2- INTRODUCCIÓN TEÓRICA

2.1- SITUACIÓN DE PARTIDA

2.1.1- Introducción

Estamos en una época donde la información ha dejado de ser un privilegio, y el trabajo no se centra en la búsqueda si no en la selección de información por lo que se hace necesario contar con los conocimientos básicos de computación aplicados a la medicina.

El SAS (Servicio Andaluz de Salud), como organismo sanitario público andaluz, está desarrollando políticas de salud que sitúan al ciudadano como eje fundamental del sistema sanitario de Andalucía.

Uno de los proyectos en los que trabaja el SAS es Diraya, cuya finalidad es obtener e implantar, mediante tecnologías informáticas, una Historia de Salud única para cada ciudadano en el ámbito de Andalucía. Inicialmente su ámbito es la Atención Primaria ya que, en gran medida, viene a sustituir las herramientas aportadas por el proyecto TASS, aprovechando la experiencia obtenida con el mismo. Se trata de dar un paso adelante dotando de integridad a los sistemas.

Diraya es un proyecto integrado por diversos servicios, entre los que podemos destacar:

- El empleo de la tarjeta sanitaria individual como sistema de identificación y acceso al sistema sanitario.
- La creación de la Base de Datos de Usuarios (BDU) para todos los ciudadanos andaluces.
- La puesta en marcha de una Historia digital de salud única o la oficina virtual Inters@s, que facilita que los ciudadanos realicen distintas gestiones sanitarias a través de internet.

La BDU se terminó en octubre de 2001 y contiene información básica de todos los andaluces, accesible en la red con el número de la tarjeta individual sanitaria.

Otro programa en el que el SAS está trabajando es Receta XXI, que también está integrado en el proyecto Diraya y afectará a los farmacéuticos de forma especial.

De toda la información sanitaria que puede existir de un paciente dentro de su historial médico nosotros centraremos nuestro proyecto en el Informe de alta.

2.1.2- Informe de alta

Es la parte del historial médico de un paciente que se realiza al finalizar el proceso asistencial de un paciente en el hospital.

La información contenida en el Informe de Alta debe ser, a grandes rasgos:

- *Datos personales del paciente*, a efectos de verificación de su identidad, lugar de residencia, formas de contacto y otros datos de interés.
- *Antecedentes médicos* de importancia tales como alergias, adicciones, patologías e ingresos previos, historia familiar, etc...
- *Datos de admisión* concretos como el nombre y tipo de hospital en el que ingresa el paciente, el agente sanitario remitente del informe o el encargado de los cuidados.
- *Circunstancias al ingreso* del paciente en el hospital: fecha, motivo, modo de ingreso...
- *Exámenes físicos* realizados al paciente durante su estancia en el hospital.

- *Juicio clínico* dictaminado por los profesionales sanitarios encargados de los cuidados.
- *Procedimientos realizados* a lo largo del período de convalecencia del paciente.
- *Tratamiento médico*: medicación prescrita por los médicos y forma adecuada de administración.
- *Circunstancias al alta*: fecha, tipo, y lugar al que se remite al paciente en el momento de recibir el alta médica.

2.2- JAVA

2.2.1- JAVA

Java es un lenguaje originalmente desarrollado por un grupo de ingenieros de Sun, utilizado por Netscape posteriormente como base para Javascript. Si bien su uso se destaca en el Web, sirve para crear todo tipo de aplicaciones (locales, intranet o internet).

Java es un lenguaje de objetos independiente de la plataforma.

Un objeto puede verse como una pieza de software que cumple con ciertas características como encapsulamiento y herencia.

Java implementa mecanismos de seguridad que limitan el acceso a recursos de las máquinas donde se ejecuta, especialmente en el caso de los Applets (que son aplicaciones que se cargan desde un servidor y se ejecutan en el cliente).

También está diseñado específicamente para trabajar en una red, de modo que incorpora objetos que permiten acceder a archivos en forma remota (via URL por ejemplo).

2.2.2- Caracteristicas de los servlets

La tecnología Servlet proporciona las mismas ventajas del lenguaje Java en cuanto a portabilidad (“write once, run anywhere”) y seguridad, ya que un Servlet es una clase de Java igual que cualquier otra, y por tanto tiene en ese sentido todas las características del lenguaje. Por otro lado los Servlets, una vez que son llamados por primera vez, quedan activos en la memoria del servidor hasta que el programa que controla el servidor los desactiva, de esta manera se minimiza en gran medida el tiempo de respuesta.

Además los Servlets se benefician de la gran capacidad de Java para ejecutar métodos en ordenadores remotos, conectar con bases de datos, etc. Se podría decir que las clases estándar de Java ofrecen resueltos muchos de los problemas que con otros lenguajes tiene que resolver el programador.

Además de las características citadas anteriormente, podemos destacar:

- Son independientes del servidor utilizado y de su sistema operativo.
- Los Servlets pueden llamar a otros Servlets, e incluso a métodos concretos de otros Servlets. De esta forma se puede distribuir de forma más eficiente el trabajo a realizar. Por ejemplo, se podría tener un Servlet encargado de la interacción con los clientes y que llamara a otro Servlet para que a su vez se encargara de la comunicación con una base de datos. De igual forma, los Servlets permiten direccionar peticiones de servicios a otros Servlets (en la misma máquina o en una máquina remota).
- Los Servlets pueden obtener fácilmente información acerca del cliente (la permitida por el protocolo HTTP), tal como su dirección IP, el puerto que se utiliza en la llamada, el método utilizado (GET, POST,…), etc.
- Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información específica acerca de un usuario determinado, personalizando de esta forma la interacción cliente-servidor. Una clara aplicación es mantener la sesión con un cliente.

- Los Servlets pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor de 3 capas (si la base de datos está en un servidor distinto).

- Asimismo, pueden realizar tareas de proxy para un applet. Debido a las restricciones de seguridad, un applet no puede acceder directamente por ejemplo a un servidor de datos localizado en cualquier máquina remota, pero el Servlet sí puede hacerlo de su parte.

- Al igual que los programas CGI, los Servlets permiten la generación dinámica de código HTML dentro de una página HTML. Así, pueden emplearse Servlets para la creación de contadores, banners, etc.

