

# Tema 4

# Lenguajes de Marcas XML

Lenguajes de Marcas y Sistemas de Gestión de la Información

# Metalinguaje XML

e**X**tensible **M**arkup **L**anguage,  
“lenguaje de marcas extensible”

- Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).
- XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.
- Es un estándar para el intercambio de información estructurada entre diferentes plataformas o aplicaciones.

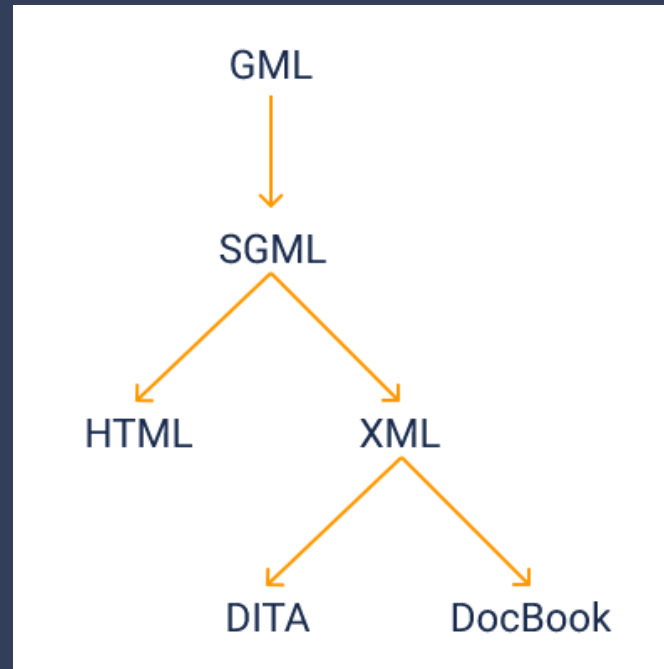
# Metalinguaje XML

e**X**tensible **M**arkup **L**anguage,  
“lenguaje de marcas extensible”

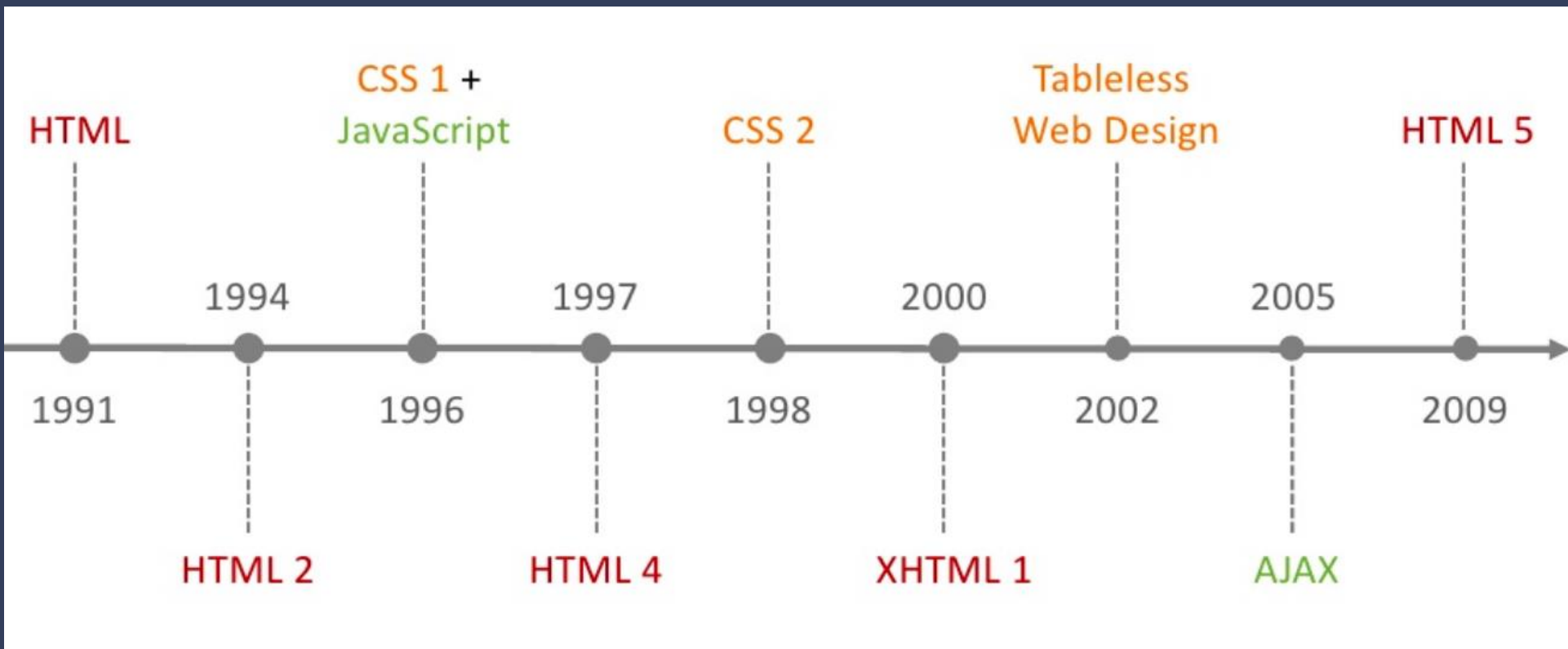
Sirve para describir datos  
usando archivos de texto.

# Metalinguaje XML

GML --> SGML --> HTML --> XML



# Evolución HTML



# Metalinguaje XML

Lenguajes basados en XML:

- SVG
- FXML
- XUL
- ODF
- (XHTML) HTML5

# SVG Scalable Vector Graphics

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
<svg xmlns="http://www.w3.org/2000/svg" height="210" width="500">
```

```
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
```

```
  <polygon points="200,10 250,190 160,210" style="fill:lime;stroke:purple;stroke-width:1" />
```

```
  <text x="80" y="100" fill="black" font-size="50">Viva el SVG!</text>
```

```
</svg>
```



# FXML JavaFX Markup Language

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.layout.VBox?>
```

```
<?import javafx.scene.control.*?>
```

```
<VBox>
```

```
  <children>
```

```
    <Label text="Hello world FXML" />
```

```
    <Button fx:id="button" text="Click me again!" onAction="#buttonClicked"/>
```

```
  </children>
```

```
</VBox>
```



# **XUL** XML User Interface Language

```
<?xml version="1.0" encoding="iso-8859-1"?>  
<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"  
  id="hello"  width="300"  height="300" >  
  <description value="Hello the World!" />  
  <button label="Salir" />  
</window>
```

# ODF Open Document Format

```
<text:h text:style-name="Heading_2">Título</text:h>
```

```
<text:p text:style-name="Text_body" />
```

```
<text:p text:style-name="Text_body">
```

Este es un párrafo. La información sobre el formato se almacena en el archivo de estilo.

