para descargarlos a cliente, trabajar con ellos y que la carga de trabajo que supongan sea significativa.

El cambio es que estas tareas se solían realizar en la parte servidora y simplemente se manda al cliente el resultado final en forma de página HTML. En este proyecto se va a tratar de mostrar como pasar parte de la carga de trabajo al cliente y descargar al servidor de este trabajo. Actualmente existen tecnologías para hacer esto (Flash, Silverlight, Applets de Java …) pero ninguna es un estándar de los navegadores sino que son Plug-In que se deben instalar por separado.

CAPÍTULO 7

TECNOLOGÍAS

7.1. ARQUITECTURA TENCOLÓGICA DE LA APLICACIÓN

Como la finalidad es mostrar resultados y pasar carga de trabajo al cliente se montará una infraestructura básica en la parte servidora.

La plataforma tecnológica elegida es ASP.NET con el Framework 2.0. Se ha elegido por la facilidad de desarrollo rápido y sencillez en que se puede montar un servidor web virtual para desplegar la aplicación rápidamente y ejecutarla. Tiene una Base de Datos local montada con el motor SQL Server Express.

CLIENTE: Navegador Google Chrome 4.1 o Superior
SERVIDOR: ASP.NET 2.0, Desarrollando con el lenguaje C#
BASE DE DATOS: SQL Server 2005 Express Edition
HERRAMIENTA DE DESARROLLO: Visual Studio 2008 Express

Todo lo que se utiliza es gratuito para desarrollar aunque para poner la aplicación en producción tendría problemas de escalabilidad. Habría que desplegarla en un servidor Web Internet Information Server (Entorno Windows) o montar un servidor Apache con el Framework Mono (Entorno Linux).

En la parte cliente se recurre a Google Chrome 4.1 o superior como navegador de referencia porque al menos teóricamente es el que más funcionalidades incorpora y las herramientas de desarrollo que proporciona para depurar el HTML y el JavaScript son muy buenas. Otros navegadores como firefox (con Firebug y JavaScript Debugger) y Explorer (con IE Developer Toolbar) pueden llegar a proporcionar herramientas similares pero hay que instalarlas aparte.

Para la parte de desarrollo JavaScript, se recurrió al depurador de Google Chrome ya que el visual Studio da un soporte muy básico al JavaScript y además solo permite depurar correctamente en Internet Explorer.

7.2. DESCRIPCIÓN DE LA PLATAFORMA .NET

La plataforma .NET apareció como alternativa a Java en entornos Windows. Tiene características comunes con Java:

- Es compilado pero se genera un lenguaje intermedio que es interpretado por una máquina virtual, en este caso el Common Language Runtime. Al final se genera un código intermedio que interpreta esta máquina virtual, el intérprete de código nativo JIT (Just-In-Time). El código que manejan se llama administrado y controla desbordamientos de memoria, permisos y consumo de memoria. De esta manera que se controlan numerosos errores en ejecución y se libera al programador de tener en cuenta tareas rutinarias como liberar la memoria.

Diagrama del CLR

Figura 2 Diagrama CLR

- .NET Framework es gratuito, mientras no cambie la política de Microsoft, y pone a disposición de los desarrolladores miles de clases bien descritas y fáciles de reutilizar. Especialmente útiles para trabajar en entornos Windows.

Y diferencias significativas:

- El desarrollo de interfaces es mucho más rápido, el entorno de desarrollo (Visual Studio) es muy potente y se integra muy bien con todas las herramientas que proporciona Microsoft. Base de datos SQL Server y servidor Web IIS. Al ser en la práctica un monopolio de

Microsoft está todo muy unificado y no hay la dispersión de Frameworks y herramientas que hay en el entorno Java.

- Aunque existe la plataforma Mono para sistemas Linux, lo cierto que esta tecnología se orienta para trabajar en entorno Windows y no tiene ningún uso comercial fuera de estos entornos.

- El IIS es menos escalable que otros servidores de aplicaciones como Weblogic o Websphere pero también es verdad que cuesta una décima parte que estos al venir incluido con la licencia del sistema operativo.

- A diferencia de Java, un único lenguaje, permite el uso de múltiples lenguajes con sintaxis y orientaciones distintas. Algunos ejemplos son C#, Visual Basic.NET, COBOL.NET, Delphi ..., hay implementaciones de varias decenas de lenguajes.

Por último mencionar que, al igual que Java, no es eficiente para aplicaciones que consuman muchos recursos. Tampoco es la mejor alternativa para programar drivers, acceder al hardware o realizar procesos batch.

En el siguiente diagrama se ven los componentes que forman un entorno .NET.

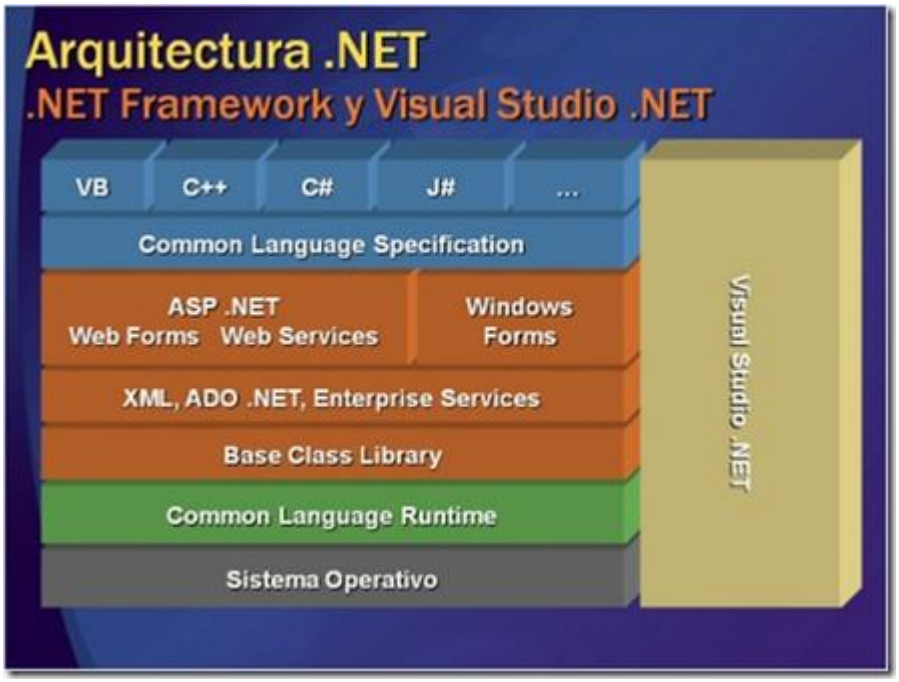


Figura 3 Componentes de un entorno .NET

La aplicación se ha desarrollado en ASP.NET 2.0 con Web Forms que siguen una arquitectura de objetos totalmente diferenciada de los Windows Forms, los cuales son para aplicaciones de escritorio.

Los Web Forms actúan generando un conjunto de clases que dan soporte a los eventos y el estado de una página HTML. De manera que modificando un conjunto de objetos en el servidor, el resultado que se le devolverá al cliente será un HTML y un JavaScript que los representa. El estado entre sucesivas peticiones se mantiene mediante una variable llamada VIEWSTATE que se envía y se recibe continuamente. La ventaja de este sistema es que libera al programador de mantener el estado de los controles (listas, campos de texto, selecciones ...) entre llamadas al servidor.

Asociado a cada página (por ejemplo WebForm.aspx) se le asocia y genera un fichero de código (WebForm.aspx.cs). En este fichero se va a escribir todo el código de servidor y se realiza el tratamiento de los eventos. Esta forma de separar la presentación de la lógica de negocio se llama CodeBehind. Cuando se compila el proyecto generará una DLL de código gestionado conteniendo todas las clases asociadas a las páginas del proyecto y a las clases de negocio.

El siguiente diagrama muestra como se organizan las clases .NET asociadas a una página.

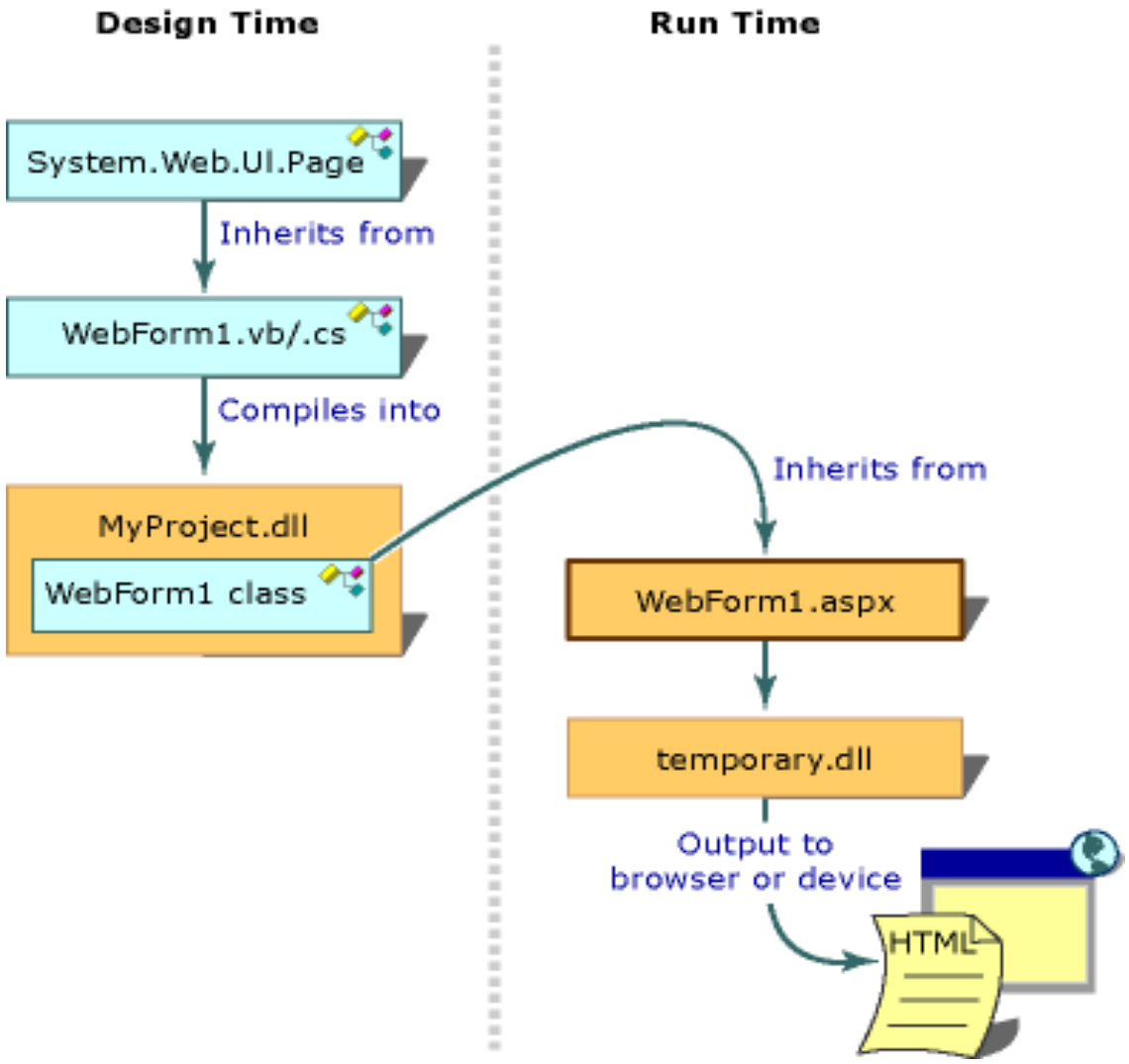


Figura 4 **Organización de clases .NET**

7.2. TECNOLOGÍA DE ACCESO A LA BASE DE DATOS

El acceso a datos que se realiza en la aplicación utiliza los Typed Dataset (Dataset Tipados), los cuales se integran muy bien con SQL Server. Es una tecnología de mapeo Objeto-Relacional que permite el acceso a las tablas de la base de datos a través de clases que se generan mediante asistentes del Visual Studio 2008. Las clases mapean los tipos de datos de las tablas a objetos que pueda manejar el .NET Framework y se encargan de añadir métodos para realizar las actualizaciones, borrados e inserciones de datos. Incluso hay controles de ASP.NET que permiten utilizar estos objetos como orígenes de datos para otros controles de una página.

Esta tecnología permite estructurar el acceso a la base de datos de manera que se cometen menos errores en tiempo de ejecución y el programador trabaja con la base de datos relacional como si fueran objetos.

La cadena de conexión a la base de datos se encuentra en el fichero Web.Config de manera que cambiando una línea de ese fichero se puede modificar la base de datos a la que referencia toda la aplicación. Esto es especialmente útil cuando se tienen entornos diferenciados de desarrollo, pruebas y producción. Lo habitual en el desarrollo de aplicaciones ASP.NET es indicar en el Web.Config todos aquellos parámetros que cambien de un entorno a otro, por ejemplo las rutas a carpetas, las cuentas de correo o determinados usuarios para ejecutar procesos.

Por último señalar que acceder a la Base de datos controlando el tipo de datos que se mandan y el tipo de datos que se muestran es importante por el aspecto de la seguridad. Evitando ataques por SQL Injection y estableciendo buenas

prácticas a la hora de desarrollar por capas. Se separa el accedo a datos de la presentación.

CAPÍTULO 8

IMPLEMENTACIÓN DE UN SITIO WEB CON HTML 5 Y CSS 3

8.1. ANÁLISIS

8.1.1. REQUISITOS HARDWARE Y SOFTWARE DE BAS

Para desarrollar y alojar el proyecto valen las siguientes plataformas Windows en todas sus versiones:

- Windows XP
- Windows 2000, 2003, 2008
- Windows Vista
- Windows 7

Solo es necesario que den soporte, y permitan instalar, un IIS superior a la versión 5.0 que pueda desplegar aplicaciones ASP.NET utilizando el .NET Framework 2.0.

Hardware Mínimo Necesario

Sobre todo viene limitado por el sistema operativo, se toma como referencia Windows XP que el que menos recursos consume. Los requisitos del IIS y del SQL Server Express no son muy elevados y para este proyecto no son relevantes.

- 512 MB de Memoria
- 10 MB de Disco Duro para desplegar el IIS y generar archivos temporales de .NET
- 5 MB de Disco Duro para la Base De Datos
- Pentium III a 1Ghz

Como se puede ver cualquier equipo de gama baja cumple estos requisitos. Para desarrollar con el visual Studio Express 2008 se necesitan bastantes más recursos.
La configuración mínima recomienda es la siguiente:

- 1GB de Memoria
- 1GB de disco para el entorno de desarrollo y la ayuda
- Pentium 4 a 1,5 GHz

8.1.2. REQUISITOS DEL SITIO WEB

Se va a desarrollar un sitio web para alojar los distintos ejemplos y proporcionarles una infraestructura básica. Los requisitos que se detallan a continuación no son específicos de ningún ejemplo en particular sino que sirven para el conjunto de ejemplos.

Se dividen en Requisitos Funcionales (RF) y Requisitos No Funcionales (RNF).

- ***Requisitos Funcionales***

RF1. Se necesita un servidor, lo más sencilla posible, para servir tablas de Base de Datos, en formato texto separado por '|'. Recibe como parámetro el nombre de la tabla y devuelve ficheros de texto al cliente.

RF2. Se creará un portal desde el cual se podrán acceder a todas las funcionalidades del sitio. Es decir las páginas de los ejemplos funcionarán de manera independiente en la medida que sea posible.