2.3- XML

2.3.1- Historia de XML

En los años sesenta, se perseguía la idea de estructurar los documentos de forma organizada, con el fin de facilitar el intercambio y la manipulación de los datos. IBM creó GML (Lenguaje de Marcado Generalizado) para satisfacer las necesidades de sus sistemas internos de edición. Empleó GML para producir libros, informes y otros documentos a partir de un solo conjunto de archivos fuente. En empresas específicas se introdujeron otras soluciones de estructuración de información, pero no se hacía nada por solucionar el tema a gran escala.

La primera tecnología de la información estandarizada y estructurada de cierta importancia fue SGML (Lenguaje de Marcado Generalizado Estándar), que también procedía de IBM. SGML se creó para proporcionar una manera de dar formato y mantener documentos legales en IBM. Posteriormente, se expandió y adaptó para ser utilizado en un amplio abanico de sectores como estándar de información de propósito general. No fue hasta 1986 que SGML emergió como estándar ISO. Aunque SGML es extremadamente potente, es muy complejo y se requiere una gran cantidad de software para procesarlo.

Debido a esta complejidad y a su elevado coste, SGML no supuso claramente una alternativa al hipertexto en los primeros días de Internet.

En 1989, Tim Berners-Lee y Anders Berglund, dos investigadores del Laboratorio Europeo de Física de Partículas (CERN), crearon un lenguaje basado en etiquetas para marcar documentos técnicos a fin de compartirlos en Internet. Este lenguaje fue finalmente ampliado en una aplicación simplificada de SGML llamada HTML, que supuso el primer formato de información estándar de la Web.

HTML tuvo mucho éxito y todo el mundo quería estar conectado. Cuando pasó la moda, tanto los particulares como las empresas empezaron a anhelar algo más que imágenes estáticas y texto. Lo que siguió fue una sucesión mareante de tecnologías tendentes a añadir interactividad a la Web. Algunas de estas tecnologías, como Java, han permanecido y florecido, mientras que otras se han quedado en la estacada. Aparte de las tecnologías interactivas, el propio HTML atravesó una rápida serie de mejoras (se añadieron más características, pero empezó a complicarse).

HTML fue originariamente diseñado como forma de presentar información estática sólo para fines de despliegue. Agregar interactividad a HTML, esperando que representara información dinámica, supuso el cambio de su propia naturaleza. Aún más importante que el cambio de información estática a dinámica fue el hecho de que se esperaba que HTML sirviera como forma de almacenar tipos específicos de datos, como transacciones financieras, catálogos de productos y resúmenes. Los desarrolladores web pronto se dieron cuenta de las ventajas de conectar páginas web a bases de datos de servidores, en cuyo caso se utilizaba HTML como interfaz de base de datos. HTML estaba cambiando de lenguaje de información basada en presentaciones a una tecnología de software.

HTML ha llevado a cabo una ímproba tarea satisfaciendo las necesidades de los desarrolladores web, considerando sus limitaciones innatas. Sin embargo, los miembros del Consorcio de World Wide Web (W3C) se dieron cuenta de que sería necesaria una alternativa mucho más amplia que HTML

para satisfacer las necesidades futuras de la Web. Quizás la limitación más patente de HTML sea su conjunto fijo de etiquetas. Es imposible que los desarrolladores web añadan etiquetas personalizadas con HTML, lo cual sería infinitamente más útil al tratar con datos pertenecientes a una aplicación o sector específicos. Por ejemplo, un editor de libros utilizaría mucho más etiquetas como <título>, <autor> e <isbn> que una etiqueta de párrafo <p> genérica.

En 1996, el W3C se propuso introducir el poder y la flexibilidad de SGML en el dominio de la Web. SGML ofrecía tres ventajas importantes que faltaban en HTML: extensibilidad, estructura y validación. Técnicamente, hay estructura en HTML, pero a la mayoría de los navegadores les trae sin cuidado si se adhiere a ella o no. SGML es mucho menos compasivo que los navegadores web a la hora de aventurarse fuera de la estructura permitida. Así, por razones puramente prácticas, HTML es un lenguaje poco estructurado.

El W3C presentó un equipo de expertos en SGML cuyo objetivo era el de crear una nueva tecnología de marcado con las ventajas nucleares de SGML y con la relativa simplicidad de HTML. El equipo “SGML para la Web” trabajó diligente y pacientemente en su nueva tecnología de marcado hasta febrero de 1998, fecha en la que el W3C lanzó la especificación XML 1.0.

2.3.2- XML

XML (eXtensible Markup Language) es un estándar industrial para representar datos, independiente del sistema. Al igual que **HTML** (HyperText Markup Language), XML encierra los datos en etiquetas, pero hay importantes diferencias entre los dos lenguajes de marcas. Primero, las etiquetas XML tienen relación con el significado del texto que encierran, mientras que las etiquetas HTML especifican cómo mostrar el texto encerrado. El siguiente ejemplo XML muestra una lista de precios con el nombre y el precio de dos cafés:

```
<priceList>
        <coffee>
```

```
            <name>Mocha Java</name>
            <price>11.95</price>
        </coffee>
        <coffee>
            <name>Sumatra</name>
            <price>12.50</price>
        </coffee>
</priceList>
```

Las etiquetas <coffee> y </coffee> le dicen al analizador que la información que hay entre ellas trata sobre un café. Las otras dos etiquetas dentro de las etiquetas <coffee> especifican que la información encerrada son el nombre del café y su precio por libra. Como las etiquetas XML especifican el contenido y la estructura de los datos que encierran es posible hacer cosas como archivar o buscar datos.

Una segunda diferencia importante entre XML y HTML es que las etiquetas XML son extensibles, permitiéndonos escribir nuestras propias etiquetas XML para describir nuestro contenido. Con HTML, estábamos limitados a usar sólo aquellas etiquetas que habían sido predefinidas en la especificación HTML.

Con la extensibilidad que proporciona XML, podemos crear las etiquetas que necesitemos para un tipo de documento en particular. Definimos las etiquetas usando un esquema de lenguaje XML. Un esquema describe la estructura de un conjunto de documentos XML y puede usarse para limitar los contenidos de los documentos XML.

Otras características también contribuyen a la popularidad de XML como un método para intercambio de datos. Está escrito en formato de texto, de forma que lo pueden leer tanto los humanos como los softwares de edición de texto. Las aplicaciones pueden analizar y procesar documentos XML, y los humanos también pueden leerlos en caso de que haya algún error en el procesamiento. Otra característica es que como un documento XML no incluye instrucciones de formateo, puede mostrarse de varias formas. Mantener los datos separados de

las instrucciones de formateo significa que los mismos datos pueden publicarse en diferentes medios.

