Unidad Didáctica 1. Introducción a los lenguajes de marcas

Temporalización: 6 sesiones

Nº sesión	Teoría	Ejercicios
1	1. Introducción 2. ¿Qué es un lenguaje de marcas? 3. Clasificación y propósitos	—
2	—	UD01_A01. Análisis de fragmentos (I) UD01_A02. Análisis de fragmentos (II)
3	4. XML vs JSON 5. Estructura y sintaxis de HTML, JSON y XML	UD01_A03. ¿Qué lenguaje usarías para...?
4	6. Lenguajes de marcas en el mundo real 7. Documentos bien formados	UD01_A04. Ejemplos en el mundo real UD01_A05. Detección errores UD01_A06. Práctica final
5	Ampliación: namespaces	UD01_A06. Práctica final
6	Repaso general	Examen tipo test

Resultado de aprendizaje:

RA1. Reconoce las características de lenguajes de marcas analizando e interpretando fragmentos de código.

Criterios de evaluación asociados:

- ◯ **RA1.CE1a**. Se han identificado las características generales de los lenguajes de marcas.
- ◯ **RA1.CE1b**. Se han reconocido las ventajas que proporcionan en el tratamiento de la información.
- ◯ **RA1.CE1c**. Se han clasificado los lenguajes de marcas e identificado los más relevantes.

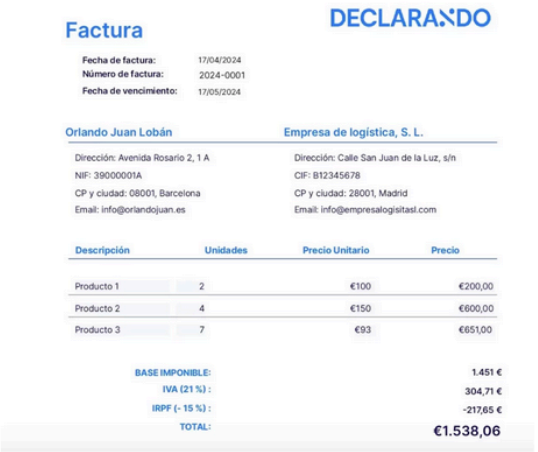
- ◯ **RA1.CE1d**. Se han diferenciado sus ámbitos de aplicación.
- ◯ **RA1.CE1e**. Se han reconocido la necesidad y los ámbitos específicos de aplicación de un lenguaje de marcas de propósito general.
- ◯ **RA1.CE1f**. Se han analizado las características propias de diferentes lenguajes de marcas.
- ◯ **RA1.CE1g**. Se ha identificado la estructura de un documento y sus reglas sintácticas.
- ◯ **RA1.CE1h**. Se ha contrastado la necesidad de crear documentos bien formados y la influencia en su procesamiento.
- ◯ **RA1.CE1i**. Se han identificado las ventajas que aportan los espacios de nombres.

Índice

---

1. Introducción

¿Qué tienen en común una página web, una factura electrónica y una app meteorológica?



Todas estas situaciones (una página web, una factura electrónica, una app del tiempo...) implican lo mismo: la **transmisión estructurada de información**. Y para lograrlo, todos **dependen de los lenguajes de marcas**.

Los lenguajes de marcas **permiten**:

- **Describir** la información (qué representa).
- **Estructurarla** para que sea comprensible.
- **Transmitirla** entre personas o entre sistemas informáticos.
- **Automatizar** su tratamiento (leer, transformar, visualizar, almacenar).

Son **esenciales** para el **intercambio de datos** en el mundo digital.

---

2. ¿Qué es un lenguaje de marcas?

Un **lenguaje de marcas** es un sistema de **codificación** que permite estructurar, describir y presentar información.

Características generales:

- Utiliza **etiquetas** para marcar partes del contenido.
- Tiene una **estructura jerárquica** (anidamiento).
- Separa el **contenido** de su **presentación** visual o lógica.
- Es **legible** tanto por **humanos** como por **máquinas**, independientemente del contexto (móvil, web, escritorio, datos).

Factura electrónica: caso real de uso de XML

Una **factura electrónica** no es simplemente un **PDF**. Es un **documento estructurado** que contiene la misma información que una factura tradicional, pero en un formato estándar y legible por ordenadores.