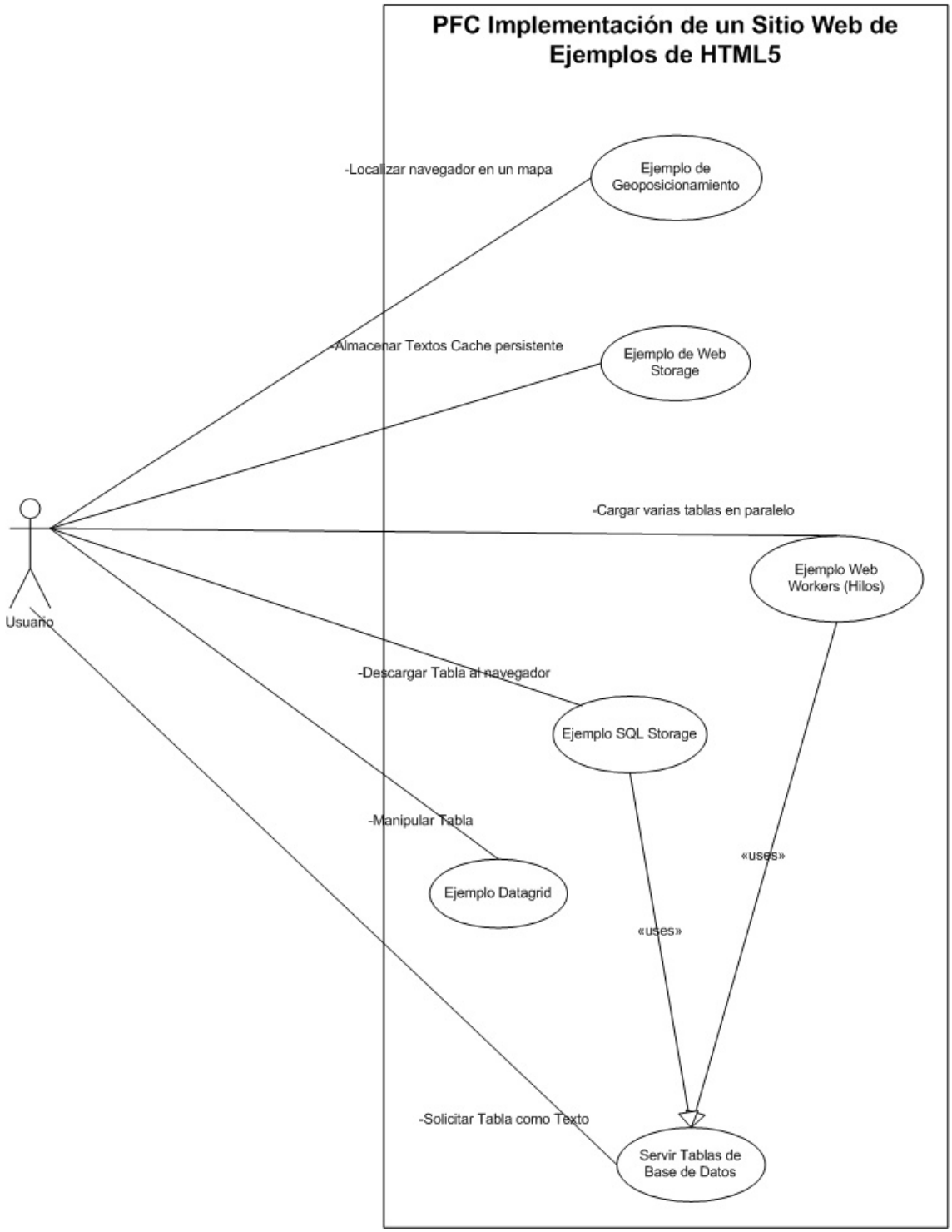
- ***Requisitos No Funcionales***

RNF1. Al tratarse de pruebas, el tiempo de respuesta no debe ser superior a 30 segundos para agilizar su visualización.

RNF2. Se toma como referencia el navegador Google Chrome versión 4.1 o superior. Los ejemplos se desarrollan optimizados para ese cliente.

8.1.3. CASOS DE USO

CASOS DE USO



Detalle de los casos de uso

Al ser un sitio web pensado para mostrar mediante ejemplos las mejoras que incorporan HTML5 y CSS3 no existe control de la seguridad ni diferentes perfiles de acceso al sistema. Por tanto solo existe un único actor que va accediendo a los distintos ejemplos que funcionan de manera independiente. El único caso que requiere conexión a Internet e interactúa con servicios externos es el ejemplo de Geoposicionamiento.

A continuación se detalla el caso de uso que proporciona tablas de la base de datos como texto. Este caso sirve a varios ejemplos y se puede considerar común, el resto de los casos se verán en la documentación específica de cada ejemplo:

Nombre	Servir Tablas de la Base de datos
Actor Principal	Usuario
Descripción	- Accede a la base de datos. - Recupera todas las filas de una tabla, pasada como parámetro. - Sirve los datos como texto separando cada fila por un salto de línea y cada columna por '|'.
Extensiones	- Se produce un error si el servidor no puede

	acceder a la base de datos
Datos	Tablas de la base de datos

El acceso a la base de datos no se hace directamente sino que se utiliza una clase DAO (Data Access Object) que se mapea los tipos de datos que se mandan y reciben, hace validaciones y oculta a la aplicación el motor de base de datos utilizado. A la hora de mantener aplicaciones es muy interesante tener localizado todo el código que permite el acceso a la base de datos. Las consultas SQL quedan ocultas y se mediante métodos de un objeto .NET.

Ejemplo de un acceso a Base de Datos

1. Se crea la instancia del objeto DAO

   AppCode.DatosTipadosBBDDTableAdapters.DiputacionesTableAdapter adaptador

   =new

   pfc_html5.AppCode.DatosTipadosBBDDTableAdapters.DiputacionesTableAdapter();

2. Tabla tipada donde devuelve los datos

   AppCode.DatosTipadosBBDD.DiputacionesDataTable tabla =new pfc_html5.AppCode.DatosTipadosBBDD.DiputacionesDataTable();

3. Finalmente cargo la tabla, realiza la consulta a la BBDD que ha especificado en el dataSet Tipado para obtener todas las filas de la tabla de Diputaciones.
   adaptador.Fill(tabla);

AJAX

En la parte cliente se realizan varias peticiones a la página ServirTabla.aspx.

Estas llamadas son mediante AJAX. A continuación indico el esqueleto básico de código necesario para realizar dichas llamadas.

```
// creo el objeto para realizar las llamadas
        var xmlhttp = new XMLHttpRequest();

// Abro una conexión y solicito datos
        xmlhttp.open("GET", urlDatos,true);

// Evento para capturar los datos devueltos de manera asíncrona
        xmlhttp.onreadystatechange=function()  {

// El estado 4 es que ha obtenido los datos correctamente
        if (xmlhttp.readyState==4) {
        var textoResultado = xmlhttp.responseText
          ……
        } // END IF
} // END FUNCTION
```

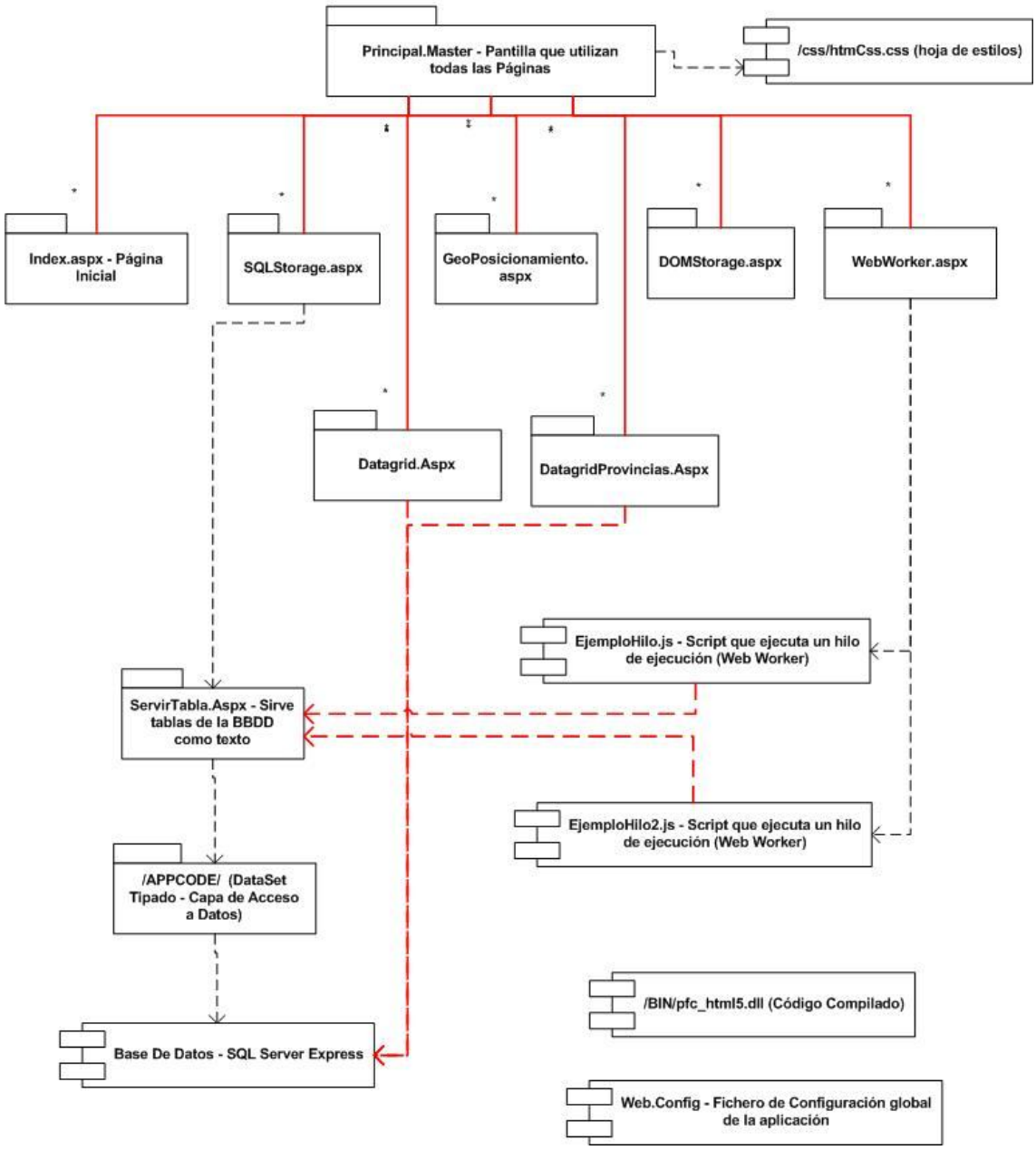
8.2. DISEÑO

8.2.1. DIAGRAMA DE PAQUETES Y COMPONENTES

Se describen los distintos módulos que componen la aplicación y se interacción. Se consideran las páginas (ASPX) como si fueran clases puesto que por detrás generan un modelo de objetos que representan cada uno de los elementos de la página. Modificando estos objetos se cambia el código HTML que se le envía al cliente.

Diagrama de Paquetes y Componentes

Se Indican las páginas web como paquetes, .NET crea un modelo de objetos para darles soporte.



A continuación pasa a describirse cada uno de los componentes:

- *Principal.Master* es la página que define la cabecera, pie de página, menú izquierdo y estructura de todas las pantallas del sitio. El resto

de páginas heredan de esta plantilla. A su vez incluye la hoja de estilos de la aplicación.

- La librería *pfc_html5.dll* incluye el código compilado de todas las clases que generan las páginas de la aplicación (ASPX). De manera que el código del servidor acaba recogido en una única librería DLL que se puede desplegar de manera independiente.

- *Web.Config* es un fichero XML donde se define la conexión a la base de datos y se declara, entre otras cosas, cual va a ser la página de error, el idioma o la codificación por defecto de todas las páginas.

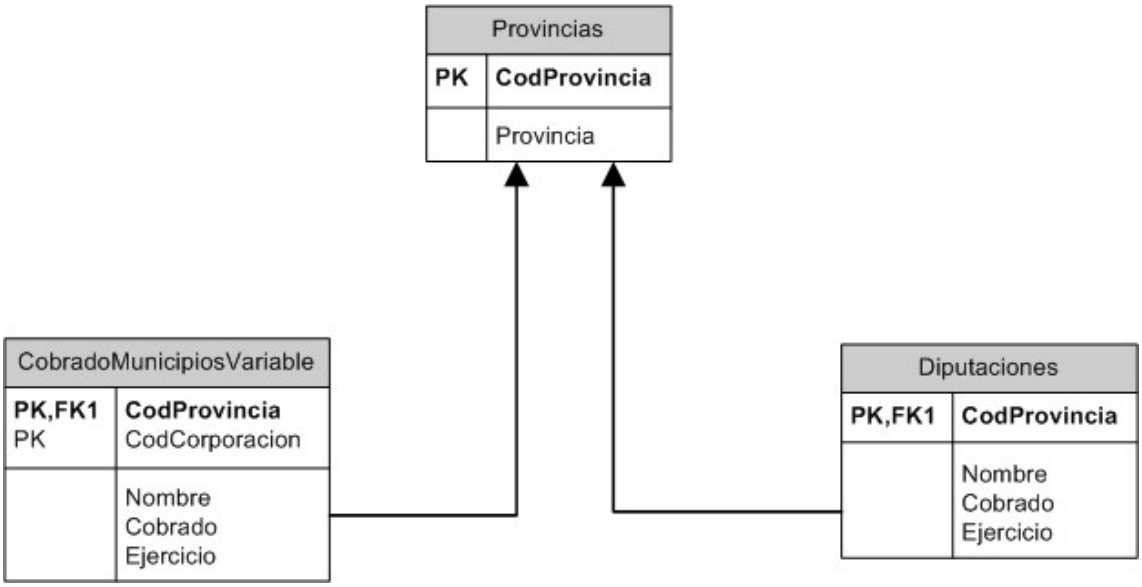
- En la carpeta *AppCode* se crea el Dataset Tipado, el modelo de objetos que acceden a la base de datos mapeando los datos de la base de datos a datos del .NET Framework.

- Cada ejemplo se ejecuta en una página distinta: (GeoPosicionamiento.aspx, SQLStorage.aspx, DOMStorage.aspx y WebWorker.aspx).

- *ServirTabla.aspx* es una página que no genera código HTML sino que accede a una tabla que se le indica como parámetro, recupera todas sus filas y las sirve como texto para que la puedan utilizar otros ejemplos.

8.2.2. MODELO DE DATOS

Como el objetivo de la aplicación es centrarse en tecnologías de cliente no se ha hecho un modelo de datos excesivamente complejo. Se pretende tener un conjunto de dato suficientemente amplio para poder manipularlos. En este caso se han cogido datos públicos de financiación en ayuntamiento y diputaciones en el año 2009. La carga de datos se realizó a partir de unos Excel descargados del portal del Ministerio de Economía y Hacienda.



Como se puede observar las tablas de pagos a ayuntamientos y diputaciones se relacionan con las provincias a través del código de provincia. Es un modelo sencillo pero que permite realizar cálculos numéricos con los datos y hacer JOINs (Uniones) entre tablas cuando se descargan a la Base de datos local del Navegador.

Los tipos de datos de los campos de las tablas son los siguientes:

Tabla de Provincias	
CodProvincia	Nchar(10) Clave Primaria
Provincia	Nchar(100) Texto en Unicode Nombre de la provincia
La tabla guarda los nombres de las provincias, la utilizan las otras 2 tablas restantes para poder mostrar el nombre de la provincia.	

Tabla de CobradoMunicipiosVariable	
CodProvincia	Nchar(10) Clave Primaria Clave Ajena

CodCorporacion	Nchar(10) Clave Primaria
NombreCorporacion	Nchar(100) Texto en Unicode Nombre del Municipio
Cobrado	Numeric(18,2) Importe en euros con 2 decimales
Ejercicio	Nchar(10)
Cobrado por Municipios Españoles en el ejercicio 2009 por parte del estado. Solo están en la tabla los municipios menores de 75.000 habitantes y por tanto los municipios más importantes como Madrid y Barcelona no aparecen. Estos municipios cobran por otro mecanismo. Tiene algo más de 8000 filas y es útil para hacer pruebas de carga con un conjunto de datos amplio.	

Tabla de Diputaciones	
CodProvincia	Nchar(10) Clave Ajena
Nombre	Nchar(100) Texto en Unicode Nombre de la provincia
Cobrado	Numeric(18,2) Importe en euros con 2 decimales
Ejercicio	Nchar(10)
Cobrado por las diputaciones en el ejercicio 2009, es el caso de las islas se pueden tener varias diputaciones por provincia.	

8.3. IMPLEMENTACIÓN

8.3.1. WEB STORAGE, NOTIFICACIONES Y DRAG & DROP

8.3.1.1. Estándares W3C a los que se refiere

WEB STORAGE

Es un mecanismo de almacenamiento que promueve la W3C. Está implantado en la mayoría de los últimos navegadores. Menos la parte del Global Storage que tiene menos soporte y funciona de manera ligeramente distinta. En el Global Storage es necesario indicar a que domino Web afectan los datos guardados.

Se define una interfaz de funciones y atributos que vale para los 3 sistemas de almacenamiento.

```
interface Storage {
  readonly attribute unsigned long length;
  getter DOMString key(in unsigned long index);
  getter any getItem(in DOMString key);
  setter creator void setItem(in DOMString key, in any data);
  deleter void removeItem(in DOMString key);
  void clear();
};
```

Los tres sistemas de almacenamiento guardan elementos como en forma de Clave y su contenido asociado. La diferencia es la persistencia de los datos y el ámbito de páginas al que se dirigen. El almacenamiento de los

datos persistentes no lo especifica el estándar, cada navegador mantendrá su propia estructura de ficheros.

- **Session Storage** → No persistente, vale para un sitio web mientras no se cierre el navegador.

- **Local Storage** → Persistente. Los datos duran un periodo de tiempo (días) después de cerrar el navegador y los almacena en ficheros.

- **Global Storage** → Persistente. Los datos los pueden usar todas las páginas de un dominio web (por ejemplo www.prueba.es) aunque estén alojadas en sitios distintos y sean aplicaciones diferenciadas.