Hay varias reglas básicas en XML. Echemos un vistazo a los objetivos establecidos por el W3C para XML:

- XML será directamente utilizable en Internet.
- XML soportará una amplia variedad de aplicaciones.
- XML será compatible con SGML.
- Será fácil escribir programas que procesan documentos XML.
- El número de características opcionales en XML debe mantenerse en un mínimo absoluto, idealmente cero.
- Los documentos XML deberán ser legibles por humanos y resultar razonablemente claros.
- El diseño de XML deberá ser preparado rápidamente.
- El diseño de XML será formal y conciso.
- Los documentos XML serán fáciles de crear.

XML se utiliza para crear otros lenguajes de marcas. Uno de los más conocidos es XSL. XSL está basado en XML. La relación entre XML y XSL es de suma importancia. Si XML son los datos en bruto, entonces XSL se usa para convertir esos datos en bruto en un formato de datos utilizable.

XML no sólo contiene datos, también puede contener la estructura e incluso la categorización de la información dentro del mismo documento. Con XML el documento lleva documentación acerca de sí mismo.

Microsoft Internet Explorer 4.0, 5.0, 5.5 y el 6.0 dan soporte a XML.

Netscape posee algún soporte para XML en Communicator/Navigator 6.0.

2.3.3- XML Schema

XML se ha consolidado definitivamente como un lenguaje estándar de información, tanto como medio de persistencia como de intercambio de información. En estas aplicaciones es de mucha relevancia qué posibilidades

nos ofrece esta tecnología de cara al modelado y la validación de la información.

En primer lugar es importante insistir en la necesidad, cada vez mayor, de poder estructurar correctamente, y de un modo más intuitivo, la información. Al igual que con la creación de la programación estructurada y posteriormente el almacenamiento relacionado de la información, actualmente está claro que ya no basta con almacenar la información identificando con que otros elementos está relacionada, sino que empieza a ser necesario, al igual que en el paradigma de la orientación a objetos, jerarquizar y estructurar la información.

Para proceder a esta estructuración dentro de un documento XML se han propuesto distintas soluciones dentro del entorno de W3C, todas enfocadas a definir un patrón externo al documento propiamente dicho que nos permita decir si el documento se adhiere a la estructura esperada o no.

Aquí vamos a tratar con dos de las principales soluciones, las **DTDs** y los **XML Schemas**, pero antes de iniciar la exposición de las diferencias entre ambas es importante echar un rápido vistazo a su origen.

Inicialmente el uso de las **DTDs** se basa en SGML, en el cual describían no sólo el vocabulario necesario para identificar todos los elementos de que iba a constar un documento, si no que también expresaba la estructura que dichos elementos debían respetar. Con la creación de XML, que recordemos que no es más que un subconjunto de SGML, fue necesario mantener la posibilidad de describir los elementos necesarios, al igual que en SGML, por ello se importaron las **DTDs** y todo su comportamiento.

Posteriormente, y debido al uso principalmente, se vio la necesidad de emplear otros métodos para describir esas necesidades inherentes a XML, con esta idea se creó XML Schema, se pretendía mejorar y ampliar la utilidad de las **DTDs.**

Como hemos indicado antes, una **DTD** es una especificación y estructuración necesaria que permite validar el contenido estructural y formal de un documento de SGML (y XML). Las **DTDs** se pueden usar para la

definición de modelos de contenido, es decir, en qué orden y qué elementos pertenecen a un elemento de orden superior en la jerarquía del documento; además permiten, aun que de modo muy limitado, imponer ciertas restricciones sobre el tipo de los elementos.

Ejemplo de DTD

```
<!ELEMENT Articulo (Cabecera, Contenido, Final)>
<!ELEMENT Cabecera (Titulo, Autor)>
<!ELEMENT Titulo ANY>
<!ELEMENT Autor ANY>
<!ELEMENT Contenido (Estracto, Cuerpo)>
.....
```

Ejemplo de XML Schema

```
<schema targetNamespace=”http://www.aqs.es/Schema_y_DTDs”
xmlns=”http://www.w3.org/2001/XMLSchema”
xmlns:AQS=”http://www.aqs.es/Schema_y_DTDs” >
<element name=”Articulo” type=”AQS:tArticulo” />
<complexType name=”tArticulo”>
<element name=”Cabecera” type=”AQS:tCabecera”/>
<element name=”Cuerpo” type=”AQS:tCuerpo”/>
<element name=”Final” type=”AQS:tFinal”/>f
        </complezType>
        <complexType name=”tCabecera”>
        <element name=”Titulo” type=”string”/>
        <element name=”Autor” type=”string”/>
        </complexType>
        .....
```

Al igual que las **DTDs**, los **Schemas** describen la estructura de la información. El motivo de la creación de este nuevo estándar para realizar la labor de las **DTDs** es, básicamente, la utilidad. Durante un tiempo, y a falta de

otra solución más ajustada, se emplearon los mecanismos que proporcionaba SGML para modelizar la información en XML. Pero el descubrimiento de nuevas aplicaciones de XML al margen de la estructuración de documentos forzó la creación de otras soluciones que ayudasen a solventar los nuevos problemas a los que se enfrentaba el mercado.

Dado que el surgir de una nueva tecnología donde ya existía otra es síntoma de que un grupo de gente cree que en el actual estado de ese campo o bien las cosas no están bien hechas o bien falta algo por hacer vamos a mostrar, que limitaciones tenían las DTDs para que fuese necesaria la aparición de XML Schema.

- Posee un lenguaje propio de escritura, lo cual deriva en problemas a la hora de:
    - el aprendizaje: no sólo hay que aprender XML si no que además hay que conseguir hacerse con el lenguaje de las **DTDs**,.
    - procesado del documento: las herramientas y parsers que se empleen para tratar los documentos de XML deben ser capaces de procesar las **DTDs** .
- No permite el uso de **namespaces**, estos permiten definir elementos con igual nombre dentro del mismo contexto, siempre y cuando se anteponga un prefijo al nombre del elemento, y lo más importante permiten el uso de elementos definidos en otros documentos.
- Tiene un tipado para los datos del documento extremadamente limitado, no permite definir el que un elemento pueda ser de un tipo numeral o de un tipo de fecha, sólo presenta variaciones limitadas sobre strings.
- El mecanismo de extensión es complejo y frágil, está basado en sustituciones sobre strings. Lo peor de dichas extensiones es que realmente no hace explicitas las relaciones en ningún momento, es decir, dos elementos que tienen definido el mismo modelo de contenido no presentan ninguna relación.