La marca vacía text:p que se ve más arriba es un párrafo en blanco (una línea vacía).

```
</text:p>
```

# ODF Open Document Format

```
<text:h text:style-name="Heading_2">Título</text:h>
```

```
<text:p text:style-name="Text_body" />
```

```
<text:p text:style-name="Text_body">
```

Este es un párrafo. La información sobre el formato se almacena en el archivo de estilo.

La marca vacía text:p que se ve más arriba es un párrafo en blanco (una línea vacía).

```
</text:p>
```

# HBM Hibernate Mapping

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd
```

# POM Project Object Model

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.paco</groupId>
  <artifactId>JavaFXReports</artifactId>
  <version>1.0-SNAPSHOT</version>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>
</project>
```

# XML genérico

```
<?xml version="1.0" encoding="uft-8"?>
```

```
<productos>
```

```
  <articulo>
```

```
    <nombre>Refresco 1LT</nombre>
```

```
    <precio>1.25</precio>
```

```
  </articulo>
```

```
  <articulo>
```

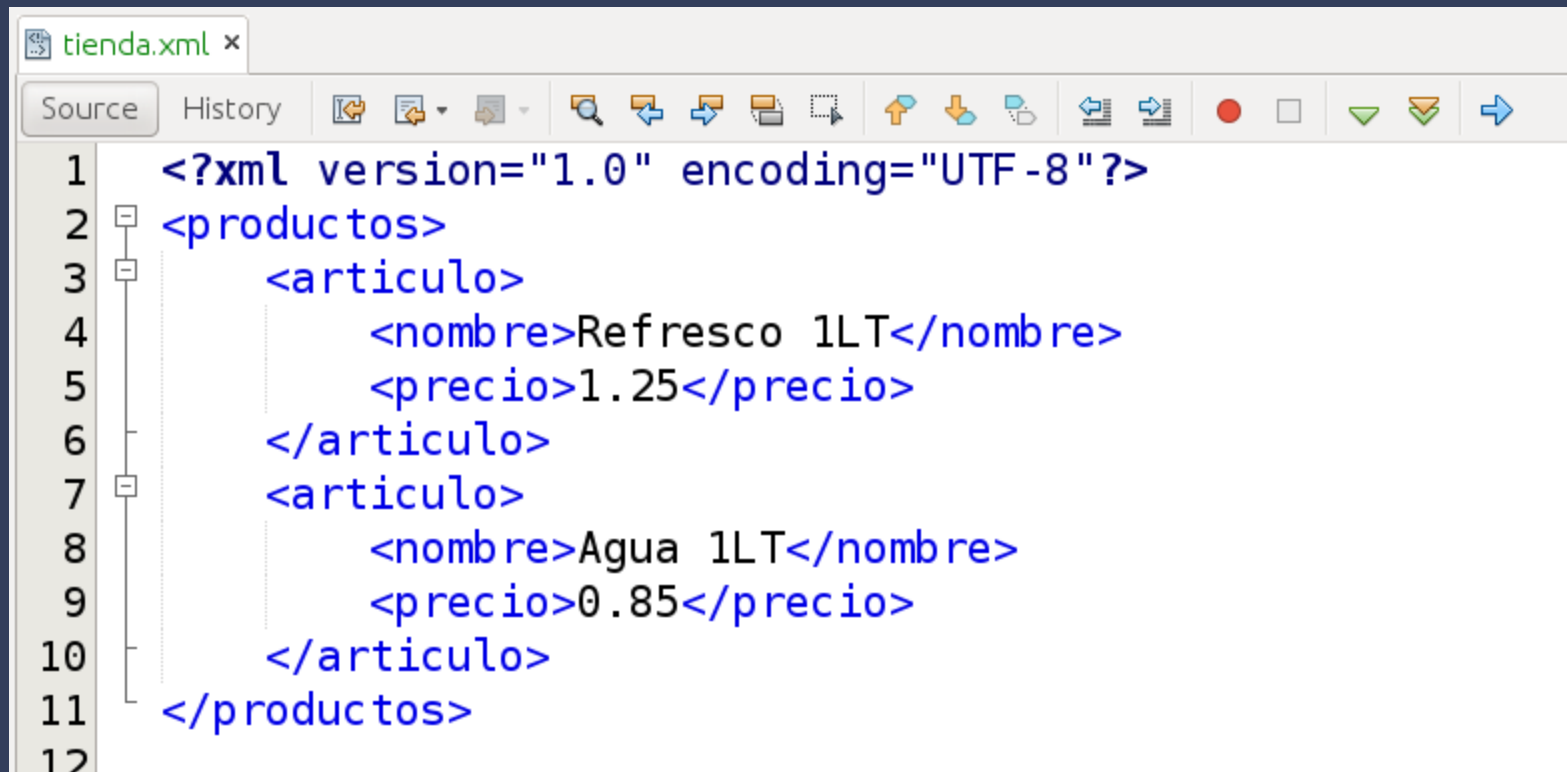
```
    <nombre>Agua 1LT</nombre>
```

```
    <precio>0.85</precio>
```

```
  </articulo>
```

```
</productos>
```