Pueden generar eventos, que se lanzan cuando se modifica un elemento. Son similares a los triggers de base de datos y tienen la siguiente interfaz:

```
interface StorageEvent : Event {
  readonly attribute DOMString key;
  readonly attribute any oldValue;
  readonly attribute any newValue;
  readonly attribute DOMString url;
  readonly attribute Storage storageArea;
  void initStorageEvent(in DOMString typeArg, in boolean
canBubbleArg, in boolean cancelableArg, in DOMString keyArg, in any
oldValueArg, in any newValueArg, in DOMString urlArg, in Storage
storageAreaArg)};
```

WEB NOTIFICATIONS

Permiten mostrar ventanas emergentes con mensajes. Los mensajes son sencillos y pueden llevar un título, una imagen y un texto. Es una forma de tener una interfaz más atractiva al estilo de los mensajes emergentes del Messenger. Junto con AJAX se pueden implementar aplicaciones de correo y de chat muy vistosas.

```
interface Notification : EventTarget {

 void          show();

 void          cancel();

        attribute Function   onclick;

        attribute Function   ondisplay;

        attribute Function   onerror;

        attribute Function   onclose;

        attribute DOMString  replaceId;

        attribute DOMString  dir;

};
```

Se pueden controlar, mediante eventos, las acciones que realizan los usuarios sobre los mensajes. Además los mensajes se muestran en orden de creación. En el chrome, solo se muestran 3 mensajes a la vez, siendo necesario que cerrar algún mensaje para mostrar el siguiente.

DRAG & DROP

Esta API en JavaScript facilita mucho el desarrollo de aplicaciones con funcionalidades de arrastrar y soltar. Todos los objetos de HTML 5 la soportan lo único que hay que hacer es marcar los objetos que deseemos manejar con el atributo 'draggable', capturar los eventos y manejar el objeto 'dataTransfer' que donde se la información que se va mueve al arrastrar. Actualmente existen librerías en javascript que emulan esta funcionalidad pero de una manera más reducida, principalmente solo para capas, y son mucho más lentas y difíciles de implementar.

Los nuevos eventos que soportan los objetos son:

- dragstart
- dragenter
- dragleave
- dragover
- drag
- drop

Y los efectos que soportan (atributo effectAlllowed), la idea es que no solo se puede copiar información sino también enlazar o mover el objeto.

- none
- copy, copyLink, copyMove, all
- link, linkMove
- move

En el ejemplo se implementan 2 eventos:

Arrastrar → Donde se guarda el ID del elemento que se arrastra.

Soltar → Donde se recupera el contenido de un elemento por su ID.

Los eventos es necesario definirlos para cada objeto que se pueda manipular, es decir si existen 10 objetos que se pueden arrastrar se deben asociar 10 eventos, uno por cada objeto.

W3C – Grupos de Trabajo

Drag & Drop
http://www.w3.org/TR/html5/dnd.html

Web Notifications
http://dev.w3.org/2006/webapi/WebNotifications/publish/

Web Storage
http://dev.w3.org/html5/webstorage/

8.3.1.2. Requisitos

Realizará un ejemplo de almacenamiento y manipulación de datos en el Local Storage y Session Storage del cliente, una versión evolucionada de las cookies. Se verá también como manipular contenido HTML con la nueva etiqueta ContentEditable que soportan todos los campos de texto en el estándar HTML5. Habilitando esta etiqueta el contenido de un campo de texto se puede manipular, la principal ventaja es que al manipular el texto se conserva el formato HTML. El ejemplo permite almacenar fragmentos de texto como elementos independientes permitiendo altas, bajas, modificaciones y recuperaciones.

Además dará la opción de mostrar los mensajes utilizando la Notification API, ventanas emergentes que aparecen en la parte inferior derecha e informan de las novedades. Es una forma de mostrar mensajes emergentes de manera similar a las alertas del Messenger o de Correo.

En el caso del Session Storage permitirá arrastrar elementos, mediante Drag & Drop, al depositarlos en una caja de texto mostrará su contenido.

8.3.1.3. Caso de Uso detallado

Nombre	Ejemplo de Web Storage, Notificaciones y Drag & Drop
Actor Principal	Usuario
Descripción	- Se permite editar y guardar un fragmento de texto en la caché del navegador. En concreto en 2 tecnologías de almacenamiento: local Storage y session storage.
Extensiones	- Se produce un error si el navegador no soporta los objetos para manejar el almacenamiento en el Local Storage o el Session Storage. También aparece si no soporta Drag&Drop o las

	Notificaciones. - El fragmento de texto se puede manipular, permitiendo altas, bajas y modificaciones. - En el caso de Session Storage se muestra un ejemplo de drag & drop, al arrastrar un elemento a un cuadro de texto muestra su contenido. - Si se habilitan las notificaciones muestra ventanas emergentes con la acción realizada.
Datos	Fragmento de texto HTML a guardar y manipular

8.3.1.4. Datos que maneja

No maneja tablas ni accede a ninguna Base de datos. Todas las operaciones se realizan en el almacenamiento local del navegador.

Utiliza un fragmento de texto HTML, manipulable por el usuario, para modificarlo y almacenarlo.

8.3.1.5. Secuencia de acciones

Detalle de acciones

Después de solicitar la página HTML al servidor, el resto de las acciones se realizan en JavaScript en el navegador, no recurriendo al servidor en ningún caso.

- Altas, Bajas o Modificaciones de elementos, se realizan llamando a funciones JavaScript de nuevos objetos del DOM que incorpora HTML5.

Ejemplos código Javascript que se utiliza para manejar los elementos:

- *window.localStorage.setItem('clave','contenido')*
- *contenido = window.localStorage.getItem('clave')*
- *window.localStorage .removeItem('clave')*

En el Session Storage se aplicarían los siguientes:

- *sessionStorage.setItem('clave','contenido')*
- *contenido = sessionStorage.getItem('clave')*
- *sessionStorage.removeItem('clave')*

- Habilitar / Deshabilitar notificaciones. Se pregunta al usuario si desea ver las notificaciones de las acciones realizadas. Si se desactiva no se ven mensajes.

  Ejemplo. Lanzar mensajes de notificación:
  window.webkitNotifications.createNotification('./images/home_logo.jpg', titulo, textoNotificacion).show();

- Drag & Drop. Solo se ha desarrollado para el session storage. Define un evento (dragstart) para arrastrar, por elemento que se desee permitir arrastrar, y otro para el evento de soltar (dragover o dragenter).

Los datos que se pasan entre los ventos se reciben con el siguiente código:

[objeto evento].dataTransfer.getData('Text')

8.3.1.6. Manual de usuario

En la figura 5 nos muestra cómo podemos HABILITAR/DESHABILITAR NOTIFICACIONES. Para esto, el sistema pide permiso al usuario, que en cualquier momento puede realizar cualquiera de las 2 acciones. El número de mensajes mostrados simultáneamente suele ser tres, aunque en el Windows Vista es posible mostrar hasta cuatro.

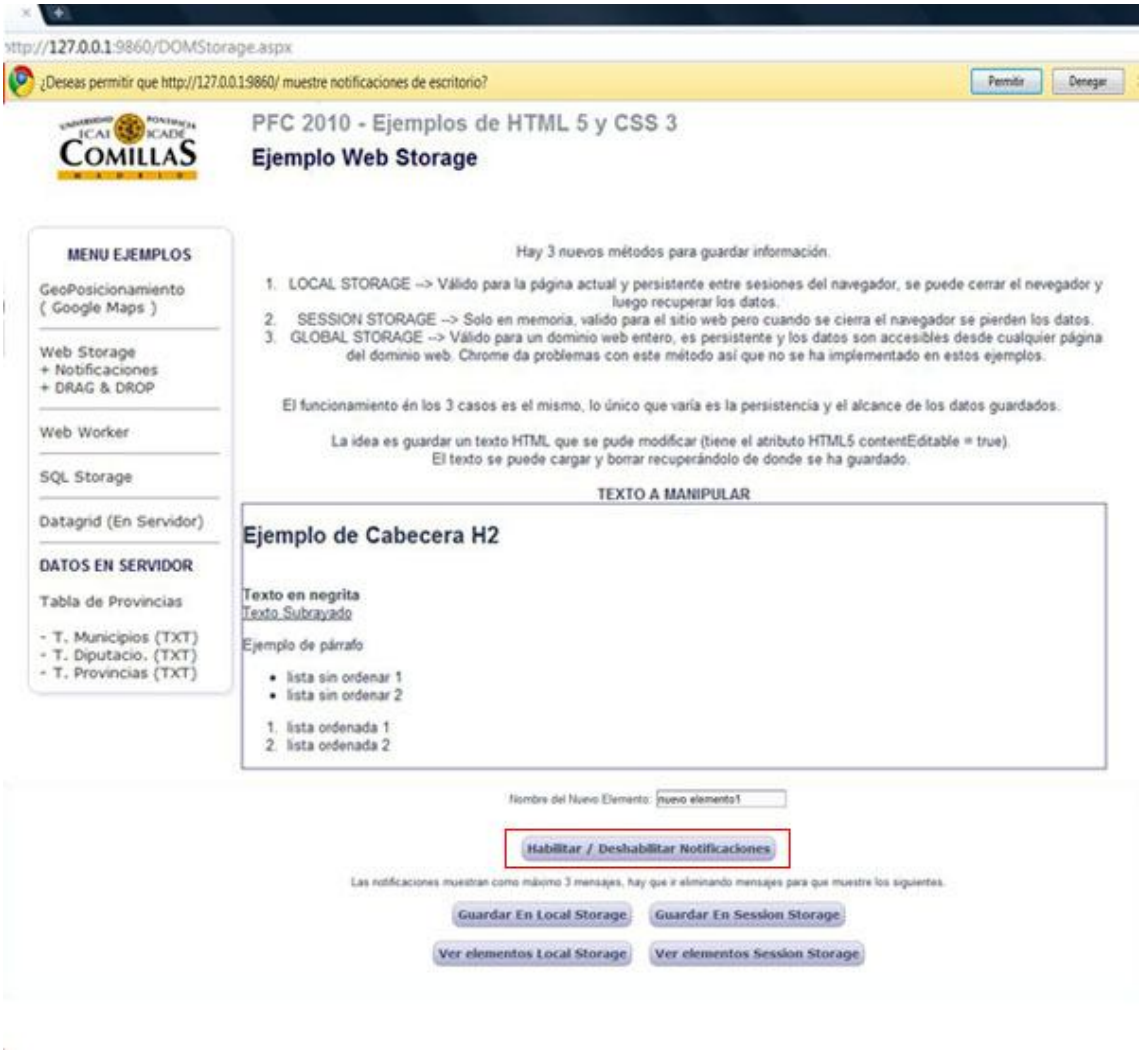


Figura 5 Habilitar o deshabilitar notificaciones

En la figura 6 muestra como se puede manipular el texto del recuadro y guardase directamente en el botón de LOCAL STORAGE, en ese caso de modificaría el documento anterior. También se da la opción de guardarlo como documento nuevo dándole un nombre diferente.

En la parte inferior nos muestra un listado con los documentos guardados, los cuales se pueden cargar para modificarlos o borrarlos.

Cada vez que se guarde, modifique o borre un elemento, si tenemos activadas las notificaciones, mostrara un mensaje con la acción realizada.

Figura 6 Local Storage

En la figura 7 muestra como los elementos vistos en la figura anterior pueden guardarse también en SESSION STORAGE, mostrando una nueva

opción DRAG & DROP. Pulsando sobre cualquier elemento de la lista se puede arrastrar al recuadro donde pone DRAG & DROP para ver el contenido.

Figura 7 Session Storage y DRAG & DROP

8.3.2. GEOPOSICIONAMIENTO

8.3.2.1. Estándar W3C al que se refiere. Geolocation API.

Está orientado sobre todo a dispositivos móviles con acceso a alguna red que proporcione información sobre su localización. Es agnóstica en cuanto a la tecnología, accede al hardware y obtiene una posición que es la que finalmente proporciona al navegador. Las formas de obtener la posición pueden ser:

- Mediante GPS
- Con aproximaciones utilizando redes móviles, por ejemplos GSM / CDMA

- Mediante información inalámbrica con redes WIFI, WIMAX, RFID, Bluetooth MAC...

- Mediante IPs de redes fijas.

La información obtenida es una posición que se puede cachear y luego trazar. En ningún caso se garantiza que se obtenga una posición real. El resultado de la consulta puede contener información adicional si el hardware lo soporta, por ejemplo la velocidad o la altura.

Coordenadas devueltas al consultar la posición

```
interface Coordinates {
  readonly attribute double latitude;
  readonly attribute double longitude;
  readonly attribute double altitude;
  readonly attribute double accuracy;
  readonly attribute double altitudeAccuracy;
  readonly attribute double heading;
  readonly attribute double speed;
};
```

Se implementa mediante el objeto **navigator.geolocation.** A la hora de devolver la posición, por cuestiones de seguridad se solicita confirmación al usuario. Esto es importante porque puede ser problemático que los navegadores obtuvieran la posición de los usuarios y la mandaran a un servidor para hacer un seguimiento.

Según esta especificación se pueden obtener las diferencias de posición en el tiempo. De manera que el navegador va recibiendo eventos periódicos con nuevas coordenadas y el espacio recorrido para que vuelva a pintar la posición. También, si se dispone del hardware necesario, puede indicar la precisión que maneja.

W3C – Grupo de Trabajo de Geolocation API

http://dev.w3.org/geo/api/

8.3.2.2. Requisitos

Deberá Geoposicionar la ubicación aproximada del navegador en el Google Maps utilizando la Geolocation API. El navegador proporcionará unas coordinadas que permitirán localizar un punto en el Google Maps.

Para pintar las coordenadas en un mapa se utiliza el API Javascript proporcionada por Google que en principio, mientras no cambien su política de negocio, es gratuita.

8.3.2.3. Caso de Uso detallado

Nombre	Ejemplo de Geoposicionamiento
Actor Principal	Usuario

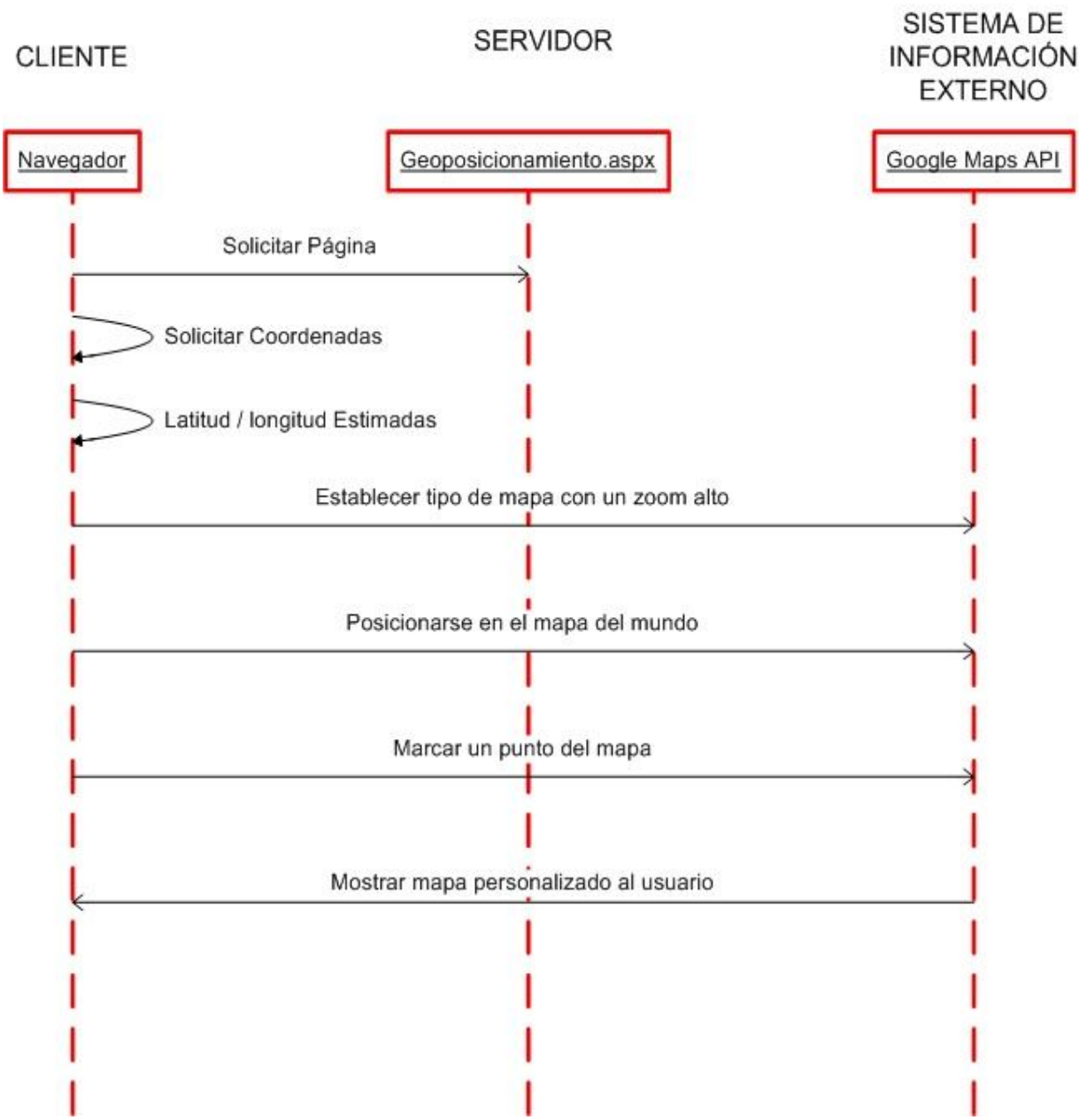
Descripción	- Se le solicita al usuario que permita enviar su posición al navegador. Por cuestiones de seguridad requiere la aprobación manual por el usuario. - Se muestra su posición utilizando el servicio de google maps. Se utilizan sus coordenadas en UTM.
Extensiones	Se puede producir error en el siguiente caso: - El navegador no soporta los objetos para manejar el geoposicionamiento.
Datos	Punto obtenido (latitud y longitud) por el navegador.