Estás pegas son superadas en la especificación de **XML Schema**, permitiendo un lenguaje mucho más expresivo. A parte de la expresividad, la

presencia de los Schemas permite un intercambio de información mucho más robusto.

A parte de solventar los problemas antes expuestos, **XML Schema**, permite una serie de ventajas adicionales que se consideraron importantes:

- Una estructura de tipos mucho más rica. En **XML Schema** se definen los tipos base que se pueden emplear dentro de esquema de XML, e.g., integer, bolean, string, date, etc.
- Permite tipos definidos por el usuario, llamados **Arquetipos**, dándoles un nombre y que se pueden emplear en distintas partes dentro del **Schema**.
- Es posible agrupar atributos, haciendo más comprensible el uso de un grupo de aspectos de varios elementos distintos, pero con denominador común, que deben ir juntos en cada uno de estos elementos.
- El trabajo con namespaces está especificado, permitiendo, dentro de la dificultad que conlleva trabajar con ellos, validar documentos con varios **namespaces**.
- Sin embargo, la característica que más resalta la utilidad de XML Schema es la posibilidad de extender **Arquetipos** de un modo específico, es decir permite lo que en términos de orientación a objetos se llama **herencia**. Considérese un esquema que extiende otro previamente hecho, se permite refinar la especificación de algún tipo de elemento para, por ejemplo, indicar que puede contener algún nuevo elemento del tipo que sea; pero dejando el resto del esquema antiguo completamente intacto.

Tal como hemos visto, es necesario empezar el *schema* definiendo los elementos más profundamente anidados dentro de la estructura jerárquica de elementos del documento XML. Es decir, tenemos que trabajar "de dentro hacia fuera".

2.3.4- DOM

El Modelo de Objetos de Documento, a diferencia de SAX, tiene su origen en el Consorcio World Wide Web (W3C). Mientras que SAX es un software de dominio público, desarrollado tras largas discusiones en la lista de distribución XML-dev, DOM es un estándar de la misma manera que lo es la especificación real de XML. DOM tampoco está diseñado específicamente para Java, sino para representar el contenido y el modelo de documentos en todos los lenguajes y herramientas de programación.

DOM se organiza en “niveles” en vez de en versiones. El Nivel Uno de DOM es una Recomendación aceptada y puede verse completa en http://www.w3.org/TR/RED-DOM-Level-1. El Nivel Uno detalla la funcionalidad y la navegación del contenido en un documento. Un documento en DOM no se limita a XML, puede ser HTML, o también otros modelos de contenido. El Nivel Dos añade al Nivel Uno módulos u opciones dirigidas a modelos específicos, como SML, HTML y Hojas de Estilo en Cascada (CSS).

La especificación DOM está diseñada para ser utilizada con cualquier lenguaje de programación. Por lo tanto, intenta utilizar un conjunto básico común de características disponibles en todos los lenguajes. La especificación DOM también intenta ser neutra en cuanto a las definiciones de interfaz. Esto permite a los programadores de Java aplicar sus conocimientos de DOM cuando trabajan con Visual Basic o Perl, y viceversa.

La especificación también trata todos los elementos del documento como un nodo, formado por un tipo y un valor. Esto proporciona un marco conceptual elegante para trabajar con todos los aspectos del documento.

El inconveniente de la neutralidad del lenguaje de DOM es que las metodologías y los patrones que se utilizan normalmente en los lenguajes de programación no se pueden emplear. Por ejemplo, en lugar de poder crear nuevos elementos (Element) mediante el conocido elemento new de Java, los programadores deben utilizar los métodos de construcción de fábrica. Los conjuntos de nodos se representan como listas de nodos (NodeList), en lugar de

los objetos normales List o Iterator. Estos pequeños inconvenientes se agregan a las prácticas de codificación poco habituales y al aumento de las líneas de código, y fuerzan al programador a aprender la metodología DOM en lugar de utilizar la forma más intuitiva.

DOM emplea una abstracción del tipo "todo integrado en un nodo". Esto significa que casi todos los elementos del documento XML, como por ejemplo Document, Element y Attr, heredan (extend) la interfaz Node. Este método no es sólo conceptualmente elegante, sino que permite que cada implementación distinta de DOM ponga al descubierto sus propias clases mediante las interfaces estándar, sin la pérdida de rendimiento que supone atravesar las clases de acomodación intermedias.

La abstracción "todo integrado en un nodo" pierde cierto valor debido al número de tipos de nodo existente y a la ausencia de uniformidad presente en los métodos de acceso. Por ejemplo, el método insertData se utiliza para establecer el valor de los nodos CharacterData, mientras que el valor de los nodos Attr (atributo) se establece mediante el método setValue. Mediante la presentación de diferentes interfaces para los distintos nodos, la uniformidad y la elegancia del modelo disminuyen y la curva de aprendizaje aumenta.

El DOM define varias interfaces, algunas de las cuales son más útiles que las otras. Cada interfaz posee un conjunto de métodos y atributos:

- **DOMImplementation**. Es una consulta al propio objeto y es independiente del documento que se le cargue. Devuelve información acerca del nivel del DOM que soporta el objeto.
- **DocumentFragment**. Es la parte ligera del documento.
- **Document**. Esta interfaz proporciona información acerca del propio documento principal que se ha cargado en el objeto. Posee varios métodos para crear nodos y atributos XML.
- **Node**. Todo en un documento puede ser considerado un nodo. Los elementos son nodos, los comentarios son nodos, y así sucesivamente. Esta interfaz contiene varios atributos generales apropiados para manipular cualquier tipo de nodo. De forma poco sorprendente, los

métodos y atributos del nodo se solapan con los de las demás interfaces. Por ejemplo, al tratar con un elemento, el atributo del nodo nodeName devolverá el mismo valor que el atributo del elemento tagName.

- **NodeList**. Es un conjunto ordenado de nodos, al que se puede acceder por medio de un índice.
- **NamedNodeMap.** Es una colección de nodos a los que se puede acceder por nombre. Los atributos de un elemento se almacenan como NamedNodeMap.
- **CharacterData.** Es una interfaz que trata sobre los datos de caracteres o con el texto de un documento.
- **Attr**. Esta interfaz trata sobre los atributos XML de un nodo. La versión abreviada Attr se usa para evitar la confusión con un atributo.
- **Element.** La mayoría de los nodos de un documento XML son elementos. Esta interfaz posee propiedades y métodos que sirven para tratar con los elementos y sus atributos XML.
- **Text.** Trata sobre el contenido de texto de un elemento. La mayor parte de la funcionalidad que requiere la maneja la interfaz CharacterData.