# XML genérico



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <productos>
3    <articulo>
4      <nombre>Refresco 1LT</nombre>
5      <precio>1.25</precio>
6    </articulo>
7    <articulo>
8      <nombre>Agua 1LT</nombre>
9      <precio>0.85</precio>
10   </articulo>
11 </productos>
12
```

The image shows a screenshot of an XML editor window titled 'tienda.xml'. The editor has a 'Source' tab selected and a toolbar with various icons. The XML code is displayed with line numbers from 1 to 12. The code defines an XML document with a root element 'productos' containing two 'articulo' elements. Each 'articulo' element has two child elements: 'nombre' and 'precio'. The first 'articulo' has 'nombre' 'Refresco 1LT' and 'precio' '1.25'. The second 'articulo' has 'nombre' 'Agua 1LT' and 'precio' '0.85'. The XML declaration at the top specifies version '1.0' and encoding 'UTF-8'.



# Cabecera XML

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

Atributos mas usuales:

- encoding: por defecto se usa UTF-8, pero también es posible usar codificaciones regionales.
- standalone: se usa para indicar si el documento hará referencias a otros documentos externos o no.

Otros elementos de la cabecera

**<?xml-stylesheet ref="simple-ie5.xsl" type="text/xsl"?>**

# Cabecera XML

- Debe estar en la primera línea del documento
- Debe contener obligatoriamente el número de versión
- Los parámetros y valores son sensibles a las mayúsculas
- Los parámetros van siempre en minúsculas
- El orden es fijo: versión, encoding y standalone.
- Las comillas pueden ser dobles o simples
- No existe etiqueta de cierre `</?xml>`

# Bloques básicos en XML

- Elementos --> etiquetas
- Atributos
- Entidades
- PCDATA (parsed character data)
- CDATA (character data)

# Etiquetado XML

Comentarios:

```
<!-- esto es un comentario -->
```

# Etiquetado XML

Etiquetas:

- `<nombre>Refresco 1LT</nombre>`

Atributos:

- `<nombre tipo="botella">Refresco 1LT</nombre>`

# Etiquetado XML

Contenido de los elementos:

- Vacío
- Datos
- Elementos XML

# Etiquetado XML

- Un elemento puede contener cualquier carácter alfanumérico. Solo se permiten como caracteres de puntuación "-", "\_", y ".".



# Etiquetado XML

- Para añadir contenido que no quiera ser parseado:

**<![CDATA[**

**esta sección no se procesa**

**<p>Por eso puede tener etiquetas</p>**

**]]>**

- Espacios en las etiquetas

**<address category = "residence">**

**<address        category    =       "       residence"       >**

# Etiquetado XML

- Queremos diseñar el formato de exportación de un blog online usando XML.
- Debe incluir para cada entrada:
  - Url de la entrada
  - Título
  - Contenido html
  - Fecha de publicación
  - Id

# Etiquetado XML

Entidades:

Entity	Entity Name	Notation
<	lt	&lt;
>	gt	&gt;
&	amp	&amp;
"	quot	&quot;
'	apos	&apos;

# Validación XML

- La validación es el proceso mediante el cual un documento XML (todos sus elementos) cumple con las especificaciones determinadas por un DTD (Document Type Declaration) o Scheme concreto.
- Se definen:
  - Documentos XML bien-formados
  - Documentos XML válidos

# Buenas prácticas

- Asigna nombres claros, cortos y consistentes (por ejemplo, todo en mayúsculas o minúsculas).
- Organiza los datos de forma lógica y anidada según su relación.
- Uso de atributos con moderación:
- Prioriza el uso de elementos para datos complejos y atributos para metadatos.
- Aplica sangría consistente para mejorar la legibilidad.
- Incluye comentarios claros sobre la finalidad de las estructuras.
- Elimina información duplicada para mantener el documento ligero y comprensible.

# Buenas prácticas con atributos

- Utiliza los atributos para almacenar datos que sean pequeños, simples y no estructurados (como identificadores o metadatos).
- Evitar datos complejos en atributos y evita indicar las unidades
- Asigna nombres breves, significativos y con un formato uniforme (por ejemplo, todo en minúsculas o camelCase).
- No abusar de los atributos, usa solo los necesarios
- No repitas información que ya está en otros atributos o elementos del XML.
- Usar valores bien definidos: define valores consistentes y controlados

# Datos complejos y metadatos

- Datos complejos → Estructura detallada (usualmente elementos).
- Metadatos → Información descriptiva y sencilla (usualmente atributos).