8.3.2.4. Datos que maneja

No requiere ningún dato del servidor ni del usuario. Las coordenadas que necesita se las pregunta directamente al navegador.

8.3.2.5. Secuencia de acciones

Diagrama de mensajes enviados entre los distintos sistemas que intervienen.



El código que se ejecuta es JavaScript y se realiza todo en el cliente.

- El navegador (Cliente) accede a la página que el servidor le devuelve como HTML y JavaScript.

- Esta página contiene un código JavaScript que se ejecuta nada más recibirla el cliente.

El código JavaScript es el siguiente:
navigator.geolocation.getCurrentPosition(pintarEnMapa, tratarErrores);

Llama a la función *pintarEnMapa* que recibe como parámetro un punto con las coordenadas UTM. Este punto lo proporciona el navegador y es parte de la interfaz que indica el estándar. Sería similar a un puntero a función.

- El JavaScript llama a un objeto de HTML5 que cumple la especificación de la Geolocation API que permite determinar la latitud y longitud del dispositivo en el sistema de coordenadas UTM. Estas coordenadas son las que utiliza Google Maps.

- Se utilizan las coordenadas devueltas por el navegador. El cómo las obtenga no es relevante, si no se dispone de ningún hardware GPS seguramente lo haga consultando por la IP.

- Se llama a Google Maps, definiendo el tipo de mapa a mostrar y el zoom. Se utiliza el objeto *google.maps.Map()*.

El API de Google Maps utilizada (V3) se puede consultar en:

http://code.google.com/intl/es-ES/apis/maps/documentation/javascript/reference.html

- Se posiciona en un punto del mapa del mundo y se incorporan los datos de latitud y longitud en ese punto. Para ello se utiliza la clase *google.maps.InfoWindow()*.

8.3.2.6. Manual de usuario

Este ejemplo se ejecuta simplemente con acceder a la página, el resultado debería ser similar a este:

Es necesario tener conexión a Internet y específicamente a Google Maps. Al utilizar un servicio externo hay que considerar que parte del comportamiento de este ejemplo depende de las decisiones que tome google a la hora de publicar sus mapas, por ejemplo restringiendo su acceso o limitando el número de consultas gratuitas por día. Es un riesgo más que hay que considerar cuando se utilizan sistemas de información externos y que por tanto no podemos controlar.

8.3.3. WEB WORKERS (Hilos de ejecución)

8.3.3.1. Estándares W3C a los que se refiere

Siguiendo con la evolución de los navegadores, los cuales se parecen cada vez más a sistemas operativos, se incorporan los Web Workers para manejar y sincronizar hilos de ejecución en paralelo. La sincronización se hace mediante paso de mensajes.

Hay que considerar que ejecutan el contenido de ficheros javascript separados y que no tienen acceso a los objetos de la página HTML que les llama y a muchos de los objetos que proporciona el navegador. Lo que hacen es mandar a la página que les llamó mensajes con objetos, en la práctica lo normal es usar tipos de datos muy básicos o pasar mensajes de texto para que los interprete la página que les llamó.

Manejo de los Web Worker en la aplicación

Creación de los Hilos de ejecución

```
var worker = new Worker('EjemploHilo.js');
```

var worker2 = new Worker('EjemploHilo2.js');

Cada hilo devolverá mensajes a la página utilizando la función:

postMessaje(objeto)

La especificación también contempla los SharedWorkers que son hilos de ejecución que pueden tener varias conexiones con páginas web. Son bastante complejos y en el ejemplo no se contemplan.

W3C – Grupos de Trabajo

HTML 5 – Web Workers

http://dev.w3.org/html5/workers/

W3C –Web Database

http://dev.w3.org/html5/webdatabase/

8.3.3.2. Requisitos

Ejemplo de 2 hilos de ejecución. Los hilos llaman al servidor le solicitan cada uno una tabla de datos en formato texto. Comienzan a descargar de manera asíncrona e independiente. Avisando a la página que los lanzó cuando lleva un determinado número de filas, indicando los instantes de tiempo en que van recibiendo las filas. El objetivo es mostrar cómo usar esta tecnología para realizar procesos en background y en paralelo.

Después de la carga se podrán visualizar las tablas descargadas y almacenadas en la Base de datos local del navegador (SQL Storage).

Las mismas tablas descargadas son los nombres de las provincias y el importe cobrado por los municipios en el ejercicio 2009. Al ser datos económicos se realizarán cálculos en el navegador de los siguientes datos: Totales por provincia y porcentaje de ingresos percibidos por cada municipio. Mostrando el resultado de los cálculos como tablas HTML. Se

hará un JOIN (unión) en la base de datos local de la tabla de nombres de provincias con lo cobrado por los municipios. El objetivos es hacer consultas de selección un poco complejas con los datos y mostrar cómo realizar cálculos en cliente sin tener que volver a llamar al servidor.

8.3.3.3. Caso de Uso detallado

Nombre	Ejemplo de Web Worker
Actor Principal	Usuario
Descripción	- Se crean 2 hilos de carga que se ejecutan simultáneamente. Cada hilo descarga una tabla distinta al navegador que se almacenan en el Session Storage. - Se van mostrando los mensajes de carga y el instante temporal en que se producen. - Se muestran las tablas descargadas como un volcado de la base de datos a tablas HTML.
Extensiones	- Se produce un error si el navegador no soporta los objetos para manejar los web workers o el

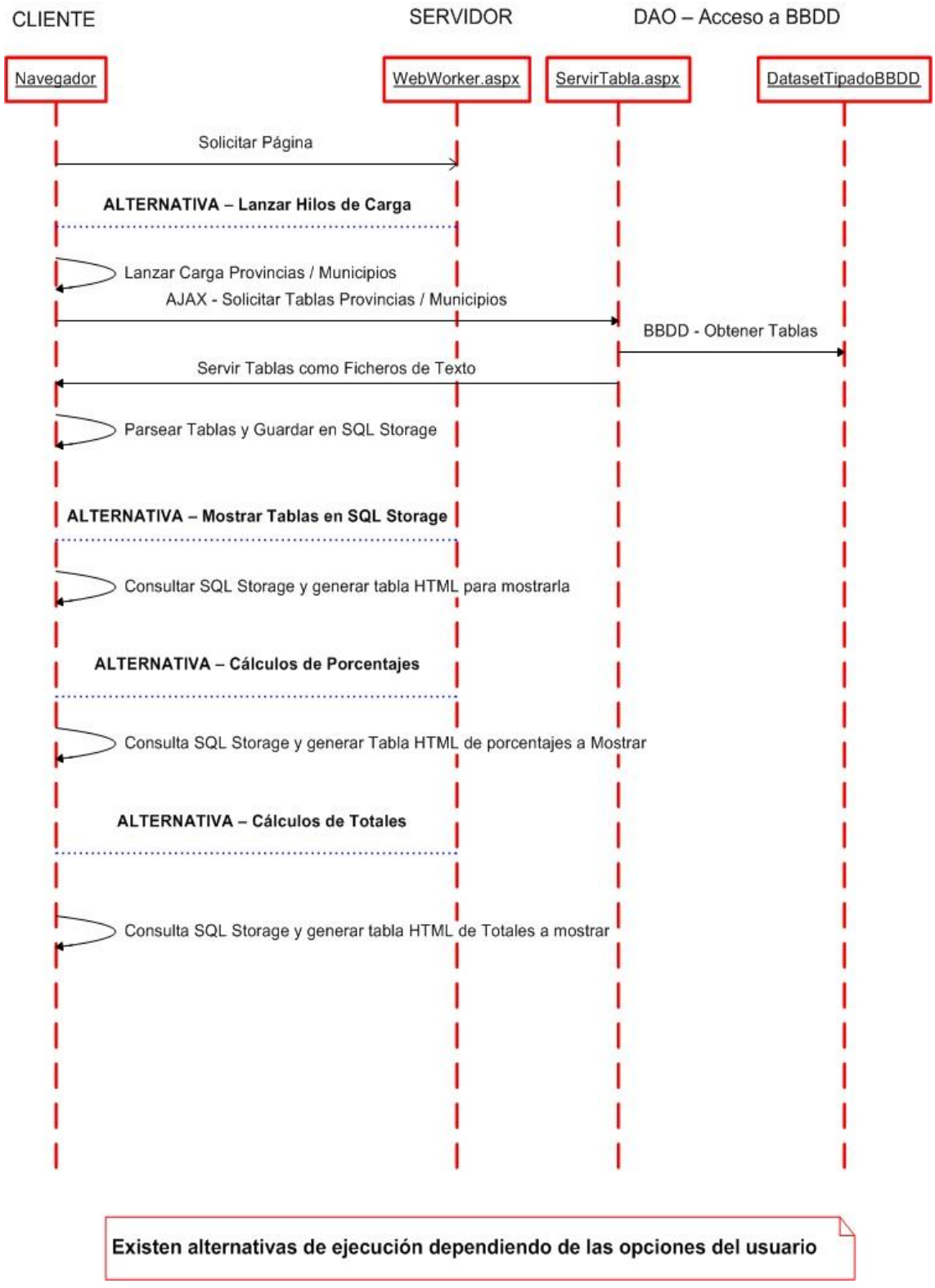
	almacenamiento en Session Storage. - Permite realizar cálculos y consultas sobre los datos almacenados en SQL Storage y mostrarlos como tablas HTML.
Datos	Tablas de Municipios y Provincias servidas como texto

8.3.3.4. Datos que maneja

Tablas de Provincias y CobradoMunicipiosVariable.

En modo de solo lectura, se descarga las tablas a una base de datos local que es la que utiliza. En ningún caso modifica los datos del servidor.

8.3.3.5. Secuencia de acciones



Detalle de las acciones

Lanzar Hilos de Carga. Se lanzan 2 procesos en background, hilos de ejecución, que ejecutan el código de 2 ficheros JavaScript. Estos ficheros llaman a la página ServirTabla.aspx utilizando AJAX. Le solicitan 2 tablas en formato texto que se las devuelve como 2 ficheros TXT que descargan y procesan obteniendo sus filas. Finalmente introducen cada fila en sus respectivas tablas de una base de datos local creada previamente. La página ServirTabla.aspx utiliza los métodos de la clase DataSetTipadoBBDD para acceder a la Base de Datos.

SQL de creación de las tablas:

1. *CREATE TABLE IF NOT EXISTS CobradoMunicipiosVariable (CodProvincia REAL, CodCorporacion REAL, NombreCorporacion TEXT, Cobrado REAL, Ejercicio TEXT)*
2. *CREATE TABLE IF NOT EXISTS Provincias (CodProvincia REAL UNIQUE, Provincia TEXT)*

Mostrar tablas almacenadas en el SQL Storage (Navegador). Se lanzan dos consultas sobre el SQL Storage recuperando todos los datos de las 2 tablas y se genera código HTML para mostrarlas mediante la etiqueta <table>.

Realizar cálculos sobre las tablas almacenadas en el SQL Storage (Navegador). Realiza consultas más complejas para forzar un poco al motor de base de datos SQLLite. Así se puede comprobar que sintaxis

maneja y su rendimiento. El resultado se pasa a una tabla HTML (etiqueta <table>) que se muestra.

Consultas SQL utilizadas

Se puede ver como se utilizan funciones de agregado y se anidan consultas y se realizan uniones entre 2 tablas.

- **Total cobrado por provincia.**

```
SELECT
CobradoMunicipiosVariable.CodProvincia,
Provincia,
SUM(Cobrado) as TotalProvincia,
Ejercicio
FROM
CobradoMunicipiosVariable, Provincias
WHERE
CobradoMunicipiosVariable.CodProvincia =
Provincias.CodProvincia GROUP BY
CobradoMunicipiosVariable.CodProvincia, Ejercicio
ORDER BY
CobradoMunicipiosVariable.CodProvincia, Ejercicio
```

- **Porcentaje Cobrado por municipio.**

```
SELECT
CobradoMunicipiosVariable.CodProvincia,
Provincia, NombreCorporacion,
(((Cobrado/(SELECTSUM(Cobrado)FROM
CobradoMunicipiosVariable))) * 100) as Total,
```

```
Cobrado,
Ejercicio
FROM
CobradoMunicipiosVariable, Provincias
WHERE
CobradoMunicipiosVariable.CodProvincia=
Provincias.CodProvincia ORDER BY
CobradoMunicipiosVariable.CodProvincia,
CobradoMunicipiosVariable.CodCorporacion
```

8.3.3.6. Manual de usuario

En la figura 8 nos muestra como se ejecuta la carga al lanzar un hilo. Muestra cómo se van cargando los datos del servidor a la BBDD local

Figura 8 Lanzar un hilo de carga

En la figura 9 se muestran las tablas cargadas en local

En la parte inferior, se puede ver como navegador ha guardado las tablas en una Base de Datos local. Esta parte se muestra pulsando Ctrl+Mayús+I

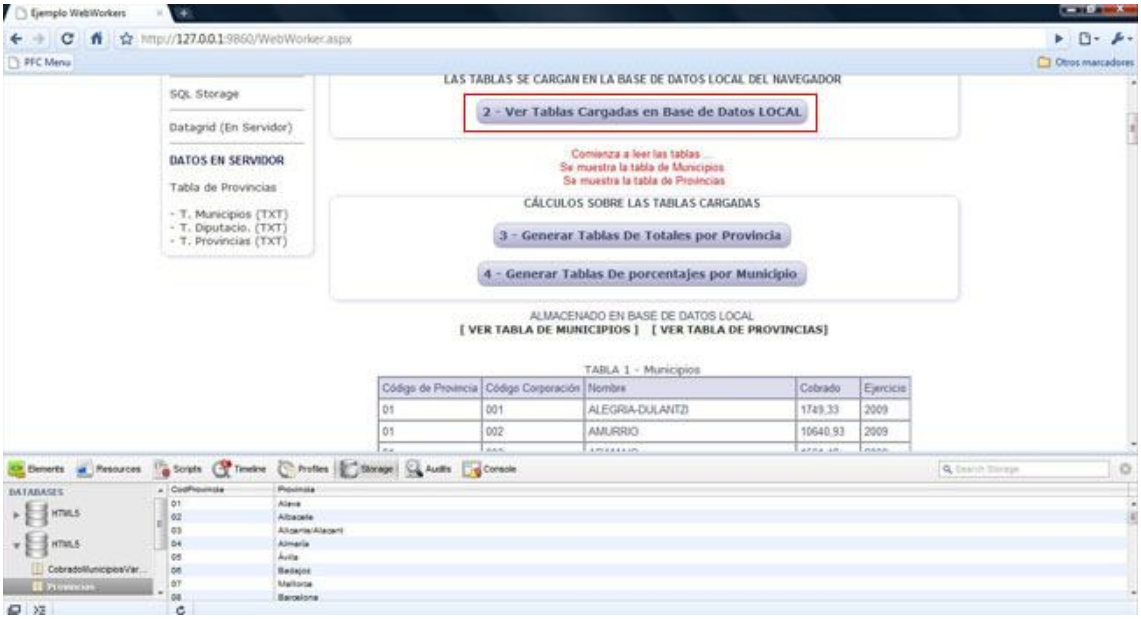
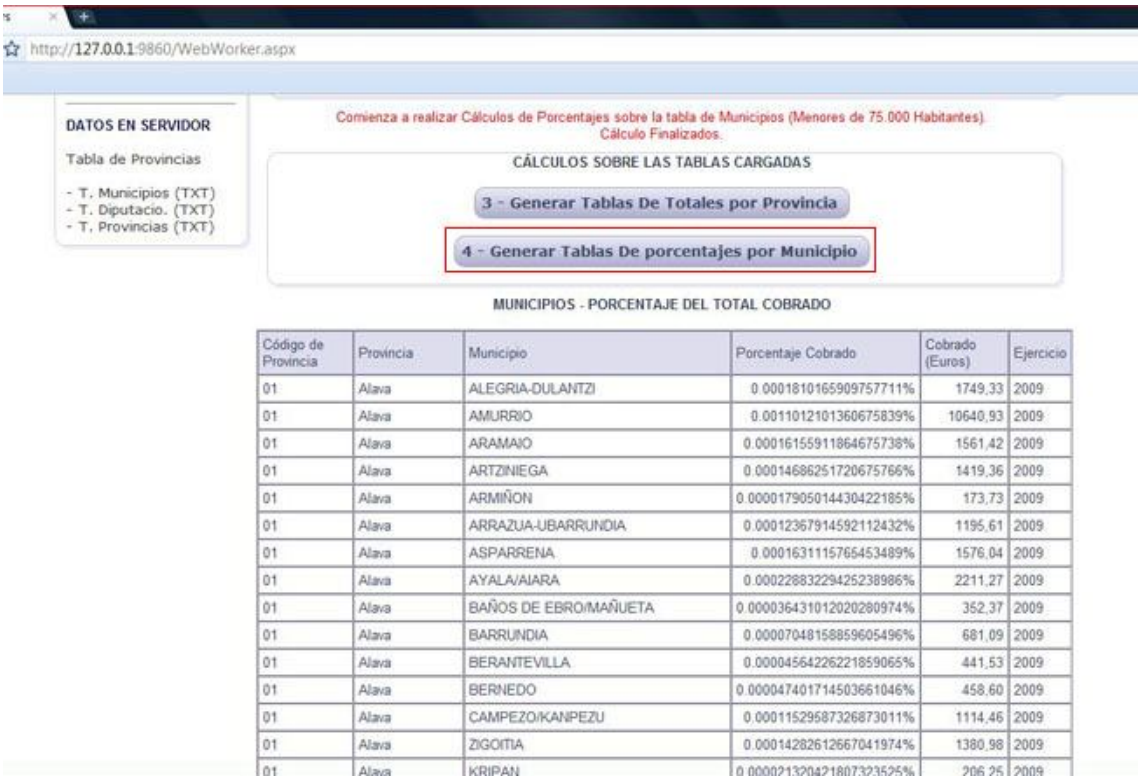


Figura 9 Muestra las tablas cargadas en la BBDD Local

En las figuras 10 y 11 nos muestra cómo se pueden realizar cálculos y pintar las pantallas en el navegador sin guardar los datos en ningún sitio, ni en servidor ni en el cliente.