Vamos a examinar algunas de las interfaces examinando los atributos y métodos específicos, la interfaz Node, que es la interfaz clave del DOM. Todo en un documento puede considerarse un nodo, incluyendo el propio documento.

Atributos de sólo lectura de la interfaz de Node

Atributos	**Descripción**
nodeName	Proporcionan los nodos nodeName y nodeValue de todos los distintos tipos de nodos
nodeValue	
nodeType	Existen 12 tipos de nodos que se describen en el DOM del W3C. Entre ellos el que devuelve el nombre del tag, el que da el atributo, el que da el texto.
parentNode	Todos los nodos, a excepción de Documento, documentFragment, entidad, notación y Attr poseen un primario.
childNodes	Devuelve una NodeList de todos los nodos secundarios del nodo.
firstChild	Los nombres de estos atributos describen su valor de retomo. Éste es el objeto de nodoque contiene el relativo respectivo. Si la relación no existe, se devolverá null.
lastChild	
previousSibling	
nextSibling	
attributes	Devuelve un NamedNodeMap de todos los atributos
ownerDocument	Este atributo devuelve el objeto de documento como un nodo.

Métodos de la interfaz Node

Métodos	**Descripción**
insertBefore	Este método inserta el nuevo nodo secundario antes del nodo del secundario de referencia y devuelve new_node.
replaceChild	Este método sustituye al nodo secundario de referencia y devuelve el nodo sustituido.
removeChild	Este método elimina el nodo secundario de referencia y devuelve el nodo eliminado
appendChild	Este método adjunta el nuevo nodo al final de los demás nodos secundarios y devuelve el nuevo nodo
hasChildNodes	Este método es un valor booleano que comprueba si un nodo determinado tiene nodos secundarios.
cloneNode	El método cloneNodeO hará una copia del nodo donado. Si se clona un elemento, tomará un argumento true o false, que hará referencia a la profundidad del clónico.

La cuestión es cómo se incorpora un motor DOM a la aplicación:

- **Java**. Los programadores de Java son los que más fácil lo tienen. Registre el motor DOM para Java como clase y utilice todos sus métodos y propiedades.

- **C++.** El código fuente del analizador en C de IBM está disponible, y se puede llamar a las clases de este motor de forma directa. El motor de IBM también está disponible como DLL. El motor DOM de Microsoft sólo se puede usar como DLL o como objeto COM. Probablemente, la manera más fácil de incorporar código en C++ consista en usar Microsoft ADO como COM.
- **Visual Basic.** El motor DOM de Microsoft puede ejemplificarse en una aplicación de VB como objeto. Cuando se ha creado el objeto se puede acceder al documento por medio de todas las interfaces del DOM.

2.3.5- XPath

XPath (Lenguaje de camino XML) es una especificación por derecho propio, pero es usada principalmente por XSLT. La especificación XPath define cómo localizar un elemento específico dentro de un documento XML. Esto se logra haciendo referencia a nodos específicos en el documento XML; aquí, nodo se refiere a cualquier porción de datos XML, incluyendo elementos, atributos o datos de texto. En la especificación XPath, un documento XML se considera un árbol de estos nodos, donde se puede acceder a cada nodo especificando su posición en el árbol en que se encuentra.

XPath es un lenguaje no XML que se usa para dirigirse a partes de un documento XML. Es un lenguaje no XML, ya que no se describe para construcciones XML, como elementos y atributos, aunque se use para dirigirse a documentos XML. La sintaxis que emplea XPath está diseñada para ser utilizada en URI y valores de atributos XML, lo que significa que debe ser muy concisa. Ésta es la razón por la que XPath no se implementa como vocabulario XML. El nombre XPath se basa en el concepto de utilizar una notación de ruta para dirigirse a los documentos XML.

Para habilitar una forma de dirigirse a partes de un documento XML, XPath trata el documento como un árbol de nodos, lo cual es un modo bastante habitual de percibir los documentos XML. Los nodos de un documento XML pueden dividirse en nodos de elementos, nodos de atributos y nodos de texto. Algunos nodos tienen nombres, que pueden estar formador por un URI de

espacio de nombres opcional y un nombre local; un nombre que incluye un espacio de nombres se denomina expandido. A continuación se exponen los distintos tipos de nodos que pueden aparecer en un árbol XPath:

- Nodos raíz
- Nodos de elementos
- Nodos de texto
- Nodos de atributos
- Nodos de espacios de nombres
- Nodos de instrucciones de procesamiento
- Nodos de comentarios

Siempre hay un nodo raíz único que sirve como raíz de un árbol XPath y que aparece como su primer nodo. Todos los elementos de un documento tienen su correspondiente nodo de elemento que aparece en el árbol bajo el nodo raíz. Dentro de un nodo de elemento están todos los demás tipos de nodos, que se corresponden con el contenido del elemento. Los nodos de elementos pueden tener un identificador único asociado, que es útil a la hora de hacer referencia al nodo con XPath.

La premisa principal que subyace a XPath es el recorrido de un documento XML para llegar a un nodo determinado. Este recorrido se lleva a cabo por medio de expresiones. La especificación XPath describe cómo usar la sintaxis XPath para construir expresiones. Cuando se evalúan las expresiones XPath, acaban siendo un objeto de datos de uno de los tipos siguientes:

- node-set. Un conjunto de nodos
- boolean. Un valor true-false
- number. Un número de coma flotante
- string. Una cadena de texto

El modo en que se evalúa una expresión XPath depende completamente del contexto de la expresión, que no viene determinado por XPath. En otras palabras, ésta es la parte abstracta de XPath que se puede utilizar con Xpointer y XSLT para dirigirse a partes de documentos. El contexto de una expresión Xpath viene determinado por Xpointer o XSLT, que a su vez determina cómo se evalúa la expresión.

2.3.6- Bases de datos nativas XML

Durante los últimos años, XML se ha convertido en una de las más importantes tecnologías para la gestión de infraestructuras basadas en el Web.

La aceptación de esta tecnología se debe principalmente a dos de sus características: su capacidad como mecanismo de integración y la separación existente entre contenidos y presentación.