```
<Libro idioma="es" formato="digital">  
  <Titulo>Aprender XML</Titulo>  
  <Autor>Juan Pérez</Autor>  
  <Editorial>TechBooks</Editorial>  
</Libro>
```



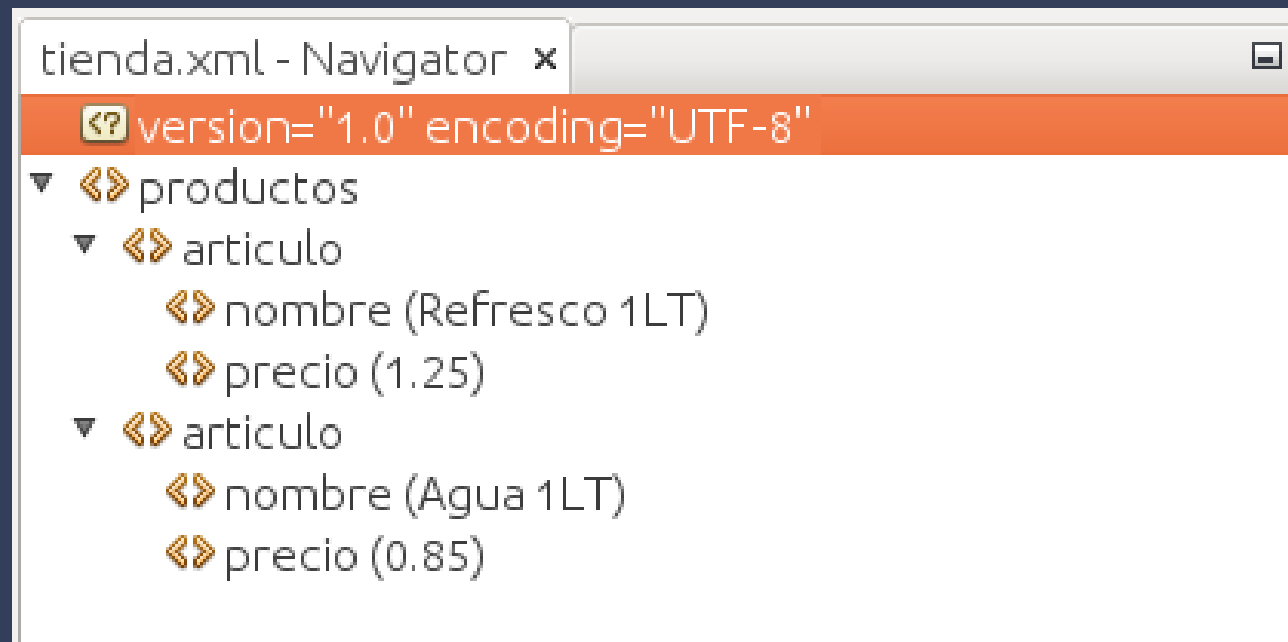
# Ejemplos típicos de metadatos

- Identificadores únicos
- Idiomas
- Formatos
- Fechas y horas
- Propiedades técnicas: tamaño o duración.
- Estados
- Versiones

# Árbol DOM

- Document Object Model (DOM) es el fundamento de un documento XML.
- Un árbol DOM es una colección de nodos de información organizados jerárquicamente.
- Esta estructura permite navegar y buscar fácilmente información en el árbol.

# Arbol DOM



# Nodos árbol DOM

Tipos de nodos:

- Raiz
- Intermedios
- Hoja

# Ejercicio de marcado

Queremos usar un documento XML para almacenar la información de contacto de todos los alumnos del centro (nombre, apellidos, teléfono y email)

Diseñar un documento de ejemplo que esté bien formado.

# Ejercicios de mercado

Pedido para el señor Francisco Romero. El pedido se compone de una Consola PlayStation 5. A entregar en la calle Huerta 3, tercer piso, letra B, el día 22-01-2024.

# Ejercicios de mercado

# Crear un documento XML con el horario de clase

[illegible]



# Ejercicios de mercado

- Crear un documento XML con los datos del trabajo de fin de grado

DATOS	
Ciclo:	Desarrollo de Aplicaciones Multiplataforma
Curso:	Segundo
Nombre alumno/a	Gerardo
Apellidos	Marín Jaime
PROYECTO INTEGRADO	
Título:	FacePi
Introducción:	FacePi es un proyecto basado en Raspberry Pi utilizando su capacidad de computación, modularidad y compatibilidad de software.
Finalidad:	Crear un dispositivo capaz de reconocer rostros.
Objetivos:	Adquisición del hardware. Instalación del SO. Puesta a punto de AWS Rekognition. Instalación y vinculación del software. Programación de scripts. Entrenamiento de la red neuronal. Testing. Conclusión del proyecto.
Medios utilizados:	Equipo de sobremesa con Windows 10. Raspberry Pi 3B Cámara
Planificación:	El proyecto se realizará por sprints semanales.
Bibliografía	

# Ejercicios de mercado

- Estamos creando una aplicación de escritorio para gestionar una tienda de electrodomésticos y queremos exportar el catálogo en un documento XML.
- Para cada electrodoméstico disponemos de: modelo, fabricante, consumo (A-F) y precio.
- Algunos tipos de electrodomésticos tienen datos específicos: lavadoras (peso), televisiones (tamaño, smartTV) y frigoríficos (estrellas, número de puertas).
- Crea el documento y añade a modo de ejemplo varios electrodomésticos diferentes.

# Ejercicios de mercado

- Crea un documento XML para almacenar la información de la petición de reserva de un formulario de la web de un restaurante.



## RESERVAS PARA GRUPOS SERVICIO GRATUITO

Obtén las mejores condiciones de precio, un trato directo y disfruta sin preocuparte de nada

Día de reserva:

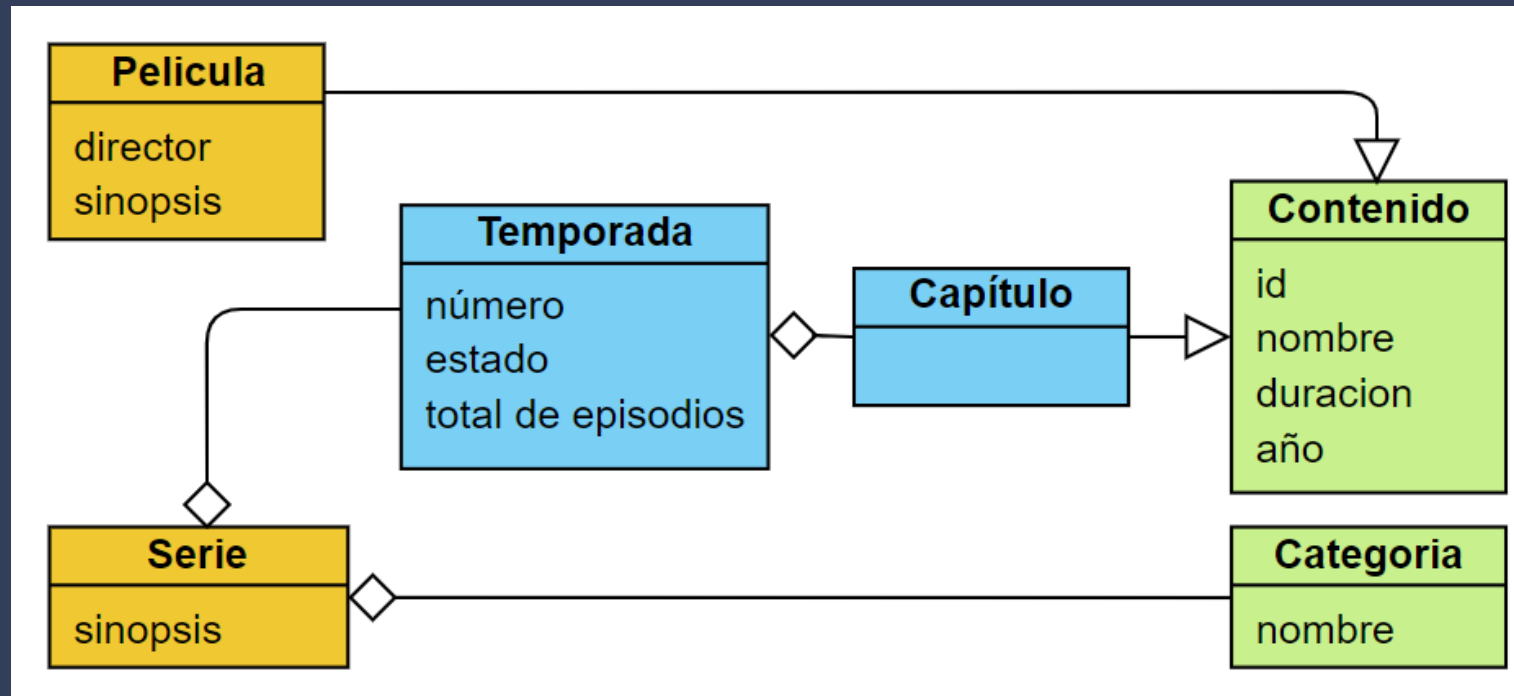
4 Personas ▼

Presupuesto/Persona

Comentario, quizá lo más importante

# Ejercicios de marcado

- Crea un documento XML para almacenar las series y películas de un programa para gestionar una biblioteca multimedia siguiendo este diagrama de clases. Añade 2 películas y 1 serie con dos temporadas y dos episodios cada una.



# Ejercicios de mercado

**film**affinity  
España

Iniciar sesión / Registrarse

Usuarios

Votar los tours

Iniciar sesión

Registrarse

España

Películas en cartelera

Cines España

Próximos estrenos

Estrenos Blu-ray venta

Próximos Blu-ray venta

Ya para alquilar

Próximamente alquiler

Video on Demand

Netflix

Netflix (próx)

Movistar +