Código de Provincia	Nombre de Provincia	Cobrado (Euros)	Ejercicio
01	Alava	363352	2009
02	Albacete	35814170	2009
03	Alicante/Alacant	197816177	2009
04	Almería	64643746	2009
05	Ávila	16946245	2009
06	Badajoz	76649146	2009
07	Mallorca	114926949	2009
08	Barcelona	124405075	2009

Figura 10 **Tabla de cálculos totales**



Código de Provincia	Provincia	Municipio	Porcentaje Cobrado	Cobrado (Euros)	Ejercicio
01	Alava	ALEGRIA-DULANTZI	0.00018101659097577 11%	1749,33	2009
01	Alava	AMURRIO	0.00110121013606758 39%	10640,93	2009
01	Alava	ARAMAIO	0.00016155911864675738%	1561,42	2009
01	Alava	ARTZINIEGA	0.00014686251720675766%	1419,36	2009
01	Alava	ARMIÑON	0.00001790501443042218 5%	173,73	2009
01	Alava	ARRAZUA-UBARRUNDIA	0.00012367914592112432%	1195,61	2009
01	Alava	ASPARRENA	0.00016311157654534 89%	1576,04	2009
01	Alava	AYALA/AIARA	0.00022883229425238986%	2211,27	2009
01	Alava	BAÑOS DE EBRO/MAÑUETA	0.00003643101202028097 4%	352,37	2009
01	Alava	BARRUNDIA	0.00007048158859605496%	681,09	2009
01	Alava	BERANTEVILLA	0.00004564226221859065%	441,53	2009
01	Alava	BERNEDO	0.00004740171450366104 6%	458,60	2009
01	Alava	CAMPEZO/KANPEZU	0.00011529587326873011%	1114,46	2009
01	Alava	ZIGOITIA	0.00014282612667041974%	1380,98	2009
01	Alava	KRIPAN	0.00002132042180732352 5%	206,25	2009

Figura 11 **Tabla de porcentajes por Municipio**

8.3.4. DATAGRID

8.3.4.1. Estándares W3C a los que se refiere

Etiqueta Datagrid

Permite manipular y trabajar con gran cantidad de datos en el cliente, sin tener que recargar la página y liberando al servidor de mucho trabajo. Actualmente se está emulando con componentes en prácticamente todos los lenguajes orientados a la Web. Por ejemplo en ASP.NET existe el GridView y en Java existen componentes de Java Server Faces que generan código HTML 4 para realizar la misma labor. Sin embargo su implementación en el lado cliente es muy complicada y lo que se está haciendo es recargar frecuentemente las páginas HTML enviando al cliente una y otra vez la misma información.

La especificación de este control permite editar el contenido, borrar filas, ocultar parte de la tabla, ordenar y finalmente enviar los cambios al servidor para que los pueda procesar.

Por ahora este componente no tiene implementación conocida en ningún navegador.

El uso básico de la etiqueta sería algo similar a esto.

```
<datagrid>
  <table>
  <tr>
        <td>Jones</td>
        <td>Allison</td>
        <td>A-</td>
        <td>B+</td>
```

```
        <td>A</td>
    </tr>
    </table>
</datagrid>
```

W3C – Grupo de Trabajo

http://dev.w3.org/html5/spec/Overview.html

8.3.4.2. Requisitos

Ejemplo de emulación, en servidor, de la futura etiqueta del componente Datagrid en HTML5, todavía no existen buenas implementaciones en los navegadores actuales. Tiene las opciones de editar un dato, borrarlo, insertarlo, ordenar y paginar. La futura etiqueta podrá realizar operaciones similares en el navegador sin necesidad de recargar la página contínuamente.

8.3.4.3. Caso de Uso detallado

Nombre	Ejemplo de Datagrid
Actor Principal	Usuario
Descripción	- Implementación con tecnología .NET de un ejemplo de cómo sería el futuro componente de Datagrid. Permite inserciones, edición y borrado de elementos de una tabla.
Extensiones	

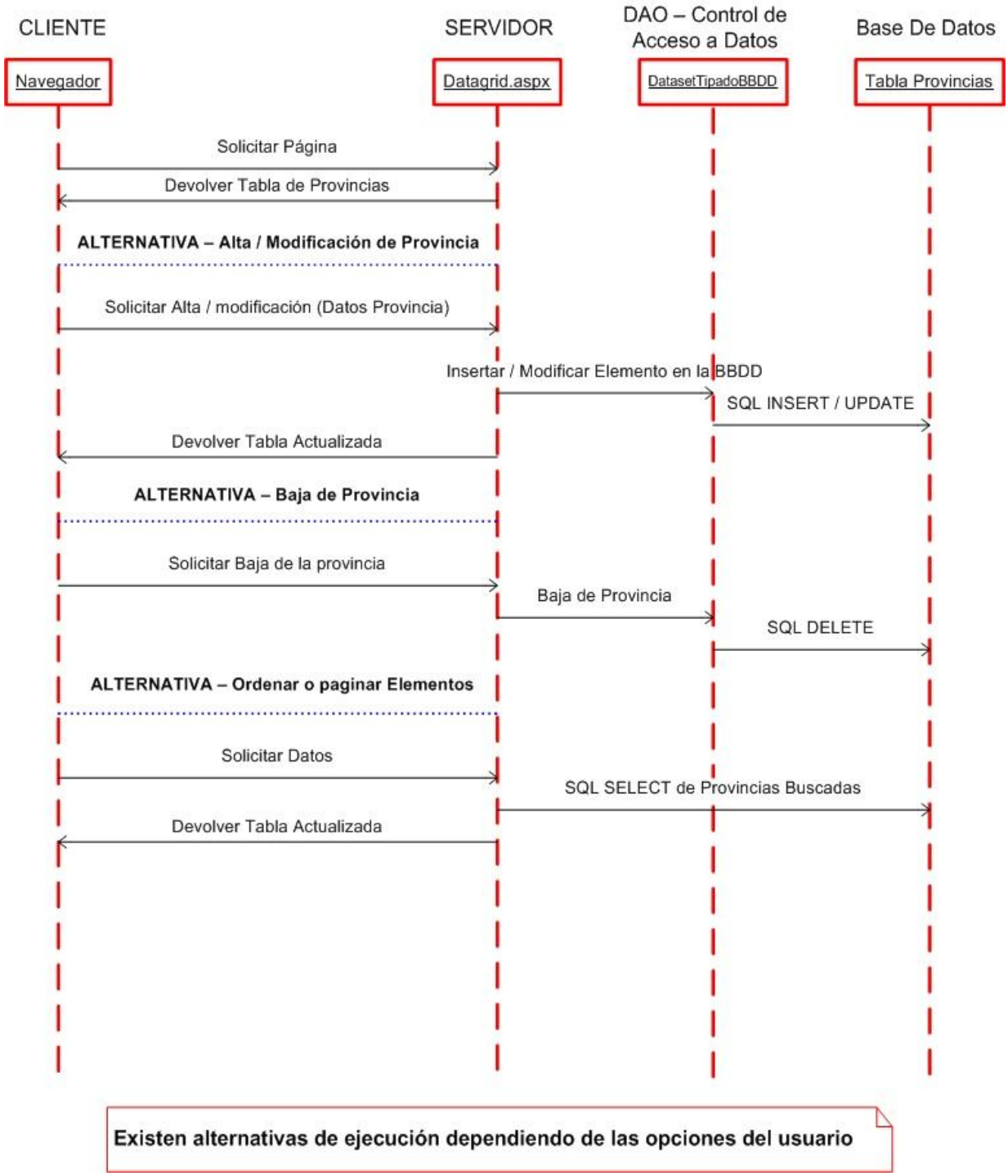
	- Se produce un error si el servidor no puede acceder a su base de datos.
Datos	Tabla de Diputaciones de la base de datos.

8.3.4.4. Datos que maneja

Accede a la tabla de datos cobrados por las diputaciones en el ejercicio 2009. El accesos es de lectura / escritura por lo que puede cambiar su contenido.

8.3.4.5. Secuencia de acciones

Mensajes enviados entre los distintos componentes que intervienen.



Detalle de las acciones

Solicitar tabla de Diputaciones. Se le pide al servidor la tabla de diputaciones con unos criterios de ordenado y paginación. El servidor (página Datagrid.aspx) utiliza un control de datos y otro de visualización que no hacen necesario programar para generar la tabla que se muestra en el navegador, todo se realiza mediante asistentes del visual Studio liberando al programador de tareas repetitivas.

Los controles ASP.NET utilizados son los siguientes:

<*asp:GridView …* > → Presentación de la tabla, se le asocia el ObjectDataSource como origen de los datos.

<*asp:ObjectDataSource …* > → Acceso al Dataset Tipado. Se indica que métodos y que parámetros utiliza para hacer borrados, inserciones, consultas y modificaciones.

Altas, bajas y modificaciones de diputaciones. Se realizan a través del Dataset Tipado que controla el acceso a la Base de Datos. Después de realizar las modificaciones en base de datos genera otra vez la tabla en HTML y se la vuelve a mandar entera al cliente.

Ejemplo de código necesario para dar de baja una diputación:

AppCode.DatosTipadosBBDDTableAdapters.DiputacionesTableAdapter adaptador
=

```
new
pfc_html5.AppCode.DatosTipadosBBDDTableAdapters.DiputacionesTa
bleAdapter();

adaptador.DeleteQuery( [ID de Diputacion] );
```

El Dataset Tipado (Clase DatosTipadosBBDD) es la capa que controla el acceso a la base de datos y controla que el tipo de datos de todos los campos que se mandan y reciben sea correcto. Además permite abstraer a la aplicación del almacenamiento de los datos facilitando el cambiar el motor de base de datos empleado.

8.3.4.5. Manual de usuario

El DATAGRID, nos muestra las posibilidades a realizar con una tabla en local.

Esta la figura 12 y la figura 13 se muestra todas las acciones que se pueden realizar sobre una tabla :

- Paginado
- Actualizar una fila: Si pulsamos sobre este botón se nos muestran unas cajas de texto donde tenemos la posibilidad de modificar los datos.
- Borrar una fila
- Añadir una nueva fila

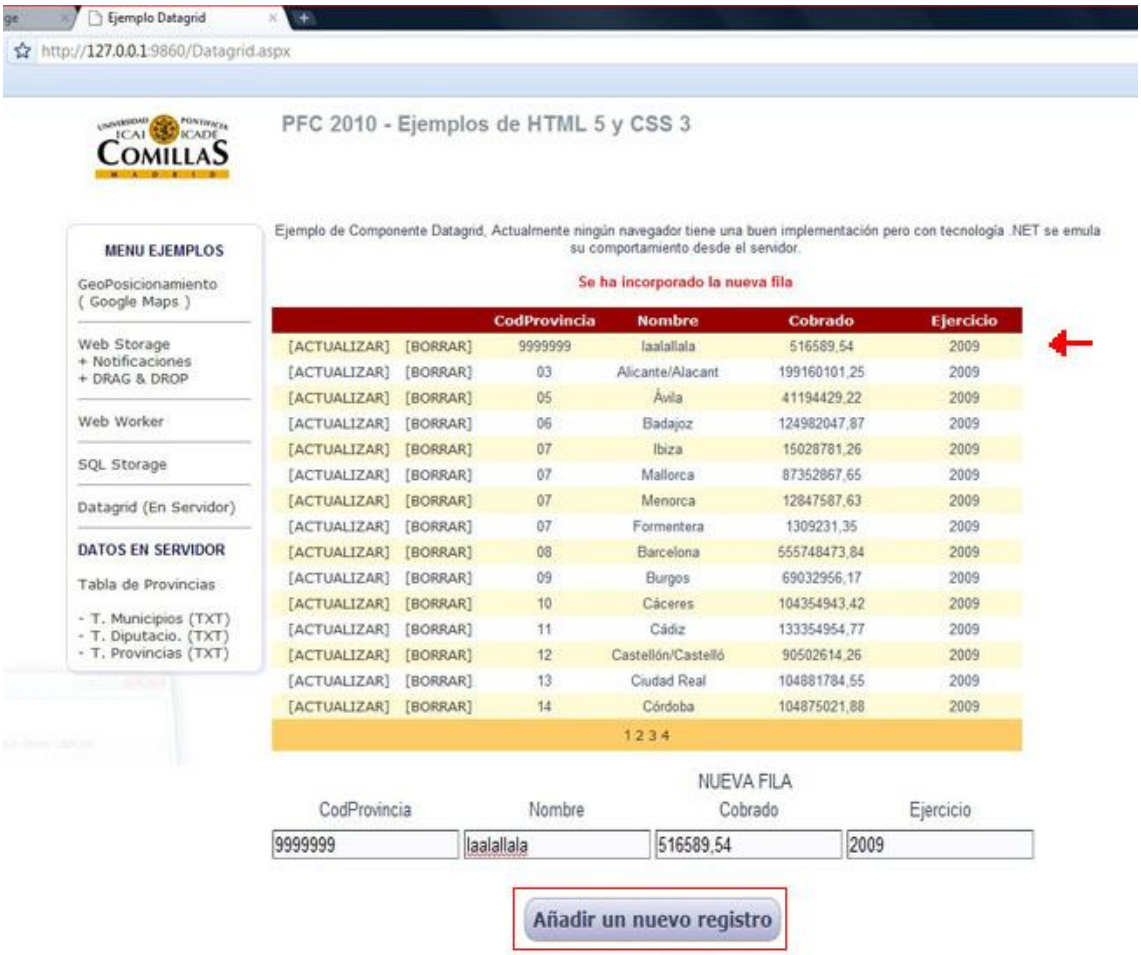


Figura 12 **Añadir un nuevo registro a la tabla**

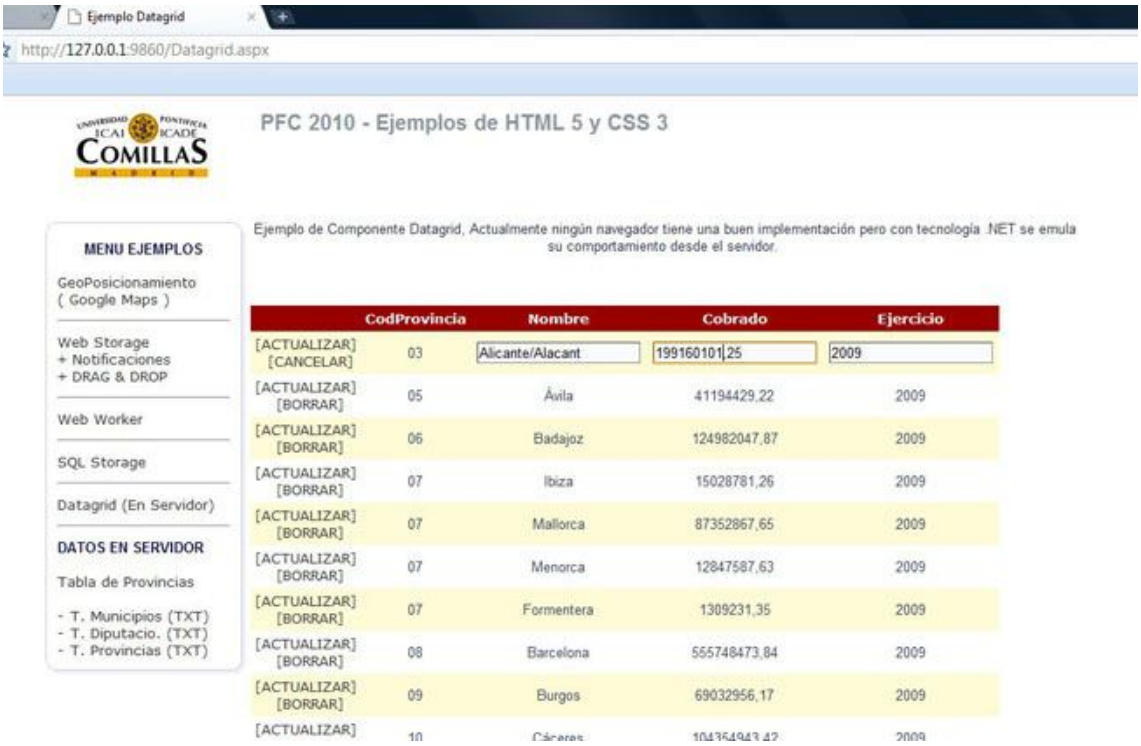


Figura 13 **Modificar un registro de la tabla**

8.3.5. SQL STORAGE

8.3.5.1. Estándares W3C a los que se refiere

Propuesta de W3C para sustituir el almacenamiento de datos en cliente, hasta el momento mediante cookies, por una base de datos local y limitada. Algunas implementaciones permiten hasta 20 Mb por sitio Web (la recomendación de W3C son un mínimo de 5 Mb).