En España y en la Unión Europea, el formato más habitual es: **Facturae** (formato XML). Su finalidad es ser enviado a organismos públicos o empresas y leído automáticamente por programas de gestión.

Ejemplo (simplificado):

```
<Factura>
  <Emisor>
    <Nombre>Distribuciones Pérez</Nombre>
    <NIF>B12345678</NIF>
  </Emisor>
  <Receptor>
    <Nombre>Ayuntamiento de Valencia</Nombre>
    <NIF>Q2816002D</NIF>
  </Receptor>
  <Total>452.60</Total>
</Factura>
```

---

3. Clasificación y propósitos de los lenguajes de marcas

Loslenguajes de marcas se clasifican en los siguientes **tipos**:

Tipo	Ejemplo	Uso principal
Propósito general	XML	Intercambio estructurado entre sistemas
Propósito específico web	HTML	Estructura y presentación web
Datos / intercambio	JSON	Envío y recepción de datos (APIs)

Fragmentos de código representativos:

- HTML

```
<h1>Bienvenidos</h1>
```

- XML

```
<libro><titulo>1984</titulo></libro>
```

- JSON

```
{ "titulo": "1984" }
```

!!! note "¿Qué tienen en común estos fragmentos? ¿Qué diferencias observas?"

📝 1. UD01_A01. Análisis de fragmentos (I)

HTML
→ Diseñado para **presentar contenido** en la web.
Ej.: páginas web, estructuras de documentos visuales.
Contiene enlaces, encabezados, listas, párrafos...

JSON
→ Diseñado para el **intercambio de datos** entre aplicaciones.
Ej.: comunicación entre frontend y backend, APIs, apps móviles.
Representa objetos, arrays, estructuras anidadas...

XML
→ Diseñado como lenguaje de marcas de **propósito general**.
Ej.: facturas electrónicas, configuraciones, interoperabilidad.
Define una estructura personalizable, con posibilidad de validación.

3.1. ¿Cuándo usar cada uno?

Situación	Lenguaje más adecuado	¿Por qué?
Crear una página web	HTML	Se interpreta directamente por navegadores
Enviar datos de una app móvil al servidor	JSON	Ligero, fácil de procesar por JavaScript
Generar una factura estructurada para Hacienda	XML	Estandarizado, validable, interoperable
Consultar el tiempo en una app	JSON	Usado en APIs de datos meteorológicos
Crear un RSS para suscribirse a noticias	XML	Requiere estructura, estándar formal
Mostrar un artículo con texto, imágenes y vídeo	HTML	Orientado a presentación multimedia

¿Qué lenguaje usarías para...?

1. Mostrar una receta de cocina en una página web
2. Sincronizar datos entre dos apps móviles
3. Guardar configuraciones de una app para abrirlas más tarde

4. Publicar un catálogo online de películas con fichas detalladas
5. Enviar un formulario desde una web a un servidor

📝 **UD02_A02. Análisis de fragmentos (II)**

4. XML vs JSON

Aspecto	XML	JSON
Origen	1998 – W3C	2001 – Derivado de JavaScript
Sintaxis	Verbosa, basada en etiquetas	Más ligera, basada en pares clave-valor
Estructura jerárquica	Sí (etiquetas anidadas)	Sí (objetos anidados y arrays)
Legibilidad para humanos	Media	Alta
Facilidad de procesamiento	Requiere parser XML	Muy fácil en lenguajes modernos (JavaScript, Python...)
Uso actual	Documentación formal, estándares, interoperabilidad	APIs, aplicaciones web, apps móviles
Validación con esquemas	Muy robusta (XSD, DTD)	Limitada (JSON Schema opcional)
Soporte para firma digital	Alto (XMLDSig)	Bajo o no estándar
Tamaño de los archivos	Mayor	Menor

En resumen:

- XML se sigue usando en ámbitos formales o normativos (administración, facturación, estándares industriales).
- JSON es el formato más utilizado hoy en día para el intercambio de datos entre aplicaciones modernas.