XML es un lenguaje idóneo para la construcción de sistemas integrados utilizando middleware orientado a mensajes o MOM. La alta extensibilidad y robustez de XML le convierte en la mejor solución para la integración de aplicaciones a través de mensajes, permitiendo una rápida adaptación a los continuos cambios en las necesidades de negocio, además de flexibilizar el intercambio de datos entre diferentes sistemas de información.

Por otro lado, la separación de los contenidos de su representación supone una mejora considerable en la gestión de la información. Los contenidos se crean y mantienen una sola vez reduciendo la multiplicidad de formatos de salida para los distintos dispositivos de visualización existentes en el mercado.

A pesar de las ventajas de las características comentadas, una última condición necesaria para el desarrollo definitivo de XML en la empresa es la del almacenamiento de la información. El dominio de XML sobre el intercambio de datos en el mercado depende de las capacidades de almacenamiento de las que se disponga. XML no puede ser una opción seria de negocio para el intercambio de información entre empresas si no se puede almacenar de forma eficiente. Los altos volúmenes de información gestionada hacen que las

conversiones de formato y el almacenamiento de datos en ficheros sean soluciones poco robustas y para nada eficientes.

La solución idónea para el almacenamiento de datos en formato XML es la de disponer de sistemas de gestión de información que sean capaces de gestionar estos datos en su formato nativo.

El almacenamiento de datos en formato XML nativo tiene diversas ventajas sobre la gestión realizada por las bases de datos relacionales convencionales. No es necesaria ninguna conversión de formatos y la estructura de los documentos se mantiene intacta. Los sistemas relacionales típicos no están dotados para la gestión jerárquica de datos que tiene XML, por lo que necesitan mecanismos de conversión a medida para el almacenamiento de información en este formato. Además tampoco disponen de mecanismos efectivos para contemplar el dinamismo de los documentos XML cuyo formato puede variar con facilidad.

Una de las soluciones más eficaces para almacenar datos en formato XML es la de las bases de datos nativas XML. Estas bases de datos proporcionan la funcionalidad necesaria para almacenar datos en formato XML y realizar consultas en la información gestionada. Están, por tanto, especializadas en almacenar datos en formato XML manteniendo el modelo XML asociado intacto.

Una base de datos nativa XML define un modelo lógico para cada documento XML y almacena y recupera información ajustada a ese modelo lógico. Como mínimo, estos modelos lógicos deben incluir elementos, atributos, secciones PCDATA y deben contemplar la ordenación de documentos.

Además, las bases de datos nativas XML tienen un documento XML como unidad fundamental de almacenamiento, al igual que una base de datos relacional tiene un registro que pertenece a una tabla determinada.

Otro aspecto a considerar es que no se requiere que tenga una estructura asociada para el almacenamiento físico de los datos. Puede ser construida sobre una base de datos relacional u orientada a objetos, o bien utilizar

formatos de almacenamiento propietarios como ficheros indexados o comprimidos.

En realidad, las bases de datos nativas XML no representan un nuevo modelo de base de datos de bajo nivel ni pretenden reemplazar a los sistemas de almacenamiento actuales. Se han planteado como una herramienta de soporte al desarrollo que permitan a los desarrolladores disponer de una solución robusta para el almacenamiento y manipulación de datos en formato XML.

2.3.7.1 – Almacenamiento de XML nativo

Las bases de datos nativas XML almacenan los documentos creando modelos lógicos. Estos modelos se basan directamente en XML o en alguna de las tecnologías relacionadas como DOM o Infoset. Estos modelos incluyen diferentes niveles de agregación y complejidad, así como un soporte completo para la gestión de datos semi-estructurados.

Los modelos son automáticamente mapeados por las bases de datos XML nativas al mecanismo de almacenamiento correspondiente. El mapeo utilizado por estas bases de datos garantizará la correcta utilización del modelo asociado al documento original.

Una vez que se ha almacenado la información, y con el objeto de aprovechar al máximo la capacidad de estos sistemas, es necesario seguir utilizando herramientas XML que permitan realizar otros tipos de operaciones.

La principal ventaja de las bases de datos nativas XML es la abstracción. El programador no tiene porqué conocer los detalles de cómo se almacena la información, pudiendo construir sus aplicaciones con total libertad.

2.3.7.2 – Dependencia del esquema

Las bases de datos nativas XML gestionan los documentos como si fueran colecciones de datos, permitiendo realizar consultas y manipulaciones sobre esos documentos en forma de conjuntos de información. Es un concepto similar al de las tablas en las bases de datos relacionales.

La diferencia con los sistemas relacionales es que no todas las bases de datos nativas XML necesitan un esquema para generar una colección de documentos. Esto quiere decir que se puede almacenar cualquier documento XML en la colección independientemente de su estructura, permitiendo la realización de consultas en todo el conjunto de datos con una gran flexibilidad.

Las bases de datos nativas XML que soportan esta característica se denominan independientes del esquema.

Disponer de colecciones de documentos independientes del esquema proporciona a la base de datos una alta flexibilidad y facilita el desarrollo de aplicaciones. Por otro lado, esta característica dificulta la tarea de los administradores de base de datos debido a posibles problemas de integridad de datos.

En ocasiones es preferible disponer de menos flexibilidad y asegurar una correcta integridad referencial, debiendo decantarse por soluciones dependientes del esquema u otro tipo de aplicaciones de almacenamiento de documentos XML.

2.3.7.3 – Capacidades de consulta

En un principio, XPath es el lenguaje de consulta estándar para bases de datos nativas XML. Así, y con el objeto de facilitar la realización de consultas en bases de datos, se ha modificado el estándar inicial para dotar a XPath de capacidades de consulta en colecciones de documentos.

En la actualidad se está desarrollando un lenguaje de consulta más orientado a bases de datos XML llamado XQuery. Aunque no hay una versión estándar del lenguaje, algunos fabricantes ya proveen prototipos del mismo en sus implementaciones.

Para optimizar la realización de consultas, las bases de datos nativas XML ya disponen de mecanismos de indexado que aumentan las capacidades de búsqueda en colecciones de documentos de gran tamaño. Los detalles de implementación de estos mecanismos de indexado pueden variar entre los

distintos fabricantes, pero casi todos ellos proveen de mecanismos de este tipo para la optimización de las consultas.

2.3.7.4 - XIndice

XIndice es una base de datos desarrollada en Java que se ha diseñado para el almacenamiento de grandes cantidades de pequeños documentos XML. Se distribuye en forma de código abierto por la Apache Software Foundation.

Puede indexar tanto valores estructurales como contenidos, almacenando los documentos XML en formato comprimido para ahorrar espacio en disco.