Las ventajas son evidentes:

- Se puede copiar parte del modelo de datos del servidor al cliente y así no hay que estar mandando continuamente información.
- La información se puede relacionar de manera compleja, mediante consultas SQL.

- Utilizado con la File API, que permite almacenar ficheros en cliente, se puede distribuir mejor la información y hacer aplicaciones distribuidas.

Sin embargo también tiene desventajas a tener en cuenta:

- Los problemas de seguridad, el usuario tiene que ser consciente de la información que comienza a almacenar. Los datos no deben ser accedidos por terceros no autorizados.

- La actualización de los datos, el usuario puede tener la falsa creencia de que lo que está visualizando son los datos actualizados del servidor.

Las implementaciones existentes por el momento están basadas en el motor SQLLite, una base de datos relacional pensada para dispositivos móviles o con recursos limitados. El dialecto SQL que soportan es por tanto el del SQLLite y la W3C ya ha alertado (Mayo de 2010) que le gustaría contar con más implementaciones antes de proponer el dialecto SQL del SQLLite como estándar.

Métodos y Objetos

- **OpenDatabase() / Sync** → Abre la base de datos de manera asíncrona, existe una versión síncrona del método que espera a obtener una respuesta.
- **Transaction / Sync** → Transacciones en lectura / escritura.

Cuentan con el método executeSQL() que es el que realmente lanza los comandos SQL.

- **readTransaction / Sync** → Transacciones en solo lectura.

- **changeVersion()** → Accede a la BBDD, data una versión de la misma.

Utiliza lamda expresions para declarar las funciones, en concreto en el ejemplo la variable t (transacción SQL) se declara de esta forma y la variable resultante de la ejecución r (ResultSet) también. Admite consultas SQLECT, INSERT, UPDATE, DELETE, CREATE … y en general todo lo que soporta SQLLite porque en la práctica es la implementación de referencia.

W3C –Grupo de Trabajo de Web Database

http://dev.w3.org/html5/webdatabase/

8.3.5.2. Requisitos

Ejemplo de almacenamiento en una Base de datos situada en el navegador (cliente). Se solicitará una tabla de Base de datos mediante AJAX. La tabla se manda como texto, se trata, se obtienen sus datos y se almacenarán en una Base de datos en el navegador (SQL Storage). Se utilizan consultas de creación, inserción y modificación de los datos en local, pudiendo trabajar con ellos sin volver a llamar al servidor.

Se dará la opción de visualizar y trabajar con la tabla en local y posteriormente se sincronizarán los datos con el servidor. Se envían

como texto, el servidor los procesa y actualiza su base de datos. Es un ejemplo de cómo manipular tablas en el navegador y subir los cambios.

8.3.5.3. Cuestiones a considerar

Al ser un ejemplo más académico se ha desarrollado pensando en la parte cliente. Sin embargo en el servidor habría que realizar muchas comprobaciones antes de actualizar los datos mandados por el navegador.

- Se deberá cuidar mucho la seguridad y mandar los datos cifrados, mediante conexiones HTTPS con certificados o mecanismos similares que garanticen la autenticidad e integridad de los datos.

- Seguramente habrá que guardar varias versiones del mismo dato en los casos que varios navegadores modifiquen el mismo datos.

- Luego habría que establecer mecanismos para la resolución de conflictos.

- Pueden aparecer condiciones de carrera al modificar un dato.

- Si los datos fueran sensibles habría que cifrarlos en el navegador porque sino se tiene un agujero de seguridad que afecta a todo el sistema de información.

En general el manejo de base de datos distribuídas es muy complejo y podría dar para hacer una aplicación específica solo para este apartado.

8.3.5.4. Cambio de paradigma. Tratamiento de los datos en cliente

Lo habitual en los desarrollos Web es realizar todos los cálculos complicados en el servidor y enviarle al navegador páginas sencillas de HTML para que visualice el resultado. Cada vez que se quiere realizar el mínimo cálculo o mostrar los datos de otra manera se vuelven a solicitar todos al servidor y este genera una página HTML nueva. En este caso concreto se realizan en el navegador muchas de las tareas asignadas normalmente al servidor. Esto implica el disponer de bases de datos distribuidas en cada cliente con la complejidad que esto conlleva para sincronizarlas.

Se aprovechan las características de javascript para generar y modificar el código HTML de la página de manera dinámica, sin recargarla, y se utiliza el SQL Storage para poder manipular la información mediante consultas SQL.

8.3.5.5. Caso de Uso detallado

Nombre	Ejemplo de SQL Storage
Actor Principal	Usuario
Descripción	- Se crea, si no existe, una base de datos en el navegador. - Se abre la base de datos y se obtiene las filas de la tabla de provincias. Los datos, obtenidos como texto, se parsear y se introducen en la base de datos del navegador. - Con esos datos se puede trabajar en el navegador sin estar conectado al servidor. Se pueden mostrar, borrar y modificar.
Extensiones	- Se produce un error si el navegador no soporta los objetos para manejar el SQL Storage. En la práctica es un motor de base de datos en memoria. - Se permite sincronizar los cambios con el servidor.
Datos	Tabla de Provincias de la base de datos del servidor.

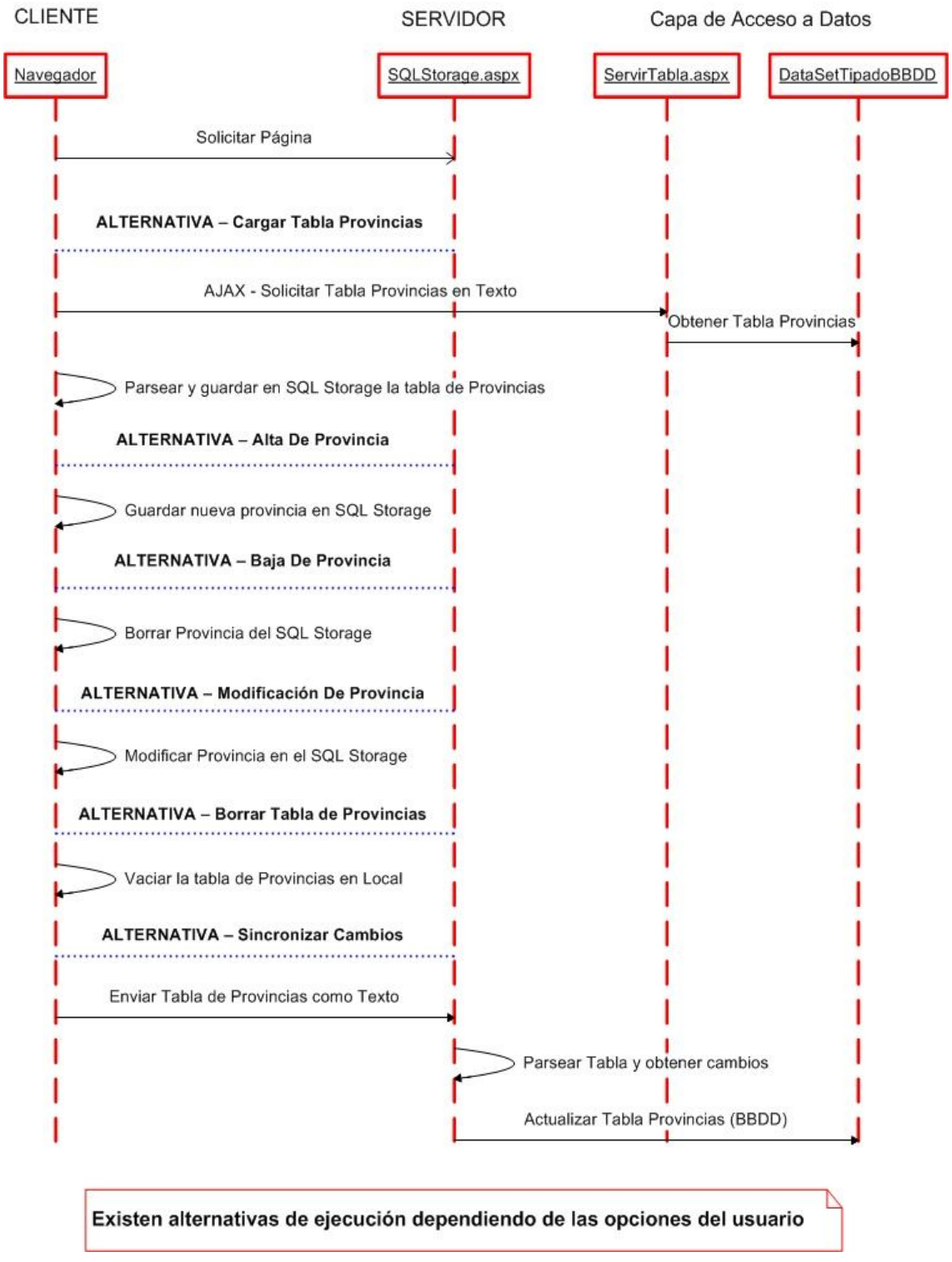
8.3.5.6. Datos que maneja

Tabla de Provincias del Servidor.

En modo de lectura y Escritura. Pudiendo modificar la tabla que se almacena en el servidor.

8.3.5.6. Secuencia de acciones

Diagrama de mensajes enviados entre los distintos componentes que intervienen.



Detalle del Diagrama

El código ejecutado en el navegador es Javascript que accede a los nuevos objetos del estándar HTML5 para manejar el motor de base de datos SQLLite.

Carga de Provincias. Hace una llamada AJAX a una página del servidor solicitando la tabla de provincias. La llamada a la página realizada es:

ServirTabla.aspx?nombreTabla=Provincias

La tabla se sirve como un fichero de texto donde las filas se separan con salto de línea y las columnas por '|'. Esta tabla se partea en javascript y se obtienen sus elementos que se insertan en una tabla creada en el SQL Storage del navegador.

Las consultas SQL que se ejecutan son las siguientes (sintaxis de SQLLite):

1. *CREATE TABLE IF NOT EXISTS Provincias (CodProvincia REAL UNIQUE, Provincia TEXT)*

2. *INSERT INTO Provincias (CodProvincia, Provincia) VALUES (?, ?)*

   Como se puede observar la consulta de inserción va parametrizada.

Altas, bajas, modificaciones y listado de Provincias. Se realizan lanzando consultas SQL al motor del navegador.

Las consultas serían las siguientes:

1. DELETE FROM Provincias WHERE CodProvincia = ?
2. UPDATE Provincias SET Provincia = ? WHERE CodProvincia = ?
3. INSERT INTO Provincias (CodProvincia, Provincia) VALUES (?, ?)
4. SELECT * FROM Provincias ORDER BY CodProvincia

Sincronizar Cambios en la tabla de provincias con el servidor. Se le manda, en un campo del formulario, la tabla de provincias serializada como texto. Este texto lo procesa el servidor y actualiza cada fila de la tabla con los datos enviados. Luego devuelve el número de filas actualizadas al navegador.

8.3.5.7. Manual de usuario

En la figura 14 muestra cómo crear una BBDD local. En la parte inferior de la pantalla muestra que se ha creado y su contenido (Ctrl+Mayús+I)

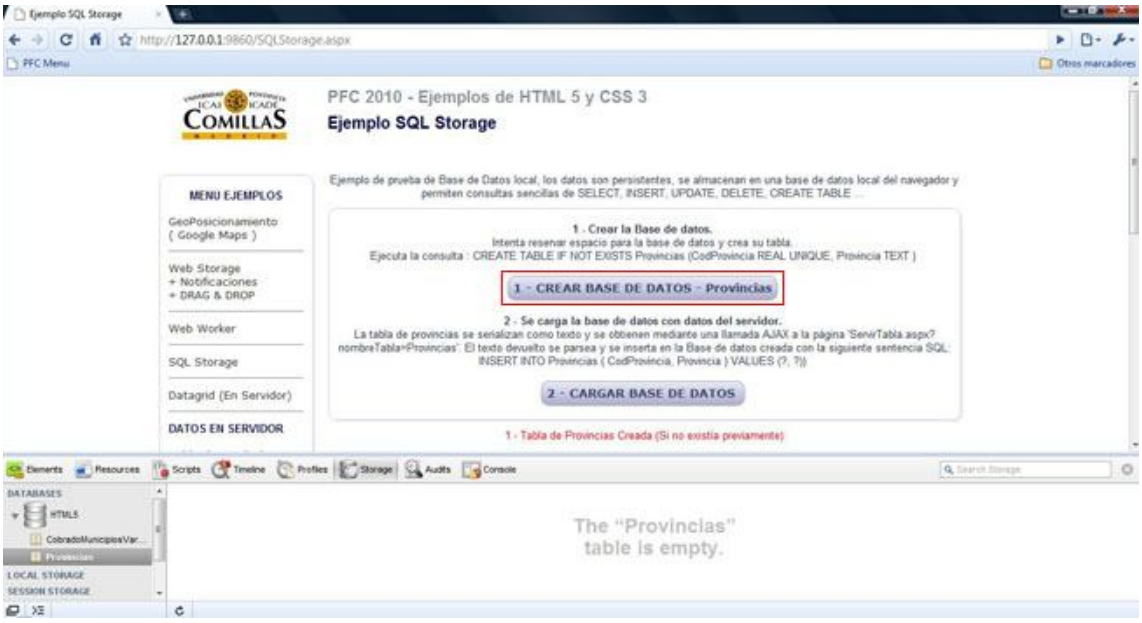


Figura 14 Crear una BBDD en local

En la figura 15 se muestra el volcado de los datos del servidor en la BBDD local creada en la figura 14

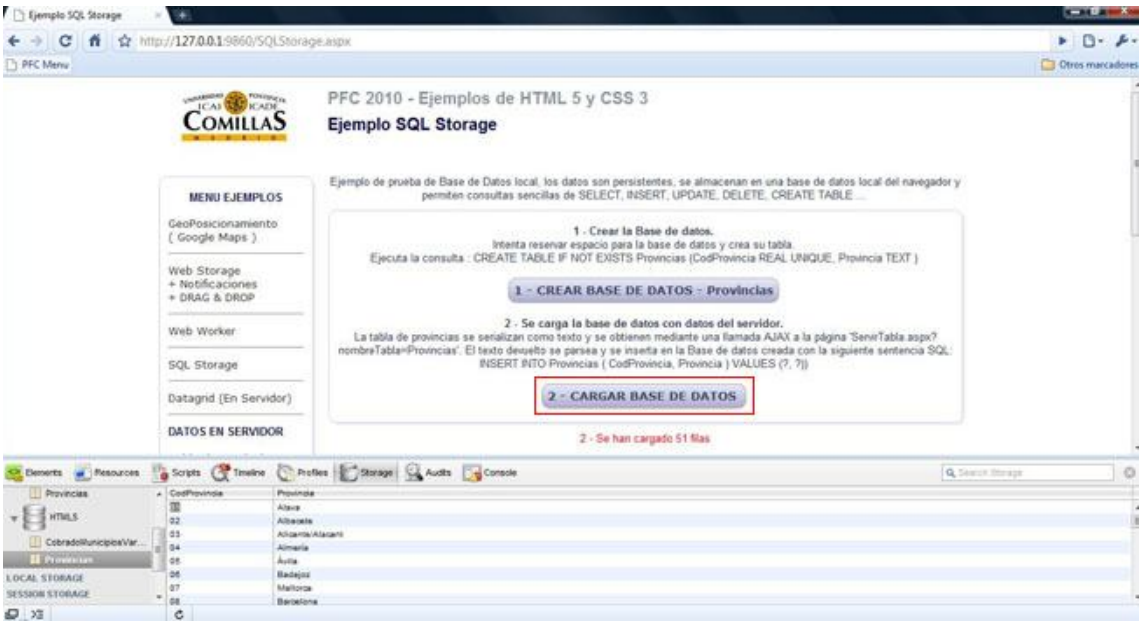


Figura 15 Carga de la BBDD Local

En la figura 16 muestra por pantalla la tabla de Provincias, creada y cargada en local. En esta tabla se pueden realizar cambios sobre los registros o añadir uno nuevo. Estos cambios se pueden sincronizar con la BBDD del servidor

Figura 16 Tabla de Provincias de la BBDD Local

En la figura 17 se muestra la sincronización de la BBDD Local con la BBDD del Servidor para actualizar los cambios. Estos cambios se muestran en “ver tabla de provincias del servidor”