Movistar + (próx)

HBO

Filmin

Rakuten TV

## Taquilla

### Taquilla España

Del 25 de diciembre al 27 de diciembre

Pos. ▴ ▾	Título	Género	Semanas ▴ ▾	Recaudación fin de semana ▴ ▾	Recaudación total ▴ ▾
1	<a href="#">Trolls 2: Gira mundial</a>	Animación	10	34.000 €	2.492.000 €
2	<a href="#">Otra vuelta de tuerca</a>	Terror	6	22.000 €	184.000 €

### Taquilla Estados Unidos

Del 25 de diciembre al 27 de diciembre

Pos. ▴ ▾	Título	Género	Semanas ▴ ▾	Recaudación fin de semana ▴ ▾	Recaudación total ▴ ▾
1	<a href="#">Wonder Woman 1984</a>	Fantástico	1	\$ 16.700.000	\$ 16.700.000
2	<a href="#">Noticias del gran mundo</a>	Western	1	\$ 2.250.000	\$ 2.250.000
3	<a href="#">Los Croods: Una nueva era</a>	Animación	5	\$ 1.745.000	\$ 30.362.000
4	<a href="#">Monster Hunter</a>	Fantástico	2	\$ 1.097.000	\$ 4.190.000

# Ejercicios de mercado

FACTURA n° 999					
Equipos Digitales S.L. Av. Valladolid Madrid 28015 C.I.F.: Q-9876543 teléfono: 91.777.66.88 fax: 91.777.66.99				Fecha: 12-01-2005 Pedido n° 731 Forma de pago: EFECTIVO	
<b>Datos CLIENTE</b>					
n° cliente: 879 Nombre: Darío Bueno Gutiérrez Dirección de envío: Av. Oporto n°7 4ºd Población: Madrid cod. postal: 28043 Provincia: Madrid					
<b>Datos FACTURA</b>					
REF.	DESCRIPCIÓN	CANT.	PRECIO.	I.V.A.	IMPORTE
MII93000F/8	MICRO PENTIUM IV 3000MHZ FB800	1	230 eur.	16,0	266,80 eur.
MB8QDIP4	PLACA BASE QDI P4	1	180 eur.	16,0	208,80 eur.
MEDD512M32	DIMM DDR 512MB 3200	2	40 eur.	16,0	92,80 eur.
HD250GSA7	DISCO DURO 250GB S-ATA 7200	4	120 eur.	16,0	556,80 eur.
<b>Base imponible</b>		<b>% I.V.A.</b>		<b>Cuota I.V.A.</b>	
970,00 eur.		16,0		155,20 eur.	
<b>TOTAL FACTURA: 1125,20 eur</b>					

# Ejercicios de mercado

No ↕	Área metropolitana ↕	2019 <sup>2</sup> ↕	2011 <sup>4</sup> ↕	Cambio en población (2011 a 2019) ↕
1	Madrid	6 120 254 <sup>2</sup>	6 052 247	▲ 68007
2	Barcelona	5 108 383 <sup>2</sup>	5 030 679	▲ 77704
3	Valencia	1 552 783 <sup>2</sup>	1 551 585	▲ 1198
4	Sevilla	1 307 209 <sup>2</sup>	1 294 867	▲ 12342
5	Málaga	974 822 <sup>2</sup>	953 251	▲ 21571
6	Bilbao	901 557 <sup>2</sup>	910 578	▼ -9021
7	Asturias	804 142 <sup>2</sup>	835 053	▼ -30911
8	Zaragoza	744 862 <sup>2</sup>	746 152	▼ -1290
9	Alicante-Elche	700 259 <sup>2</sup>	698 662	▲ 1597
10	Murcia	655 667 <sup>2</sup>	643 854	▲ 11813

# Ejercicios de mercado

*Platanus orientalis*

Nombre común: Platano

Vegetación: Caducifolio

Altura: De 20 a 25 metros

Forma y estructura: Copa ovoidal. Tronco principal recto. Ramaje expandido

Color en primavera: Haz verde medio, envés verde claro

Color en otoño: Ocre

Resistencia a las heladas: Heladas fuertes (hasta -20°C)

*Quercus ilex*

Nombre común: Encina

Vegetación: Perenne

Altura: En torno a 25 metros

Forma y estructura: Copa esférica o elíptica irregular. Tronco principal recto. Ramaje tortuoso

Color en primavera: Plateado en hojas jóvenes. En hojas antiguas, haz verde oscuro, envés plateado

Resistencia a las heladas: Heladas fuertes (hasta -15°C)



# Extraer el árbol DOM

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='yes'?>
<phonebook>
  <company>
    <cname>Microsoft</cname>
    <exchange>09-999000</exchange>
  </company>
  <president>Bill Gates
  <extension>09-9990011</president></extension>
  <secretary>Katharine Finch <extension>09-9990012</secretary></extension>
  <company>
    <cname>Oracle</cname>
    <exchange>09-888000</exchange>
  </company>
  <president>Larry Ellison
  <extension>09-8880011</president></extension>
  <secretary>Helen Calhoun
  <extension>09-8880012</secretary></extension>
</phonebook>
```

