



I.E.S. "POLITÉCNICO JESÚS MARÍN"
Departamento de Informática
Módulo: "Lenguajes de marca y sistemas de gestión de la información"



1.DTD						
1.1 Declaración del DTD.			1.2 Vinculación desde el archivo XML			
<!DOCTYPE elementoPadre [elementos hijo y atributos]>			Vinculación Interna		Vinculación Externa	
			<pre><?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE Casas_Rurales [<ELEMENT Casas_Rurales (Casa)*> <ELEMENT Casa (Dirección, Descripción, Estado, Tamano)> <ELEMENT Dirección (#PCDATA) > <ELEMENT Descripción (#PCDATA) > <ELEMENT Estado (#PCDATA) > <ELEMENT Tamano (#PCDATA)]> <Casas_Rurales> <Casa>....</Casa> </Casas_Rurales></pre>		Privado <pre><?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE concesionarios SYSTEM "concesionario.dtd"> <concesionarios> ... </concesionarios></pre> Público <pre><!DOCTYPE concesionarios PUBLIC "IDENT" "concesionario.dtd"> <concesionarios> ... </concesionarios></pre>	
1.3 Elementos			XML		DTD	
Contienen ya los datos <!ELEMENT elemento (#PCDATA) > Es un elemento vacío <!ELEMENT elemento EMPTY>			<pre><Estado>Aceptable</Estado>
</pre>		<pre><!ELEMENT Estado (#PCDATA) > <!ELEMENT br EMPTY></pre>	
Contienen elementos hijos <ELEMENT elemento_padre (hijo1c, hijo2c, (hijo4c hijo5c))> c (cardinalidad) puede valer: ? : uno o ninguna vez * : de cero a N veces + : de 1 a N veces nada (por defecto): 1 vez			Secuencia <pre><email> <de>Juan</de> <para>Ana</para> <para>Eva</para> <asunto>Reunión</asunto> </email></pre> Opcionalidad <pre><aviso> <titulo>Ventas</titulo> <grafico>foto.jpg<grafico> </aviso></pre>		<pre><!ELEMENT email (de, para+, cc*, bc*, asunto?, cuerpo?)> <!ELEMENT aviso (titulo, (parrafo grafico))> <aviso> <titulo>Ventas</titulo> <parrafo>bla, bla...<parrafo > </aviso></pre>	
1.3 Atributos	<!ATTLIST	nombre_elemento	nombre_atributo	tipo_atributo	caracter>	
es una cuatriada. (Aconsejable escribir una cuatriada por cada atributo)		nombre del elemento al que se asigna el atributo	nombre del atributo	<u>CDATA:</u> cadena alfanumérica (cualquier cosa). <u>(valor1 valor2):</u> puede valer valor 1 o valor2 <u>ID:</u> clave, no se repite <u>IDREF:</u> clave ajena de otro elemento	<u>"valor":</u> valor por defecto (se "") <u>#IMPLIED:</u> opcional <u>#REQUIRED</u> obligatorio <u>#FIXED:</u> obligatorio y fijo	<pre><perro fecha_nacimiento="2020-01-01">Pastor Alemán</perro> <!ELEMENT perro (#PCDATA)> <!ATTLIST perro fecha_nacimiento CDATA "23-02-1988"> <persona sexo="hombre"> <nombre>Juan</nombre> <apellidos>López Pérez</apellidos> </persona> <!ELEMENT persona (nombre, apellidos)> <!ATTLIST persona sexo (hombre mujer) #REQUIRED</pre>

2.XSD		
2.1 Declaración del documento .XSD (se guarda con la extensión .xsd) <pre><?xml version="1.0" encoding="UTF-8"?></pre> <pre><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"></xs:schema></pre>		
2.2 Vinculación desde un archivo XML En el elemento raíz del xml se define <u>espacio de nombres</u> con un prefijo (ej. xsi) y con el atributo noNamespaceSchemaLocation (debe llevar el prefijo) se indica la ruta del archivo con el .xsd <pre><vehiculos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="XSDvehiculos.xsd"></pre>		
2.3 Tipos de datos simples (se asignan al atributo type=)		
Van precedidos del prefijo del espacio de nombres Por, ej. xs xs:string : secuencia de caracteres xs:integer : numero entero xs:decimal : numero con parte decimal xs:date : fecha específica del calendario gregoriano, en formato "YYYY-MM-DD"; xs:time : una instancia de tiempo que ocurre cada día, en formato "hh:mm:ss" Ver más tipos y ejemplos		