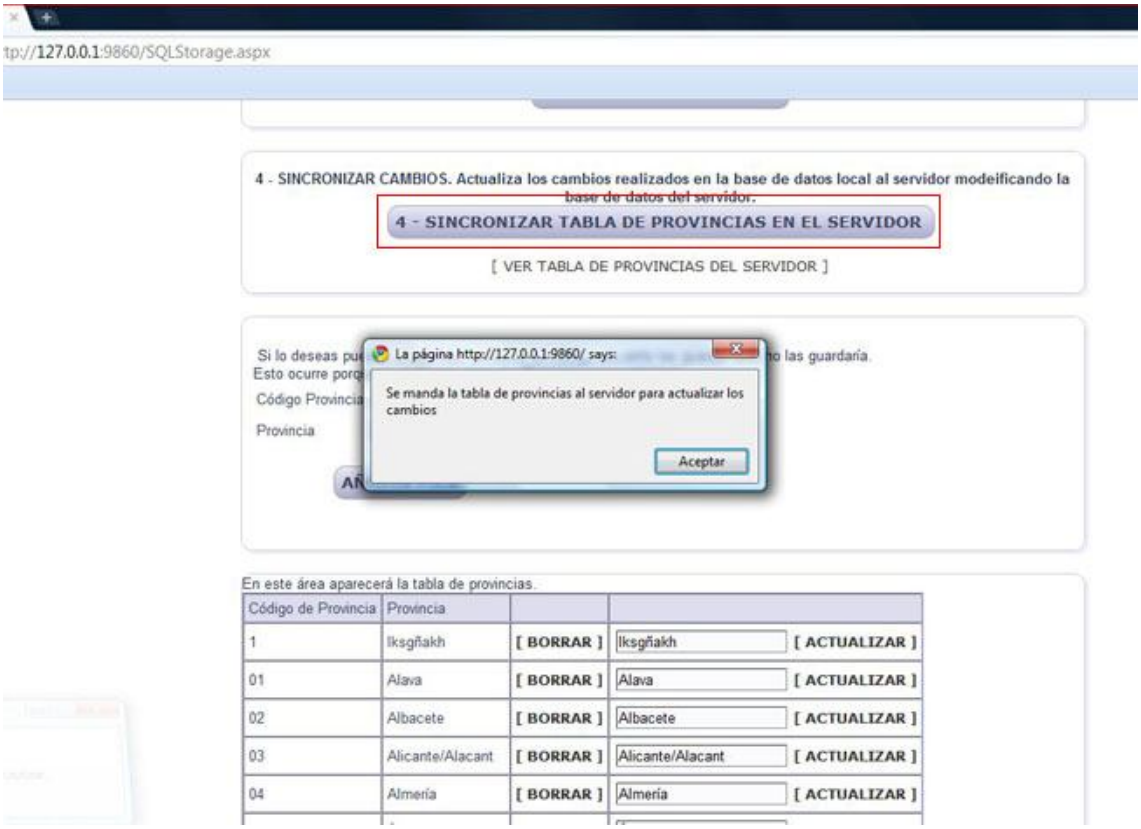


Figura 17 Sincronización de BBDD Local con BBDD del Servidor

CAPÍTULO 9

TECNOLOGÍAS PROBLEMÁTICAS

9. TECNOLOGÍAS PROBLEMÁTICAS QUE SE DESCARTARON PARA LOS EJEMPLOS

Como en todo proyecto de investigación a siempre hay apartados que se consiguen desarrollar con mayor o menor éxito. En el caso de SQL Storage y los Web workers la funcionalidad era mayor a la esperada pero también ha habido 2 ejemplos, en principios sencillos, que no se han visto viables para poder ser desarrollados. La tecnología no estaba madura y se podría considerar que están un poco en fase Beta.

9.1. WEB SOCKETS

En principio se parecen mucho a los sockets de C y permitían establecer comunicaciones bidireccionales. El problema vino cuando se detectó que utilizan un protocolo específico, web sockets protocol (ws), del que no se ha localizado soporte por ningún servidor web comercial. Hay módulos para apache, Google tiene un servidor basado en python, e implementaciones totalmente experimentales en desarrollo. Por lo tanto se descartó al no poder probar la parte servidora.

9.2. EJECUCIÓN OFFLINE (Trabajar sin conexión)

Permite establecer un documento (manifest) donde se indican todos los ficheros necesarios para ejecutar la página sin conexión a Internet. Serían como aplicaciones de escritorio que no necesitarían conexión a Internet nada más que para obtener datos para trabajar. En la práctica no conseguí que almacenara ficheros grandes y los objetos que permiten refrescar la caché o moverse entre las distintas versiones de la misma me dieron error 11 en ejecución, en la especificación indica que se refieren a errores internos en la implementación. Los navegadores empleados en las pruebas fueron Firefox 4.6 y Chrome 5.0.

CAPÍTULO 10

IMPLANTACIÓN

10. IMPLANTACIÓN

Este proyecto no está enfocado a ser implantado en un entorno de producción real. Se espera que solo exista un par de usuarios simultáneos en la base de datos y el servidor Web, por tanto la escalabilidad no es un problema. Tampoco lo es la disponibilidad ni la seguridad, la aplicación no tiene control de acceso y existe un único perfil de usuario.

Sin embargo creo interesante mencionar como sería la arquitectura de procesos y usuarios que intervendrían si se desplegara el proyecto en un entorno de producción real.

Servidor Web IIS 5.0 o Superior

Se desplegaría en un IIS 5.0+ con soporte a ASP.NET 2.0. Habría que controlar los permisos de escritura y exploración en los directorios. El proyecto se desplegaría compilado en un directorio virtual y se le asignaría un pool de recursos de memoria y un porcentaje máximo de uso de CPU.

Los procesos del IIS se ejecutan normalmente bajo el usuario IUSR_[Nombre de la máquina] estando limitado lo que pueden realizar a los permisos de este usuario.

El código del .NET Framework se ejecuta mediante el usuario ASPNET y tiene controlados los recursos y ficheros a los que puede acceder. Esto es importante controlarlo porque si se intenta acceder a memoria o a ficheros no autorizados daría excepciones de seguridad en tiempo de ejecución.

Base de Datos – SQL Server 2005 o Superior

La base de datos no sería una edición Express sino que tendría que ser una edición Server correctamente instalada y configurada. La política de Microsoft es exigir que el sistema operativo sobre el que se instale un SQL Server en producción sea, al menos, la edición Server del sistema operativo y descarta sistemas operativos como Windows XP para estas instalaciones.

La conexión a la Base de Datos se puede realizar con usuarios del sistema operativo o autorizar al usuario ASPNET a acceder a la misma. Sin embargo lo más recomendable es crearse usuarios propios de la base de datos y crear usuarios diferenciados de lectura, escritura y administración. Este último nunca deberá utilizarse desde la aplicación Web.

CAPÍTULO 11

PLANIFICACIÓN DEL PROYECTO

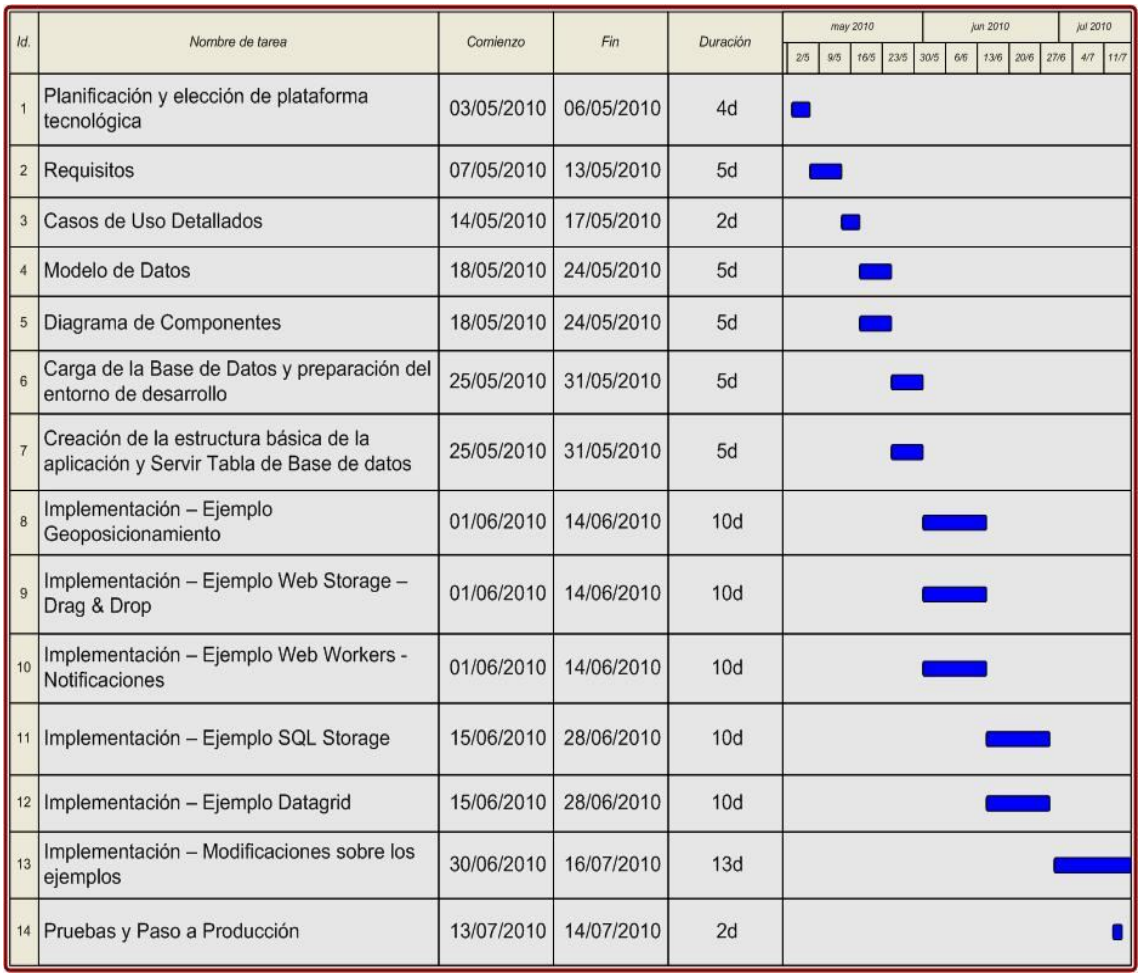
11. PLANIFICACIÓN DEL PROYECTO

La metodología de desarrollo a empleada está basada en una metodología ágil (Extreme Programming) con ciclos de desarrollo cortos e incrementales en los que se va mejorando la funcionalidad paulatinamente. Es muy habitual desarrollar de esta forma aplicaciones Web. La razón es que las páginas pueden actuar muchas veces de forma independiente, es normal solicitar resultados rápidamente y los requisitos suelen variar de forma significativa durante el desarrollo, los proyectos web se quedan obsoletos rápidamente y por tanto se busca una metodología que consuma el menor número de recursos.

La facilidad de adaptarse bien a los cambios en los requisitos es especialmente crítica en este proyecto en concreto. Los ejemplos son más bien pruebas de concepto de unos estándares que todavía no están cerrados y que están implementados de forma desigual en cada navegador, de hecho pueden sufrir cambios mientras se desarrolla el ejemplo. Por tanto es previsible que algunos no se puedan desarrollar del todo y otros solo funcionen en un conjunto reducido de navegadores.

Como el objetivo no es hacer un sistema comercial se han simplificado bastante los tiempos de desarrollo pero las desviaciones son más importantes al ser tecnologías con las que no se tiene experiencia y sobre las que hay poca documentación.

La planificación seguida, a modo orientativo, es la siguiente:



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Planificación y elección de plataforma tecnológica	03/05/2010	06/05/2010	4d
2	Requisitos	07/05/2010	13/05/2010	5d
3	Casos de Uso Detallados	14/05/2010	17/05/2010	2d
4	Modelo de Datos	18/05/2010	24/05/2010	5d
5	Diagrama de Componentes	18/05/2010	24/05/2010	5d
6	Carga de la Base de Datos y preparación del entorno de desarrollo	25/05/2010	31/05/2010	5d
7	Creación de la estructura básica de la aplicación y Servir Tabla de Base de datos	25/05/2010	31/05/2010	5d
8	Implementación – Ejemplo Geoposicionamiento	01/06/2010	14/06/2010	10d
9	Implementación – Ejemplo Web Storage – Drag & Drop	01/06/2010	14/06/2010	10d
10	Implementación – Ejemplo Web Workers - Notificaciones	01/06/2010	14/06/2010	10d
11	Implementación – Ejemplo SQL Storage	15/06/2010	28/06/2010	10d
12	Implementación – Ejemplo Datagrid	15/06/2010	28/06/2010	10d
13	Implementación – Modificaciones sobre los ejemplos	30/06/2010	16/07/2010	13d
14	Pruebas y Paso a Producción	13/07/2010	14/07/2010	2d

El proyecto se planifica siguiendo las típicas fases de un ciclo de vida software. Sin embargo hay que tener en cuenta que todas las fases son iterativas y no implica tener una fase totalmente cerrada para continuar con la siguiente, limitación de los modelos de desarrollo en cascada. De hecho según se fueron desarrollando los ejemplos se fueron incorporando funcionalidades que se vieron interesantes para comprender bien las novedades que incorpora HTML5.

A continuación se detalla un poco más cada fase:

- ***Planificación y elección de la plataforma:*** Se determinó el tiempo necesario y el número de ejemplos a desarrollar. Se eligió la plataforma .NET, entre otras razones por la facilidad para hacer desarrollos rápidos.

- ***Requisitos (análisis):*** Se corresponde con la fase de análisis a alto nivel y recoge los requisitos del portal a desarrollar y de cada ejemplo.

- ***Casos de Uso Detallados (análisis):*** Se detalla mediante un diagrama UML y el posterior detalle de los casos de uso como se comportará el sistema frente a peticiones de los usuarios.

- ***Modelo de Datos (Diseño):*** Se determina la estructura de tablas necesaria, sus relaciones y se implementa en el motor de base de datos relacional elegido, SQL Server Express 2005.

- ***Diagrama de Componentes (Diseño):*** Se descompone la aplicación en las páginas de los ejemplos, componentes de acceso a datos, componentes de lógica de negocio … Luego se refleja cómo interactúan, como se relacionan entre ellos y que interdependencias tienen.

- ***Carga de la Base de Datos:*** Es una tarea previa a comenzar el desarrollo. Se cargan los datos a partir de ficheros Excel descargados de la página del Ministerio de Economía y Hacienda. Permite tener un

conjunto de datos amplio con los que jugarán los ejemplos. En esta tarea también se incluye la preparación del entorno de desarrollo:

instalar el Visual Studio, configurar el SQL Server Express y tareas de soporte que no impliquen desarrollo.

- ***Creación de la estructura básica de la aplicación y servir tablas (Implementación):*** Con esta tarea comienza la codificación de la aplicación. Se desarrolla el sitio Web, los distintos ficheros que lo componen y las clases comunes a todos los ejemplos.

- ***Desarrollo de los Ejemplos (Implementación):*** Se va desarrollando cada ejemplo como una aplicación independiente, en realidad todas estas tareas se podrían hacer en paralelo por distintas personas.

- ***Modificaciones sobre los ejemplos (Implementación):*** Hay que planificar una tarea para pruebas y modificaciones sobre los ejemplos.

- ***Pruebas y Paso a producción (Pruebas e Implantación):*** En este caso el paso es directo porque son ejemplos que no tienen porque ser escalables y no son aplicaciones reales. Se muestran desde el entorno de desarrollo. En la práctica este paso es muy importante y requiere una planificación concienzuda. Junto con el mantenimiento acaba consumiendo gran parte del ciclo de vida de cualquier sistema de información.

CAPÍTULO 12

VALORACIÓN ECONÓMICA

12. VALORACIÓN ECONÓMICA

Para completar la planificación voy a dar una valoración aproximada de lo que costaría realizar un proyecto similar con los perfiles adecuados. Teniendo en cuenta una fase de implementación real que considere el implantar un entorno de producción con las versiones Server de los productos y con consideraciones de seguridad y escalabilidad.

Las jornadas se valoran como 3 horas laborables porque se asume dedicación parcial al proyecto.

Perfil	Jornadas	Horas	Coste / Hora	Total (Euros)
Analista / jefe de proyecto .NET	21	63	45	2.825
Programador Senior .NET	68	204	30	6.120
Técnico de Sistemas Microsoft	5	15	35	525
Técnico de Base de Datos SQL Server	10	30	35	1.050
TOTAL	**104**	**312**		**11.520**

También hay que tener en cuenta que al ser un proyecto de investigación es muy difícil valorar cuantas horas se necesitan y el programador y el analista necesitarán tiempo de auto formación para poder realizar el proyecto.

También habría que considerar el coste de los equipos necesarios para desarrollar y pasar a producción. Se pueden tomar como referencia 2 equipos de gama media-baja de marca HP.