# Extraer el árbol DOM

```
<library>
<book id=2003-Blanken>
<title>Intelligent XML Search</title>
<editor>Henk A. Blanken et al.</editor>
<publisher>Springer Verlag</PUBLISHER>
</book>
<journal id="TODS_1_2003 publ_year="2003">
<title>ACM Transactions on Databases</title>
<volume>24<number>1</number></volume>
</journal>
<misc id="Dipl2002-12" id="TR2002-01-04">
<author>U. Known</author>
<title>The Diploma Thesis that didn't appear
<year>2002</year>
<pages>0</pages>
</misc>
</library>
```

# Ejercicios de mercado

Queremos desarrollar una aplicación móvil para las previsiones meteorológicas. En datos de aplicaciones móviles generalmente se transfiere en XML a través de HTTP a través de Internet a un servidor web.

Diseñar dos formatos XML:

- Un formato para la solicitud de pronóstico del tiempo. Debe contener la región y la fecha para donde se solicita la previsión meteorológica.
- Otro formato para que los servidores respondan a esta solicitud. Debe contener un breve descripción y una descripción larga del tiempo pronosticado. Agregue contenido que permita a la aplicación móvil añadir información meteorológica adicional, como el día y temperatura.

# Ejercicios de mercado

Queremos desarrollar una aplicación para gestionar una biblioteca. Internamente los datos se almacenarán en una base de datos, pero la aplicación dispondrá de opciones para realizar la exportación de toda la biblioteca en formato XML y el historial de préstamos.

Para organizar la biblioteca se utiliza la siguiente información:

- Ubicación: Sección y estantería
- Libro: Título, ISBN, idioma, autor/es, categorías, edición, año, editorial y resumen.
- Ejemplar: Estado, fecha de compra, listado de prestamos
- Préstamo: Usuario, fecha de préstamo, fecha de devolución.
- Usuario: Nombre, apellidos, fecha nacimiento, documento identificador.

Diseñar un formato XML para realizar la exportación e importación de los datos de la aplicación con 3 ejemplos completos.

# Elementos Vs Atributos

- Información importante en los elementos
- Información adicional (metada) en los atributos

```
<nota id="p501">  
  <para>Luis</para>  
  <de>Francisco</de>  
  <asunto>Tarea</asunto>  
  <contenido>No olvides estudiar XML</contenido>  
  </nota>
```

# Alternativas a XML: JSON

## JSON JavaScript Object Notation

Se trata de un estándar de comunicación abierto diseñado para ser legible y representar estructuras de datos y arrays. Se considera una alternativa al XML en la comunicación de información porque es mucho más legible y sencillo de utilizar.

# Alternativas a XML: JSON

## JSON JavaScript Object Notation

Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C

# Alternativas a XML: JSON

```
{  
  "Código":3141592654,  
  "Producto": "Jarabe de limón",  
  "Indicaciones":"Tomar cuando le duela la garganta",  
  "Contraindicaciones":"No tomar más de una cucharada cada  
tres horas",  
  "Tamaños":["250ml","500ml","1l"],  
  "Receta":false  
}
```



# Objetos y arrays en JSON

```
{  
  "empleados": [  
    {"nombre": "Pablo", "función": "currante"},  
    {"nombre": "Lucas", "función": "supervisor"},  
    {"nombre": "Andrés", "función": "currante"},  
    {"nombre": "Pérez", "función": "supervisor del supervisor"},  
    {"nombre": "Pablo", "función": "desconocida"}  
  ]  
}
```

# Objetos y arrays en JSON

```
<cesur>
  <alumno dni="859438504">
    <nombre>Jose</nombre>
    <apellido1>Perez</apellido1>
    <apellido2>Alselio</apellido2>
    <telefonos>
      <telefono tipo="movil">987654321</telefono>
      <telefono tipo="casa">987654325</telefono>
      <telefono tipo="trabajo">987654231</telefono>
    </telefonos>
    <mail><![CDATA[todojunto@lucas.com]]></mail>
    <ciclo curso="1">DAW</ciclo>
    <nacimiento>01/01/2021</nacimiento>
  </alumno>
</cesur>
```

# Conversión XML a JSON

```
<?xml version="1.0" encoding="uft-8"?>
```

```
<productos>
```

```
  <articulo>
```

```
    <nombre>Refresco 1LT</nombre>
```

```
    <precio>1.25</precio>
```

```
  </articulo>
```

```
  <articulo>
```

```
    <nombre>Agua 1LT</nombre>
```

```
    <precio>0.85</precio>
```

```
  </articulo>
```

```
</productos>
```

```
{  
  "productos": [  
    {  
      "nombre": "Refresco 1LT",  
      "precio": "1.25"  
    },  
    {  
      "nombre": "Agua 1LT",  
      "precio": "0.85"  
    }  
  ]  
}
```

# Ejercicio de JSON

Queremos diseñar una API para un programa de gestión de alumnos. El objetivo es que el programa cliente pueda hacer peticiones al servidor indicando una clase y un módulo, y el servidor responda un JSON con la siguiente información:

- Información general de la clase: ciclo, curso y tutor
- Información del módulo: nombre del módulo, profesor, número de aprobados y número de suspensos
- Listado de los nombres de todos los alumnos con la nota obtenida en cada trimestre.