Los documentos se estructuran en una jerarquía de colecciones y pueden ser consultados utilizando XPath. Como lenguaje de actualización se puede emplear XUpdate, y para la gestión de enlaces entre documentos se utiliza un lenguaje experimental similar a XLink.

Soporta tres tipos de interface de programación: XML:DB, un API para CORBA y un plugin XML-RPC que soporta accesos desde PHP, Perl y Applescript. La funcionalidad del servidor es extensible utilizando la tecnología XMLObjects.

2.3.7.5 – Estudio comparativo

Nombre	Licencia	Sistema Almacen.	T	E	C	I[1]	A	J	Acceso	API	Otros
4Suite	Open Source	Orientado a Objetos	Sí	¿?	Sí	Sí	Sí	Sí	HTTP, RPC, FTP, CORBA, SOAP	Python	SAX, DOM, XSLT, XInclude, XPointer, XLink, XPath, RDF, XUpdate
Birdstep RDM XML	Comercial	Orientado a Objetos	Sí	No	No	Sí	No	Sí		C, C++, Java	SAX, DOM, XPAth, Dispositivos móviles
Centor Interaction Server	Comercial	Solución Propietaria	¿?	Sí	Sí	Sí	Sí	¿?	JDBC, ODBC, EJB	C++, Java	Motor de workflow y estilos. Gestor de presentación
Cerisent XQE	Comercial	Solución Propietaria	Sí	¿?	Sí	¿?	¿?	¿?			XQuery, XPath, XML schemas, SGML, HTML
Coherty XML DB	Comercial	Solución Propietaria	Sí	Sí	Sí	Sí	Sí	Sí	HTTP	Java, JSP	XPath, XSL, XSQL, XML/SQL, DOM
DBDOM	Op. Source	Relacional	Sí	Sí	Sí	Sí	Sí	Sí	JDBC	Java	DOM, PosgreSQL
dbXML	Comercial	Solución Propietaria	¿?	Sí	Sí	Sí	Sí	¿?	HTTP, XML/RPC	Java, Javascript, JSP, Python	XPath, XSLT, ViXen, XUpdate, XQuery

[1] T: Transaccionalidad; E: Eventos o Triggers; C: Concurrencia; I: Indexado; A: Control de Acceso; J: Enlazado de datos

Nombre	Licencia	Sistema Almacen.	T	E	C	I[2]	A	J	Acceso	API	Otros
DOM-Safe	Comercial	Sistema Propietario	Sí	¿?	Sí	¿?	¿?	¿?	HTTP, WebDAV		XPath, XLink, SAX2, DOM
eXist	Open Source	Relacional	¿?	¿?	Sí	Sí	Sí	¿?	HTTP, XML/RPC, WebDAV	Java	XPath, Oracle, MySQL, PostgreSQL
eXtc	Comercial	Sistema Propietario	Sí	¿?	Sí	¿?	Sí	Sí	HTTP, SOAP, WSDL		DOM, SQL, XPath
eXtensible Information Server	Comercial	OO, relacional y otros	Sí	Sí	¿?	Sí	¿?	Sí	COM, .NET	Java, VisualBasic	XConnects, XPath, XQuery, Updategrams
GoXML DB	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	Sí	JDBC, GoXML DB Explorer, WebDAV	Java, scripts	XQuery, DRDs, XML schemas, MD5
Infonyte DB	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	Sí	HTTP	Collection, JSP, Java, Servlets	DOM, XQL, XPath, XSLT
Ipedo XML Database	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	Sí	SOAP, .NET, COM, EJB, WSDL, HTTP	Java	XQuery, DTDs, XML schemas, XPath, XSLT, DOM

[2] T: Transaccionalidad; E: Eventos o Triggers; C: Concurrencia; I: Indexado; A: Control de Acceso; J: Enlazado de datos

Nombre	Licencia	Sistema Almacen.	T	E	C	I[3]	A	J	Acceso	API	Otros
Lore	Invest.	Semi Estr.	¿?	¿?	Sí	Sí	Sí	¿?			DataGuide
Lucid XML DM	Comercial	Propietario	Sí	¿?	Sí	Sí	Sí	¿?		Java	XPath
MindSuite XDB	Comercial	OO	Sí	Sí	Sí	Sí	Sí	¿?	CORBA	Java, C++	DOM, XPath, XSLT
Natix	Comercial	Ficheros	No	¿?	Sí	Sí	Sí	¿?		Java, C++	XPath
Neocore XML Management System	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	¿?	HTTP, EJB, HTTPS, COM, .NET	Java, C++	XPath, X-Aware
Ozone	Open Src.	OO	Sí	No	Sí	¿?	Sí	¿?	RMI	Java	DOM
Sekaiju	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	¿?	HTTP, COM, SOAP	Visual Basic	RelaxNG, Xbatch
SQL/XML-IMDB	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	¿?	.NET, HTTP	C++, PHP, VBasic, Delphi, Perl	XQuery, SQL92
Tamino XML Server	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	¿?	JDBC, ODBC, SOAP, WebDAV	JSP	X-Tension, XPort, DataMap, DOM, JDOM, SAX, XSLT, XQuery, CSS, XPlorer, XNode, XApplication

[3] T: Transaccionalidad; E: Eventos o Triggers; C: Concurrencia; I: Indexado; A: Control de Acceso; J: Enlazado de datos

Nombre	Licencia	Sistema Almacen.	T	E	C	I[4]	A	J	Acceso	API	Otros
TeamXML	Comercial	Sistema Propietario	¿?	¿?	Sí	Sí	Sí	Sí	EJB, SOAP, .NET	OpenAPI (Java)	TeamSite
TeraText DBS	Comercial	Sistema Propietario	¿?	¿?	Sí	Sí	Sí	¿?	SOAP, WebDAV, LDAP	Java, C++	SGML, Unicode, PDF, Z39.50, MsWord
TEXTML	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	Sí	¿?	COM, COM+, ASP, WebDAV, .NET	Java, C++	MTS, SAP, Lotus, DataExchanger
TigerLogic XDMS	Comercial	Sistema Propietario	Sí	¿?	Sí	Sí	¿?	¿?	SOAP, HTTP, JCA	Java	XPath, XSLT, DTD, XML schemas, XA
TOTAL XML	Comercial	Objeto-Relacional	¿?	¿?	Sí	¿?	Sí	¿?	JDBC, ODBC	Java	DTD, SQL, XPath, SAX, StrivaDETAIL, DOM
Virtuoso	Comercial	Propietario (Relacional)	Sí	Sí	Sí	Sí	Sí	Sí	JDBC, ODBC, HTTP, SMTP, LDAP, POP3, OLE-DB, J2EE, WebDAV, SOAP, .NET		XA, XPath, XQuery, XSLT, SQL92