(Fecha de los precios – Julio 2010)

Servidor - HP ProLiant ML150 G6 E5504 2,0GHz Xeon Quad Core
(250 GB de Disco, 2GB de Memoria)

http://h10010.www1.hp.com/wwpc/es/es/sm/WF06b/15351-15351-241434-3328424-3328424-3884323-3918837.html
1.205 + IVA

Equipo de Desarrollo – HPZ200 Pentium i5 3,5 GHz
(160 GB de disco, 4GB de Memoria)

http://h10010.www1.hp.com/wwpc/es/es/sm/WF06a/12454-12454-296719-4050763-4050763-4100568.html
1.259 + IVA

Otra alternativa es pasar la aplicación a producción contratando los servicios de una empresa de hosting / housing, con un servidor dedicado o compartido. Esto permite externalizar parte de los problemas de mantenimiento y asegurar un cierto ancho de banda y disponibilidad. Los precios son bastante asequibles y se mueven en un rango de 40 – 100 euros al mes para equipos compartidos.

CAPÍTULO 13

CONCLUSIONES

13. CONCLUSIONES

HTML Versión 5 y CSS Versión 3 son dos tecnologías con un conjunto de estándares interrelacionados propuestos por el organismo W3C, se pueden contar más de 50 documentos actualmente en fase de estudio. Los implementan los distintos motores de navegación (WebKit, Trident, Gecko …) y sobre estos motores se apoyan los navegadores comerciales. Los estándares no están actualmente cerrados y se van ampliando con las opiniones de los fabricantes involucrados.

Hay ciertos temas muy polémicos, por ejemplo la estandarización de los codecs de video y audio, en los que W3C mantiene la postura de no imponer ningún estándar obligatorio sino que hace recomendaciones. Por tanto en la práctica el soporte depende de decisiones comerciales. Hay que tener en cuenta que los productos comerciales que fueran propuestos por W3C obtendrían un beneficio económico claro pero a su vez un rechazo por navegadores open source como son Chrome o Firefox. El controlar un estándar es una manera de controlar parte del negocio que se genera a través de la web.

Todavía quedan varios años antes de que se puedan cerrar todos los estándares relacionados. Actualmente, en Mayo de 2010, se sigen ampliando y muchos están en fase de borrador. La tendencia actual es a incluir cada vez más funcionalidades en los navegadores de manera que dejen de ser clientes simples y puedan analizar datos o crear efectos gráficos complejos. Se van transformando en pequeños sistemas operativos que compilan e interpretan código para generar aplicaciones. En este sentido en el futuro del estándar esta el manejo de dispositivos de entrada / salida con lo cual se les puede ver como una especie de máquinas virtuales con la ventaja de que el código HTML 5 + CSS 3 + JavaScript que se genera se puede ejecutar en todo tipo de plataformas.

Sin embargo el rendimiento que proporcionan es bastante malo, se espera que mejore con el tiempo. Se están sentando las bases para que sea posible distribuir aplicaciones comerciales complejas como páginas HTML que trabajan OffLine, es decir las descargas y funcionan en local como una aplicación más.

HTML 5 puede dar un impulso importante a la web semántica siempre que los desarrolladores y las empresas hagan un esfuerzo adicional en etiquetar el contenido que proporcionan. Por ahora las expectativas no son muy optimistas porque influyen intereses comerciales ya que en muchos casos interesa que el usuario visualice la página y su publicidad.

No he conseguido localizar información acerca de cómo Google u otros indexadores tratarán el contenido HTML 5, esto es importante porque si premian a los desarrolladores que lo usen es posible que el estándar se adopte más rápido.

Finalmente comentar que, aunque la adopción es importante, todavía hay numerosas tecnologías explicadas en este documento no he visto implementadas correctamente en ningún navegador. Por ejemplo el datagrid.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Al ser un estándar en desarrollo y con implementaciones en distintos navegadores comerciales que van mejorando cada día, la fuente de información ha sido prácticamente Internet puesto que no hay libros que traten con profundidad el tema. En cualquier caso de existir esos libros se quedarían desactualizados puesto que lo que propone W3C y lo que implementan los navegadores va variando cada mes a lo sumo.

Grupo de Trabajo de estandarización de CSS
http://www.w3.org/Style/CSS/current-work#table

W3C – CSS3 Selectors
http://dev.w3.org/csswg/selectors3/

CSS3 Units & Values
http://dev.w3.org/csswg/css3-values/

CSS3 Multi Columns and grids
http://dev.w3.org/csswg/css3-multicol/
http://dev.w3.org/csswg/css3-grid/

CSS3 – Transiciones, Animaciones y transformaciones
http://dev.w3.org/csswg/css3-transitions/
http://dev.w3.org/csswg/css3-animations/
http://dev.w3.org/csswg/css3-2d-transforms/

CSS3 – Speech
http://www.w3.org/TR/2004/WD-css3-speech-20041216/

CSS3 – Media Queries

http://www.w3.org/TR/2009/CR-css3-mediaqueries-20090915/

CSS3 – Text

http://dev.w3.org/csswg/css3-text/

CSS3 –Background & Borders

http://dev.w3.org/csswg/css3-background/

Grupo de Trabajo de estandarización de HTML 5

http://dev.w3.org/html5/spec/Overview.html

Estándar Webs del W3C

http://www.w3.org/standards/webdesign/

HTML 5 - Geolocation API

http://dev.w3.org/geo/api/

HTML 5 – Web Sockets API

http://dev.w3.org/html5/websockets/

HTML 5 – Web Workers

http://dev.w3.org/html5/workers/

W3C –File Writers

http://www.w3.org/TR/2010/WD-file-writer-api-20100406/

W3C –Web Database
http://dev.w3.org/html5/webdatabase/

W3C – System Information API (acceso al hardware)
http://www.w3.org/TR/2010/WD-system-info-api-20100202/

Wikipedia (Inglés), como punto de partida para ir a páginas de referencia en cada tecnología:

- Referencia de términos
- Introducción a las etiquetas y tecnologías
- Referencia de soporte en navegadores

http://en.wikipedia.org/

Referencia de implementación de HTML5 y CSS3 en los navegadores actuales. La actualizan frecuentemente.
http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5)
http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(CSS)

Artículo: Nuevos Elementos en HTML 5
http://www.ibm.com/developerworks/library/x-html5/

Formularios en HTML 5
http://net.tutsplus.com/tutorials/html-css-techniques/rethinking-forms-in-html5/

Implementaciones de HTML 5 en los navegadores

http://wiki.whatwg.org/wiki/Implementations_in_Web_browsers

http://a.deveria.com/caniuse/

Canvas 3D

http://my.opera.com/timjoh/blog/2007/11/13/taking-the-canvas-to-another-dimension

MathML

http://www.mozilla.org/projects/mathml/demo/basics.xhtml

Ejemplos SMIL

http://www.w3schools.com/smil/smil_examples.asp

ANEXO I:

GLOSARIO DE TÉRMINOS

ANEXO I: GLOSARIO DE TÉRMINOS

- **AJAX:** Recarga asíncrona de páginas Web. Solo se recarga una parte de la página evitando recargar la página entera.

- **API:** Aplication Programming Interface. Interfaz de Programación disponible.

- **DOM:** Document Object Model. Modelo de objetos para poder modificar sus propiedades, en este documento se refiere a javascript.

- **CSS3:** Cascading Style Sheets. Hojas de Estilo que se aplican en cascada.

- **ECMAScript:** Lenguaje de scripting estandarizado propuesto por W3C. En realidad se le conoce normalmente por javascript, en la práctica ambos lenguajes se emplean indistintamente. En el documento nos referiremos a JavaScript.

- **HTML5:** Hypertext Markup Lenguage. Lenguaje de Marcas de Hipertexto. Se refiere a conceptos y contenidos enlazados entre sí.

- **SSML.** Speech Synthesis Markup Language. Para reproducir conversaciones en audio.

- **SVG:** Lenguaje de marcas para gráficos vectoriales. Permite definir gráficos vectoriales con etiquetas XML. Se puede incrustar en código HTML en determinados navegadores.

- **XHTML:** Especificación de HTML que cumple además los estándares y la estructura de un documentos bien formado en XML.

ANEXO II:

REFERENCIA DE HTML VERSIÓN 5

ANEXO II: REFERENCIA HTML VERSIÓN 5

Referencia oficial de W3C (Mayo 2010)

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
a	Hyperlink	globals; href; target rel; media; hreflang type	HTMLAnchorElement
abbr	Abbreviation	globals	HTMLElement
address	Contact information for a page or section	globals	HTMLElement
area	Hyperlink or dead area on an image map	globals; alt; coords shape; href; target; rel; media; hreflang type	HTMLAreaElement
article	Self-contained syndicatable or reusable composition	globals	HTMLElement
aside	Sidebar for tangentially related content	globals	HTMLElement
audio	Audio player	globals; src; preload autoplay; loop;	HTMLAudioElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
		controls	
b	Keywords	globals	HTMLElement
base	Base URL and default target browsing context for hyperlinks and forms	globals; href; target	HTMLBaseElement
bdo	Text directionality formatting	globals	HTMLElement
blockquote	A section quoted from another source	globals; cite	HTMLQuoteElement
body	Document body	globals; onafterprint; onbeforeprint; onbeforeunload; onblur; onerror; onfocus; onhashchange; onload; onmessage onoffline; ononline onpagehide; onpageshow;	HTMLBodyElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
		onpopstate; onredo; onresize; onstorage; onundo; onunload	
br	Line break, e.g. in poem or postal address	globals	HTMLBRElement
button	Button control	globals; autofocus; disabled; form; formaction; formenctype; formmethod; formnovalidate; formtarget; name; type; value	HTMLButtonElement
canvas	Scriptable bitmap canvas	globals; width; height	HTMLCanvasElement
caption	Table caption	globals	HTMLTableCaptionElement
cite	Title of a work	globals	HTMLElement
code	Computer code	globals	HTMLElement
col	Table column	globals; span	HTMLTableColElement
colgroup	Group of columns in a table	globals; span	HTMLTableColElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
command	Menu command	globals; type; label; icon; disabled; checked; radiogrou	HTMLCommandElement
datalist	Container for options for combo box contr	globals	HTMLDataListElement
dd	Content for corresponding dt element(s)	globals	HTMLElement
del	A removal from the document	globals; cite; datetime	HTMLModElement
details	Disclosure control for hiding details	globals; open	HTMLDetailsElement
dfn	Defining instance	globals	HTMLElement
div	Generic flow container	globals	HTMLDivElement
dl	Association list consisting of zero or more name-value groups	globals	HTMLDListElement
dt	Legend for corresponding dd element(s)	globals	HTMLElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
em	Stress emphasis	globals	HTMLElement
embed	Plugin	globals; src; type; width; height; any*	HTMLEmbedElement
fieldset	Group of form controls	globals; disabled; form; name	HTMLFieldSetElement
figcaption	Caption for figure	globals	HTMLElement
figure	Figure with optional caption	globals	HTMLElement
footer	Footer for a page or section	globals	HTMLElement
form	User-submittable form	globals; accept-charset; action; autocomplete; enctype; method; name; novalidate; target	HTMLFormElement
h1, **h2**, **h3**, **h4**, **h5**, **h6**	Section heading	globals	HTMLHeadingElement
head	Container for document metadata	globals	HTMLHeadElement
header	Introductory or	globals	HTMLElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
	navigational aids for a page or section		
hgroup	heading group	globals	HTMLElement
hr	Thematic break	globals	HTMLHRElement
html	Root element	globals; manifest	HTMLHtmlElement
i	Alternate voice	globals	HTMLElement
iframe	Nested browsing context	globals; src; srcdoc name; sandbox; seamless; width; height	HTMLIFrameElement
img	Image	globals; alt; src; usemap; ismap; width; height	HTMLImageElement
input	Form control	globals; accept; alt; autocomplete; autofocus; checked disabled; form; formaction; formenctype; formmethod; formnovalidate; formtarget; height; list; max; maxlengt	HTMLInputElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
		min; multiple; nam pattern; placeholde readonly; required size; src; step; type value; width	
ins	An addition to the document	globals; cite; datetime	HTMLModElement
kbd	User input	globals	HTMLElement
keygen	Cryptographic key-pair generator form control	globals; autofocus; challenge; disabled form; keytype; nam	HTMLKeygenElement
label	Caption for a form control	globals; form; for	HTMLLabelElement
legend	Caption for fieldset	globals	HTMLLegendElement
li	List item	globals; value*	HTMLLIElement
link	Link metadata	globals; href; rel; media; hreflang; type; sizes	HTMLLinkElement
map	Image map	globals; name	HTMLMapElement
mark	Highlight	globals	HTMLElement
menu	Menu of	globals; type; label	HTMLMenuElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
	commands		
meta	Text metadata	globals; name; http equiv; content; charset	HTMLMetaElement
meter	Gauge	globals; value; min; max; low; high; optimum; form	HTMLMeterElement
nav	Section with navigational links	globals	HTMLElement
noscript	Fallback content for script	globals	HTMLElement
object	Image, nested browsing context or plugin	globals; data; type; name; usemap; forr width; height	HTMLObjectElement
ol	Ordered list	globals; reversed; start	HTMLOListElement
optgroup	Group of options a list box	globals; disabled; label	HTMLOptGroupElement
option	Option in a list bo or combo box control	globals; disabled; label; selected; valu	HTMLOptionElement
output	Calculated output value	globals; for; form; name	HTMLOutputElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
p	Paragraph	globals	HTMLParagraphElement
param	Parameter for object	globals; name; value	HTMLParamElement
pre	Block of preformatted text	globals	HTMLPreElement
progress	Progress bar	globals; value; max form	HTMLProgressElement
q	Quotation	globals; cite	HTMLQuoteElement
rp	Parenthesis for ruby annotation text	globals	HTMLElement
rt	Ruby annotation text	globals	HTMLElement
ruby	Ruby annotation (s)	globals	HTMLElement
samp	Computer output	globals	HTMLElement
script	Embedded script	globals; src; async; defer; type; charset	HTMLScriptElement
section	Generic document or application section	globals	HTMLElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
select	List box control	globals; autofocus; disabled; form; multiple; name; siz	HTMLSelectElement
small	Side comment	globals	HTMLElement
source	Media source for video or audio	globals; src; type; media	HTMLSourceElement
span	Generic phrasing container	globals	HTMLSpanElement
strong	Importance	globals	HTMLElement
style	Embedded styling information	globals; media; typ scoped	HTMLStyleElement
sub	Subscript	globals	HTMLElement
summary	Caption for details	globals	HTMLElement
sup	Superscript	globals	HTMLElement
table	Table	globals; summary	HTMLTableElement
tbody	Group of rows in table	globals	HTMLTableSectionElement
td	Table cell	globals; colspan; rowspan; headers	HTMLTableDataCellElement
textarea	Multiline text field	globals; autofocus; cols; disabled; form	HTMLTextAreaElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
		maxlength; name; placeholder; readonly; required rows; wrap	
tfoot	Group of footer rows in a table	globals	HTMLTableSectionElement
th	Table header cell	globals; colspan; rowspan; headers; scope	HTMLTableHeaderCellElement
thead	Group of heading rows in a table	globals	HTMLTableSectionElement
time	Date and/or time	globals; datetime; pubdate	HTMLTimeElement
title	Document title	globals	HTMLTitleElement
tr	Table row	globals	HTMLTableRowElement
ul	List	globals	HTMLUListElement
var	Variable	globals	HTMLElement
video	Video player	globals; src; poster; preload; autoplay; loop; controls; width; height	HTMLVideoElement
wbr	Line breaking	globals	HTMLElement

Element	Description	Attributes & Events	Interface ECMAScript / Javascript
	opportunity		

NUEVOS EVENTOS

Aunque se han visto la mayoría en apartados anteriores es interesante destacar los siguientes eventos que tienen mucho que ver con las nuevas funcionalidades:

- **onbeforeprint, onafterprint** → Para preparar el documento para imprimir.
- **onplay, oncanplay, oncanplaythrough, ondurationchange, onvolumecahnge** → para reproducir música o video.
- **oncontexmenu** → para obtener un menú contextual al pulsar el segundo botón del ratón.
- **ondrag *, ondrog** → Para tratar el Drag & Drop de elementos.
- **ononline, onoffline** → Para manejar páginas y aplicaciones desconectadas.
- **onstorage** → para almacenar objetos de manera persistente.

Referencia de HTML 5 (Archivo PDF independiente)

Archivo Externo - Html-Cheat-Sheet.Pdf (130 KB)

ANEXO III:

REFERENCIA DE CSS VERSIÓN 3

ANEXO III: REFERENCIA DE CSS VERSIÓN 3

En Internet hay varios resúmenes en PDF interesantes, he incorporado a este documento la referencia que considero el más interesante.

Archivo Externo - Css-Cheat-Sheet.pdf (122KB)

ANEXO IV:

DESCARGA DE NAVEGADORES

ANEXO IV: DESCARGA DE NAVEGADORES

Google Chrome

Tiene un instalador Web que descarga OnLine la última versión.

http://www.google.es/chrome

Mozilla Firefox

http://www.mozilla-europe.org/es/firefox/

Opera

http://www.opera.com/download/

Apple Safari

Es el navegador de los Macs y del iPhone.

http://www.apple.com/es/safari/download/

Internet Explorer

Es el que más cuota de mercado tiene pero menor soporte a HTML 5 en su versión 8, la Beta de la versión 9 se espera que le amplio soporte.

IE v8

http://www.microsoft.com/downloads/details.aspx?FamilyID=341c2ad5-8c3d-4347-8c03-08cdecd8852b&displayLang=en

IE v9 (Beta)

http://ie.microsoft.com/testdrive/