2.4 Elementos simples. (solo contienen datos simples NO hijos NO atributos)	XML	XSD
	<pre><nombre>Juan</nombre></pre> <pre><edad>23</edad></pre> <pre><nota>8.5</nota></pre>	<pre><xs:element name="nombre" type="xs:string"/></pre> <pre><xs:element name="edad" type="xs:integer" /></pre> <pre><xs:element name="nota" type="xs:decimal" /></pre>
	2.3 Elementos Complejos con hijos simples (Tienen hijos o atributo)	<pre><xs:element name="alumno"></pre> <pre><xs:complexType></pre> <pre><xs:sequence></pre> <pre><xs:element name="nombre" type="xs:string" /></pre> <pre><xs:element name="edad" type="xs:integer" /></pre> <pre><xs:element name="nota" type="xs:decimal" /></pre> <pre></xs:sequence></pre> <pre></xs:complexType></pre> <pre></xs:element></pre>
Si no importa el orden en lugar de xs:sequence se escribe xs:all		
	Elección (uno de varios) <i>(xs:choice)</i> <pre><empleado></pre> <pre><hombre>Juan</hombre></pre> <pre></empleado></pre> <p>O</p> <pre><empleado></pre> <pre><mujer>Maria</mujer></pre> <pre></empleado></pre>	<pre><xs:element name="empleado"></pre> <pre><xs:complexType></pre> <pre><xs:choice></pre> <pre><xs:element name="hombre" type="xs:string" /></pre> <pre><xs:element name="mujer" type="xs:string" /></pre> <pre></xs:choice></pre> <pre></xs:complexType></pre> <pre></xs:element></pre>
	Cardinalidades (ver más) <ul style="list-style-type: none"> Es un atributo del hijo Por defecto es 1,1 minOccurs =número mínimo de ocurrencias del hijo dentro del padre maxOccurs =número máximo de ocurrencias del hijo dentro del padre	<pre><xs:element name="alumno"></pre> <pre><xs:complexType></pre> <pre><xs:sequence></pre> <pre><xs:element name="nombre" type="xs:string" /></pre> <pre><xs:element name="nota" type="xs:decimal"</pre> <pre>minOccurs="1" maxOccurs="10" /></pre> <pre><xs:element name="telefono" type="xs:integer"</pre> <pre>minOccurs="0" maxOccurs="unbounded"/></pre> <pre></xs:sequence></pre> <pre></xs:complexType></pre> <pre></xs:element></pre>
<i>Suponemos un número máximo de 10 notas y mínimo de una y puede no tener teléfono o un número ilimitado de ellos</i>		

<h2>2.4 Elementos Complejos con Hijos Complejos</h2>	<pre> <alumnos> <alumno> <nombre>Juan</nombre> <nota>8.5</nota> </alumno> </alumnos> <alumnos> <alumno> <nombre>Juan</nombre> <nota>8.5</nota> </alumno> </alumnos> </pre>	<p>Anidamiento:</p> <pre> <xs:element name="alumnos"> <xs:complexType> <xs:sequence> <xs:element name="alumno" > <xs:complexType> <xs:sequence> <xs:element name="nombre" type="xs:string" /> <xs:element name="nota" type="xs:decimal" /> </xs:sequence> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </pre> <p>Referencia:</p> <pre> <xs:element name="alumnos"> <xs:complexType> <xs:sequence> <xs:element ref="alumno" minOccurs="0" maxOccurs="unbounded"></xs:element> </xs:sequence> </xs:complexType> </xs:element> </pre> <pre> <xs:element name="alumno" > <xs:complexType> <xs:sequence> <xs:element name="nombre" type="xs:string" /> <xs:element name="nota" type="xs:decimal" /> </xs:sequence> </xs:complexType> </xs:element> </pre>