# Errores comunes

<periodo\_liquidacion periodo="01/03/2011 a 31/03/2011">30 dias</periodo\_liquidacion>

<formProfesional porcentaje="0,10%">

<profesion GrupoDeCotizacion="1"> Ingeniero </profesion>

<devengo nombre = "salario base">1500€</devengo>

<empresa C.I.F="">

<estructurales unidad="horas">60</estructurales>

<Determinacion.Bases.cotizacion.SSSG>

<i:t></i:t>

# Errores comunes

```
<bccc>
```

```
  <incapTemporal></incapTemporal>
```

```
  <remTemporal>1692</remTemporal>
```

```
  <prorrataPE>250</prorrataPE>
```

```
  <total>1942</total>
```

```
</bccc>
```



# Tema 4.5

# Definición de esquemas y vocabularios: DTD

Lenguajes de Marcas y Sistemas de Gestión de la Información

# DTD: Document Type Declaration

- Es una forma de describir un lenguaje XML de forma concreta y precisa.
- En ellos se define el vocabulario y la estructura de un documento XML de manera que se pueda considerar como válido.
- Un DTD puede especificarse dentro del documento XML o de forma separada.



# DTD: Document Type Declaration

```
<!DOCTYPE direccion[  
  <!ELEMENT direccion (nombre,empresa,telefono)>  
  <!ELEMENT nombre (#PCDATA)>  
  <!ELEMENT empresa (#PCDATA)>  
  <!ELEMENT telefono (#PCDATA)>  
>
```

# Reglas DTD

- El DTD debe aparecer al inicio del documento, justo despues de la cabecera XML.
- Las declaraciones de los elementos deben empezar con !
- El nombre del DTD debe coincidir con el elemento raiz del documento.
- El DTD debe terminar siempre con ]>

# Elementos DTD

- `<!ELEMENT br EMPTY>`
- `<!ELEMENT from (#PCDATA)>`
- `<!ELEMENT note ANY>`
- `<!ELEMENT note (to,from,heading,body)>`
- `<!ELEMENT note (message ? + * )>`
- `<!ELEMENT note (#PCDATA|to|from|header|message)*>`

# Ejemplo XML con DTD

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE direccion[
  <!ELEMENT direccion(nombre,empresa,telefono)>
  <!ELEMENT nombre(#PCDATA)>
  <!ELEMENT empresa (#PCDATA)>
  <!ELEMENT telefono (#PCDATA)>
]>

<direccion>
  <nombre>Tanmay Patil</nombre>
  <empresa>TutorialsPoint</empresa>
  <telefono>(011) 123-4567</telefono>
</direccion>
```

# Ejemplo XML con DTD

```
<note>  
  <to>John</to>  
  <from>Jenny</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend</body>  
</note>
```

# Ejemplo XML con DTD

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE empresa[  
  <!ELEMENT empresa (nombre,cif) >  
  <!ELEMENT nombre (#PCDATA) >  
  <!ELEMENT cif (#PCDATA) >  

```

```
<empresa>  
  <nombre>CESUR</nombre>  
  <cif>893721893F</cif>  
</empresa>
```

# ¡Cuidado con el orden!

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE empresa[  
  <!ELEMENT empresa (nombre,cif) >  
  <!ELEMENT nombre (#PCDATA) >  
  <!ELEMENT cif (#PCDATA) >  

```

```
<empresa>  
  <cif>893721893F</cif>  
  <nombre>CESUR</nombre>  
</empresa>
```



# ¡Cuidado con el orden!

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE empresa[  
  <!ELEMENT empresa (nombre | cif)+ >  
  <!ELEMENT nombre (#PCDATA) >  
  <!ELEMENT cif (#PCDATA) >  

```

```
<empresa>  
  < cif>893721893F</cif>  
  < nombre>CESUR</nombre>  
</empresa>
```



# Ejemplo XML con DTD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE direccion [
  <!ELEMENT direccion (nombre,compañia,telefono)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT compania (#PCDATA)>
  <!ELEMENT telefono (#PCDATA)>
]>
<direccion>
  <nombre>Centro Málaga Este</nombre>
  <compañia>CESUR</compañia>
  <telefono>(011) 123-4567</telefono>
</direccion>
```

# Ejemplo XML con DTD

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```

# Ejemplo XML con DTD externo

```
<?xml version="1.0"? standalone="no">
<!DOCTYPE note SYSTEM "note.dtd">

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

## Note.dtd

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

# Ejercicio: Crear DTD desde XML (sin usar atributos)

```
<bebidas>
  <bebida>
    <codigo>L45</codigo>
    <precio>8</precio>
  </bebida>
  <bebida>
    <codigo/>
    <precio>6</precio>
  </bebida>
</bebidas>
```

# Ejercicio: Crear DTD desde XML (sin usar atributos)

```
<productos>
  <producto>
    <identificacion>
      Quedan 14 unidades.
      <codigo>MAR264</codigo>
    </identificacion>
    <nombre>martillo</nombre>
  </producto>
  <producto>
    <identificacion>
      No hay stock.
      <codigo>DES387</codigo>
      <id>678984</id>
    </identificacion>
    <nombre>destornillador</nombre>
  </producto>
</productos>
```

# Ejercicio: Crear XML desde DTD (sin usar atributos)

```
<!DOCTYPE NEWSPAPER [  
  <!ELEMENT NEWSPAPER (ARTICLE+)>  
  <!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>  
  <!ELEMENT HEADLINE (#PCDATA)>  
  <!ELEMENT BYLINE (#PCDATA)>  
  <!ELEMENT LEAD (#PCDATA)>  
  <!ELEMENT BODY (#PCDATA)>  
  <!ELEMENT NOTES (#PCDATA)>  

```

# Ejercicio: Crear DTD del ejercicio de FilmAffinity (sin usar atributos)

**film**affinity  
España

Iniciar sesión / Registrarse

Usuarios

Votar los tours

Iniciar sesión

Registrarse

España

Películas en cartelera

Cines España

Próximos estrenos

Estrenos Blu-ray venta

Próximos Blu-ray venta

Ya para alquilar

Próximamente alquiler

Video on Demand

Netflix

Netflix (próx)

Movistar +

Movistar + (próx)

HBO

Filmin

Rakuten TV

Taquilla

Taquilla España

Del 25 de diciembre al 27 de diciembre

Pos. ▴ ▾	Título	Género	Semanas ▴ ▾	Recaudación fin de semana ▴ ▾	Recaudación total ▴ ▾
1	<a href="#">Trolls 2: Gira mundial</a>	Animación	10	34.000 €	2.492.000 €
2	<a href="#">Otra vuelta de tuerca</a>	Terror	6	22.000 €	184.000 €

Taquilla Estados Unidos

Del 25 de diciembre al 27 de diciembre

Pos. ▴ ▾	Título	Género	Semanas ▴ ▾	Recaudación fin de semana ▴ ▾	Recaudación total ▴ ▾
1	<a href="#">Wonder Woman 1984</a>	Fantástico	1	\$ 16.700.000	\$ 16.700.000
2	<a href="#">Noticias del gran mundo</a>	Western	1	\$ 2.250.000	\$ 2.250.000
3	<a href="#">Los Croods: Una nueva era</a>	Animación	5	\$ 1.745.000	\$ 30.362.000
4	<a href="#">Monster Hunter</a>	Fantástico	2	\$ 1.097.000	\$ 4.190.000

# Ejercicio: Crear DTD (sin usar atributos)

Queremos definir un formato XML para almacenar la información de los alumnos y personal de CESUR.