[4] T: Transaccionalidad; E: Eventos o Triggers; C: Concurrencia; I: Indexado; A: Control de Acceso; J: Enlazado de datos

Nombre	Licencia	Sistema Almacen.	T	E	C	I[5]	A	J	Acceso	API	Otros
XDB	Open Source	Relacional	¿?	¿?	¿?	¿?	¿?	¿?		C++	XPath, DOM, PosgreSQL, SAX, DTD, XML schemas, XMLQuery,
XDBM	Open Source	Sistema Propietario	¿?	¿?	¿?	¿?	¿?	¿?			DOM
X-Hive/DB	Comercial	OO y Relacional	Sí	¿?	Sí	Sí	Sí	Sí	JDBC, WebDAV	Java	XQuery, XPath, DOM, XSLT, Xlink, XPointer
XIndice	Open Source	Sistema Propietario	No	No	¿?	Sí	¿?	Sí	XML:DB, CORBA, XML-RPC	Java	XPath, Xlink, XUpdate, XMLObjects
Xyleme Zone Server	Comercial	Sistema Propietario	¿?	Sí	Sí	Sí	Sí	¿?	SOAP	Java, C++	XQuery,

[5] T: Transaccionalidad; E: Eventos o Triggers; C: Concurrencia; I: Indexado; A: Control de Acceso; J: Enlazado de datos

2.4- APIS

2.4.1- dbXML

La idea principal del dbXML es proporcionar una manera simple de salvar y de manejar una gran cantidad de documentos XML. Esto se logra salvando documentos en colecciones, donde cada documento individual se salva en una forma pre-analizada y comprimida. Esto incrementa notoriamente al velocidad al trabajar con datos XML.

Almacenar documentos en colecciones proporciona un mecanismo fácil para preguntar y manipular los documentos como conjunto. Esta Api presenta una gran flexibilidad para el manejo de documentos mixtos, incluso mezclar los documentos de esquemas totalmente diversos en la misma colección.

DBXML es un sistema de gerencia de datos diseñado específicamente para las colecciones de documentos XML. Se incluye fácilmente en aplicaciones existentes, altamente configurable, y extensible.

La compresión de XML permite que los documentos sean salvados en forma de tokens, permitiendo un funcionamiento rápido y la conservación de la memoria.

Los documentos se ocultan en memoria para mejorar su funcionamiento. El almacenamiento en memoria inmediata se puede controlar mediante la colección y los niveles del documento.

El dbXML incluye:

- Almacenamiento de las colecciones de documentos XML.
- Motor de Base de datos multihilo, que optimiza los datos XML.
- Almacenamiento comprimido pre-analizado del documento.
- Motor XPath query.
- Índices de la colección para mejorar el rendimiento de las consultas.

El servidor del dbXML viene con un conjunto de las herramientas de la línea de comandos que permiten realizar todas las funciones básicas de administración.

DBXML proporciona además el XUpdate, el cual proporciona un lenguaje simple para actualización de documentos XML. Permite declarar específicamente qué cambios deben ser realizados.

2.5- CONCLUSIONES

Las aplicaciones Web de la actualidad se enfrentan con un número abundante de problemas que ni siquiera se habían tenido en cuenta hace diez años. Los sistemas, distribuidos a lo largo de kilómetros, deben funcionar rápidamente y de manera intachable. Los datos, provenientes de sistemas heterogéneos, bases de datos, servicios de directorio y aplicaciones, deben transferirse sin perder ni un solo decimal. Las aplicaciones deben ser capaces de comunicarse no sólo con otros componentes de negocio, sino con todo un

conjunto de sistemas de negocio, a menudo entre diferentes compañías y tipos de tecnología. Los clientes ya no están limitados a clientes de tipo “pesado”, sino que pueden ser navegadores de web con soporte de HTML, teléfonos móviles con soporte de WAP (Protocolo de Aplicación Inalámbrico) u ordenadores de bolsillo con lenguajes de marcado totalmente diferentes. Los datos y la transformación de éstos, se han vuelto la pieza central decisiva de todas las aplicaciones desarrolladas en la actualidad.

XML ofrece el medio para que los programadores cumplan todos estos requisitos. **XML** hace los datos portables. Por otro lado la plataforma **Java** hace el código portable. Y los APIs Java para XML hacen fácil el uso de XML. Si ponemos todo esto junto tenemos la combinación perfecta: **portabilidad de datos, portabilidad de código y facilidad de uso**. De hecho, con los APIs de Java para XML, podremos obtener los beneficios del XML con un uso directo de XML muy pequeño.

XML se reconoce hoy en día como uno de los lenguajes de mayor impacto para la integración de servicios a través de la web. Es por eso que las empresas están descubriendo rápidamente los beneficios de usar XML para la integración de los datos, tanto internamente para compartir datos legales entre departamentos, como externamente para compartir datos con otras empresas. Y a causa de la integración de datos que ofrece XML, se ha convertido en indispensable para los nuevos proyectos relacionados con la Web.

Las arquitecturas de aplicaciones distribuídas hacen uso extensible de este lenguaje como mecanismo integrador entre los diferentes niveles de una aplicación. Desde esta perspectiva, la inmensa mayoría de documentos XML son creados por programas y procesados por otros para posteriormente ser distribuídos.

La parte realmente dura del desarrollo de servicios Web es la programación de la infraestructura, o "tuberías", como las capacidades de seguridad y de mensajería, el manejo de transacciones distribuidas, y el control de almacenes de conexiones. Otra dificultad es que esos servicios web deben ser capaces de manejar un enorme número de usuarios simultáneamente, por

eso las aplicaciones deben ser altamente escalables. Estos requerimientos son los que ofrece exactamente la plataforma Java TM, Enterprise Editon (J2EE TM). Si añadimos a esto que la plataforma J2EE es una tecnología demostrada con múltiples vendedores ofreciendo productos compatibles a día de hoy, nos encontramos con que la plataforma J2EE es la mejor plataforma para desarrollar servicios Web. Y con los nuevos APIs de Java para XML, el desarrollo de servicios Web se está simplificando cada vez más.