<h2>2.5 Atributos en elementos complejos</h2> <ul style="list-style-type: none"> • <xs:attribute name= type= /> • Después de la declaración de los subelementos • Siempre tiene un tipo simple • No impone un orden • Si no se define un tipo, será del tipo: anySimpleType. (Cualquier cadena de caracteres válidos) • Si no se dice nada es opcional • Tiene a su vez tres atributos para restringir los valores: use, default y fixed <ul style="list-style-type: none"> ○ use: <ul style="list-style-type: none"> ▪ required: el atributo es obligatorio ▪ optional: puede o no aparecer (es el valor por defecto) ○ default: <ul style="list-style-type: none"> ▪ valor por defecto del atributo ▪ fixed: valor fijo. Puede aparecer o no, pero si aparece solo puede tener ese valor. 	<pre> <alumno dni="323B"> <nombre>Juan</nombre> <nota>8.5</nota> </alumno> </pre>	<pre> <xs:element name="alumno" > <xs:complexType> <xs:sequence> <xs:element name="nombre" type="xs:string" /> <xs:element name="nota" type="xs:decimal" /> </xs:sequence> <xs:attribute name="dni" type="xs:string" use="required"></xs:attribute> </xs:complexType> </xs:element> </pre>

<p>2.6 Atributos en elementos simples</p> <ol style="list-style-type: none"> 1. El elemento simple pasa ser complejo. (xscomplexType) 2. Se indica que su contenido es simple (xssimpleContent) 3. Se hace una extensión sobre el tipo simple del elemento añadiendo el atributo) <pre><xs:extension base="xs:string"> <xs:attribute name= /> </xs:extensión></pre>	<pre><alumno dni="323B"> <nombre>Juan</nombre> <nota eval="1">8.5 </nota> </alumno></pre> <p><i>En este ejemplo el tipo simple del elemento nota es xs:decimal y el del atributo "eval" puede ser "xs:integer"</i></p>	<pre><xs:element name="alumno" > <xs:complexType> <xs:sequence> <xs:element name="nombre" type="xs:string" /> <xs:element name="edad" type="xs:integer" /> <xs:element ref="nota" /> </xs:sequence> <xs:attribute name="dni" type="xs:string" use="required"></xs:attribute> </xs:complexType> </xs:element></pre> <pre><xs:element name="nota"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:decimal"> <xs:attribute name="eval" type="xs:integer"></xs:attribute> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element></pre>
<p>2.7 Tipos complejos con un nombre</p> <p>Si un tipo complejo se va a utilizar varias veces puede ser útil darle un nombre y reutilizarlo por ese nombre</p>	<pre><alumno dni="323B"> <nombre>Juan</nombre> <nota eval="1">8.5 </nota> </alumno></pre>	<pre><xs:element name="alumno" type="persona"> </xs:element> <xs:complexType name="persona"> <xs:sequence> <xs:element name="nombre" type="xs:string" /> <xs:element ref="nota" /> </xs:sequence> <xs:attribute name="dni" type="xs:string" use="required"></xs:attribute> </xs:complexType> </xs:schema></pre>
<p>2.8 Tipos restringidos de tipos simples (facetas) Ver más aquí</p> <p>Se pueden definir tipos simples basado en restricciones a</p>	<pre><alumno dni="323B"> <nombre>Juan</nombre> <nota eval="1">8.5 </nota> </alumno></pre>	<p>Restricciones minInclusive y maxInclusive (equivalente a \leq y \geq, para $>$ y $<$ utilizar minExclusive y maxExclusive)</p> <p><i>Restringir el valor del atributo eval a 1,2 o 3</i></p> <pre><xs:element name="nota"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:decimal"> <xs:attribute name="eval" type="tipoevaluacion"></xs:attribute> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element></pre> <pre><xs:simpleType name="tipoevaluacion"> <xs:restriction base="xs:integer"> <xs:minInclusive value="1"/> <xs:maxInclusive value="3"/> </xs:restriction> </xs:simpleType></pre> <div> <p>Los nuevos nombres no llevan el prefijo xs:</p> </div>