Debe incluir:

- Información personal y de contacto de alumnos y profesores.
- Información docente de los alumnos (ciclo, fecha de matriculación y módulos que cursa en la actualidad)
- Información docente de los profesores: Ciclos y módulos que imparte cada profesor



# Atributos en DTD

**<!ATTLIST elemento atributo tipo valor>**

<!ATTLIST persona dni CDATA "00000X">

# Tipos de atributos en DTD

<b>CDATA</b>	Caracteres
<b>(en1 en2 ..)</b>	Selección de una lista
<b>ID</b>	Identificador único
<b>IDREF</b>	Identificador de otro elemento
<b>IDREFS</b>	Lista de identificadores
<b>NMTOKEN</b>	Caracteres válidos como nombre de elemento
<b>NMTOKENS</b>	El valor puede contener uno o varios valores de tipo NMTOKEN separados por espacios en blanco.

# Tipos de atributos en DTD

valor	Valor por defecto
#REQUIRED	Obligatorio
#IMPLIED	Opcional
#FIXED value	Valor fijo

# Atributos en DTD

**<!ELEMENT cuadrado EMPTY>**

**<!ATTLIST cuadrado lado CDATA "0">**

**<!ATTLIST persona id ID #REQUIRED>**

**<!ATTLIST pago tipo (tarjeta|metalico) "tarjeta">**

# Ejercicio DTD

Crear un DTD para definir una relacion de personas de las que tenemos relacion. La información de cada uno será: nombre, apellidos, emails y parentesco. Indicar como atributos el dni y la edad.

# Ejercicio para corregir

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cuadros [
<!ELEMENT cuadros (cuadro*)>
<!ELEMENT cuadro EMPTY>
<!ATTLIST cuadro titulo ID #REQUIRED>
<!ATTLIST cuadro autor CDATA #REQUIRED>
]>
```

```
<cuadros>
  <cuadro titulo="Adán y Eva" autor="Alberto Durero" />
  <cuadro autor="Lucas Cranach, el viejo" titulo="Adán y Eva"/>
</cuadros>
```

# Ejercicio: Crear DTD con atributos

Queremos definir un formato XML para almacenar la información de los alumnos y personal de CESUR.

Debe incluir:

- Información personal y de contacto de alumnos y profesores.
- Información docente de los alumnos (ciclo, fecha de matriculación y módulos que cursa en la actualidad)
- Información docente de los profesores: Ciclos y módulos que imparte cada profesor

# Ejercicio: Crear DTD

Queremos desarrollar una aplicación móvil para las previsiones meteorológicas. En aplicaciones móviles generalmente se transfiere en XML a través de HTTP a través de Internet a un servidor web.

Diseñar dos formatos XML:

- Un formato para la solicitud de pronóstico del tiempo. Debe contener la región y la fecha para donde se solicita la previsión meteorológica.
- Otro formato para que los servidores respondan a esta solicitud. Debe contener una breve descripción y una descripción larga del tiempo pronosticado. Agregue contenido que permita a la aplicación móvil añadir información meteorológica adicional, como el día y temperatura.



# Ejercicio: Crear DTD

Queremos desarrollar una aplicación para gestionar una biblioteca. Internamente los datos se almacenarán en una base de datos, pero la aplicación dispondrá de opciones para realizar la exportación de toda la biblioteca en formato XML y el historial de préstamos.

Para organizar la biblioteca se utiliza la siguiente información:

- Ubicación: Sección y estantería
- Libro: Título, ISBN, idioma, autor/es, categorías, edición, año, editorial y resumen.
- Ejemplar: Estado, fecha de compra, listado de prestamos
- Préstamo: Usuario, fecha de préstamo, fecha de devolución.
- Usuario: Nombre, apellidos, fecha nacimiento, documento identificador.

Diseñar un formato XML para realizar la exportación e importación de los datos de la aplicación con 3 ejemplos completos.

# Ejercicio DTD

Código	Diario	Origen	Destino	Hora salida	Hora llegada	Estado
V22	SI	New York	Chicago	9:30	11:30	R
V23	NO	New York	Miami	10:15	11:15	C

- Hay que guardar el **nombre** del aeropuerto, los datos de cada **vuelo** agrupados y la **fecha** del panel, en ese orden.
- En la DTD, sólo el **código** de un vuelo y su **estado** deben representarse mediante atributos.
- Se tiene que indicar que el **código** ha de ser único y obligatorio para cada vuelo.
- Los posibles **estados** de un vuelo son **C** (Cancelado), **E** (En hora), **R** (Retrasado). El valor por defecto debe ser **E**.
- En la DTD debe indicarse que al menos tiene que aparecer un **vuelo** y, para cada uno de ellos, se tiene que guardar la información en el mismo orden en el que aparece en el panel.
- Para indicar si un vuelo es **diario**, se debe utilizar un elemento vacío que, respecto a cada vuelo, podrá aparecer (en el caso de sí ser diario) o no aparecer (en el caso contrario).

# Fallos del ejercicio

<!ELEMENT h1 (strong)\*>

<!ELEMENT nav (ul)\*>

<!ELEMENT ul (li)\*>

<!ELEMENT li (#PCDATA)>

<!ELEMENT head (title)>

<!ELEMENT a (#PCDATA)>

<!ELEMENT a (#PCDATA | strong)\*>

<!ELEMENT body (h1\*, h2\*, h3\*, h4\*, p\*, ul\*, ol\*, hr\*,img\*)>