Enumeration (solo son válidos los valores enumerados)	length	maxLength y minLength
<pre><xs:simpleType name="tipoevaluacion"> <xs:restriction base="xs:integer"> <xs:enumeration value="1"/> <xs:enumeration value="2"/> <xs:enumeration value="3"/> </xs:restriction> </xs:simpleType></pre>	<pre><xs:simpleType name="dni"> <xs:restriction base="xs:string"> <xs:length value="8" /> </xs:restriction> </xs:simpleType></pre> <p><i>(si queremos que la longitud del dni sea exactamente 8)</i></p>	<pre><xs:simpleType name="tiponombre"> <xs:restriction base="xs:string"> <xs:minLength value="1"/> <xs:maxLength value="10"/> </xs:restriction> </xs:simpleType></pre> <p><i>(si queremos que la longitud del nombre sea como mínimo 1 y máximo 10)</i></p>
fractionDigits (número de decimales) y totalDigits (número de dígitos)	<h2>Pattern</h2> <p>(restringir los valores a los que cumplan un patrón)</p>	
<pre><xs:simpleType name="tiponota"> <xs:restriction base="xs:decimal"> <xs:fractionDigits value="1"/> <xs:totalDigits value="2"/> </xs:restriction> </xs:simpleType></pre> <p><i>(si queremos que la nota solo tenga un decimal y como mucho 2 dígitos)</i></p>	<pre><xs:element name="iniciales"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:pattern value="[A-Z][A-Z][A-Z]"/> </xs:restriction> </xs:simpleType> </xs:element></pre> <p><i>(el elemento "iniciales" solo aceptará 3 letras mayúsculas)</i></p>	<pre><xs:element name="iniciales" type="tipoiniciales"></xs:element></pre> <pre><xs:simpleType name="tipoiniciales"> <xs:restriction base="xs:string"> <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/> </xs:restriction> </xs:simpleType> </xs:element></pre> <p><i>(el elemento "iniciales" solo aceptará 3 letras mayúsculas o minúsculas)</i></p>
Pattern (otros ejemplos)		
<pre><xs:pattern value="[abc]"/></pre>	(solo puede valer o a o b o c)	
<pre><xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/></pre>	Cinco dígitos	
<pre><xs:pattern value="([a-z])*"/></pre>	Un numero de cero o más letras mayúsculas	
<pre><xs:pattern value="([a-z][A-Z])+"/></pre>	Un número indeterminado de parejas (al menos una) de una letra minúscula y otra mayúscula	
<pre><xs:pattern value="hombre mujer"/></pre>	Solo puede tomar los valores "hombre" o "mujer"	
<pre><xs:pattern value="[a-zA-Z0-9]{8}"/></pre>	Una serie 8 caracteres que pueden ser letras minúsculas, mayúsculas y números.	
<pre><xs:simpleType name="tipotelefono"> <xs:restriction base="xs:string"> <xs:pattern value="[0-9]{9}"/> </xs:restriction> </xs:simpleType></pre>	El tipo teléfono estaría compuesto por 9 dígitos	
<pre><xs:simpleType name="tipotelefono"> <xs:restriction base="xs:string"> <xs:pattern value="[0-9]{3}-[0-9]{6}"/> </xs:restriction> </xs:simpleType></pre>	El tipo teléfono estaría compuesto por 3 dígitos un guion y 6 dígitos. Esto es posible porque el guion no se utiliza para hacer patrones, en otro caso, cuando queremos que coincida exactamente un carácter que se utiliza para hacer patrones ([, (, +, *) se le pone antes la barra \, \[, \+, *. En cualquier caso, siempre se puede utilizar la barra \ para hacer coincidir un carácter concreto. También funcionaría <pre><xs:pattern value="[0-9]{3}\-[0-9]{6}"/></pre>	
<pre><xs:simpleType name="tipotelefono"> <xs:restriction base="xs:string"> <xs:pattern value="\([+][0-9]{3}\)[0-9]{9}"/> </xs:restriction> </xs:simpleType></pre>	El tipo teléfono sería de la forma (+034)926534675	