5. Estructura y sintaxis de HTML, JSON y XML

5.1. Anatomía de un documento HTML5

Partes de un documento HTML5. Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Mi primera página</title>
  </head>
  <body>
    <h1>Hola mundo</h1>
    <p>Esta es una web sencilla.</p>
  </body>
</html>
```

Puntos a destacar:

- Apertura y cierre de **etiquetas**
- **Estructura jerárquica**
- **Separación** entre estructura (), presentación () y configuración ()

5.2. Estructura de un objeto JSON

JSON representa datos como **objetos** o **arrays**.

Características clave:

- Basado en pares **clave:valor**
- Utiliza **llaves** {} para objetos, **corchetes** [] para arrays
- **No usa etiquetas**, sino comillas y signos de puntuación

Ejemplo:

```
{
  "titulo": "Batman Begins",
  "año": 2005,
  "géneros": ["acción", "superhéroes"],
  "director": {
    "nombre": "Christopher Nolan",
    "edad": 55
  }
}
```

Puntos a destacar:

- Las **claves** siempre entre **comillas**
- Los **valores** pueden ser texto, número, booleano, array u objeto
- **Muy utilizado** en aplicaciones web y móviles

5.3. Sintaxis básica de XML

XML estructura datos mediante **etiquetas personalizadas**.

Características:

- Se definen **etiquetas personalizadas** según el contexto
- Requiere una **estructura bien formada** (cada apertura debe cerrarse)
- Puede tener **atributos** y anidamiento complejo

Ejemplo:

```xml
<videojuego plataforma="PS5">
  <titulo>Spider-Man 2</titulo>
  <genero>Acción</genero>
  <precio>59.99</precio>
</videojuego>
```

Puntos a destacar:

- Las etiquetas tienen **nombre libre** pero coherente
- Se pueden usar **atributos** (como plataforma="PS5")
- **No admite errores** de anidamiento o cierre

!!! warning "Etiqueta VS atributo"

```
En XML, puedes representar la misma información de varias formas.

**Opción A** – plataforma como atributo:

```xml
<videojuego plataforma="PS5">
  <titulo>Spider-Man 2</titulo>
  <genero>Acción</genero>
  <precio>59.99</precio>
</videojuego>
```
```

Opción B – plataforma como subelemento:

```xml
<videojuego>
  <plataforma>PS5</plataforma>
  <titulo>Spider-Man 2</titulo>
  <genero>Acción</genero>
  <precio>59.99</precio>
</videojuego>
```

Ambas son sintácticamente válidas. La diferencia está en lo semántico.

En nuestro ejemplo:

- plataforma="PS5" es una información que **clasifica** el videojuego, que no necesita más estructura y no se espera que tenga subcampos (por ejemplo: fabricante, modelo, año…).

- Por eso, es habitual representarlo como **atributo**.

- Además, los atributos **no pueden tener hijos**, solo valores simples (texto sin etiquetas internas).

Pero si quisiéramos ampliar la plataforma, lo cambiaríamos a subelemento:

```xml
<plataforma>
  <nombre>PS5</nombre>
  <fabricante>Sony</fabricante>
</plataforma>
```

📝 **UD01_A03. ¿Qué lenguaje usarias para...?**

---

6. Lenguajes de marcas en el mundo real

6.1. XML en RSS, SVG y XSL

Contenido teórico:

- **RSS** (Really Simple Syndication): feed de noticias basado en XML.
- **SVG** (Scalable Vector Graphics): imágenes vectoriales definidas con XML.
- **XSL/XSLT** (Extensible Stylesheet Language): transformación de XML a otros formatos (HTML, texto,...).

Ejemplo breve de RSS:

```xml
<rss version="2.0">
  <channel>
    <title>Noticias DAM</title>
    <item>
      <title>Convocatoria examen</title>
      <link>https://centro.edu/examen</link>
      <pubDate>Thu, 10 Jul 2025 12:00:00 +0200</pubDate>
    </item>
  </channel>
</rss>
```

6.2. JSON en APIs y Open Data

Contenido teórico:

- **APIs RESTful:** intercambio de datos con JSON.
- **Portales Open Data:** catálogos de datos públicos en JSON.

Ejemplo breve de respuesta API (curl en consola):

```
curl -s https://api.ejemplo.com/peliculas/123 | jq .
```

```
{
  "id": 123,
  "titulo": "Amélie",
  "director": "Jean-Pierre Jeunet",
  "año": 2001
}
```

6.3. HTML como base de páginas web

Contenido teórico:

- **Estructura semántica** con etiquetas `<header>`, `<nav>`, `<main>`, `<footer>`.
- **Accesibilidad** y SEO: uso correcto de `<h1>…<h6>`, `<alt>` en imágenes.

Ejemplo de fragmento real de GitHub Pages:

```
<header>
  <h1>Mi portafolio</h1>
  <nav>
    <ul>
      <li><a href="#sobre-mi">Sobre mí</a></li>
      <li><a href="#proyectos">Proyectos</a></li>
    </ul>
  </nav>
</header>
```

📝 **UD01_A04. Ejemplos lenguajes de marcas en el mundo real**

7. Documentos bien formados vs mal formados

7.1. Documento bien formado

¿Qué significa que un documento esté "bien formado"?

Un documento está bien formado cuando **respeta las reglas sintácticas** del lenguaje.
Si no lo hace, no podrá ser **interpretado** por programas o navegadores.

En **XML**:

- Todas las **etiquetas** deben **cerrarse**.
- Las **etiquetas** deben estar correctamente **anidadas**.
- Solo puede haber un **único** elemento **raíz**.
- Los **atributos** deben estar entre **comillas**.

En **JSON**:

- Las **claves** deben estar entre **comillas** dobles.
- Las **comas** deben colocarse correctamente (sin dejar una al final).
- **No** puede haber **comentarios**.
- Las **estructuras** deben estar **cerradas** correctamente con } o].

Ejemplo de XML bien vs mal formado

❌ Mal formado:

```
<persona>
  <nombre>Elena</nombre>
  <edad>34
</persona>
```

Errores:

- Falta etiqueta de cierre para
- El contenido no se puede procesar automáticamente

✅ Bien formado:

```
<persona>
  <nombre>Elena</nombre>
  <edad>34</edad>
</persona>
```

—

Ejemplo de JSON bien vs mal formado

❌ Mal formado:

```
{
  "usuario": "paco",
  "edad": 28,
  "activo": true
}
```

Errores:

- Coma final después del último elemento → no permitida

✅ Bien formado:

```
{
  "usuario": "paco",
  "edad": 28,
  "activo": true
}
```

7.2. ¿Por qué es tan importante?

Los lenguajes de marcas suelen ser **procesados por máquinas**. Si un documento está mal formado:

- No puede ser leído o procesado
- Se produce un error o fallo en la carga
- Puede corromper procesos automáticos (facturación, apps, APIs...)

En la **web**, una etiqueta mal cerrada puede romper toda la página.
En una **factura electrónica**, un XML mal formado la invalida para su envío o firma.

📝 **UD01_A05. Detección errores fragmentos XML y JSON**

7.3. Validadores

- Para XML → www.xmlvalidation.com
- Para JSON → www.jsonlint.com
- En VSCode: puedes instalar extensiones que te alertan de errores automáticamente
- En XML avanzado (UD4): veremos cómo validar usando un "modelo" (XSD)

Ampliación – Introducción a los espacios de nombre (namespaces) en XML

¿Qué son los nombres de espacio en XML?

Los **espacios de nombre (namespaces)** son un mecanismo que permite evitar **conflictos de nombres** cuando un documento XML mezcla etiquetas que pueden tener el mismo nombre pero distinto significado o procedencia.

Por ejemplo, imagina que tienes dos vocabularios XML diferentes: uno para datos bibliográficos y otro para datos técnicos, ambos usan la etiqueta `<titulo>`, pero con significados distintos. Sin namespaces, el parser no puede diferenciar entre ambos.

¿Cómo funcionan?

Un namespace se define asociando un **prefijo** con una **URI** (Identificador Uniforme de Recursos, que actúa como un identificador único, no tiene que ser una URL real). Luego se usan esos prefijos para calificar las etiquetas.

Ejemplo simple sin namespace (problema):

```xml
<pelicula>
  <titulo>Matrix</titulo>
  <titulo>Título técnico</titulo>
</pelicula>
```

Aquí hay dos `<titulo>`, pero no sabemos cuál es cuál.

Ejemplo con namespaces (solución):

```xml
<pelicula xmlns:info="http://www.ejemplo.com/info"
xmlns:tecn="http://www.ejemplo.com/tecnico">
  <info:titulo>Matrix</info:titulo>
  <tecn:titulo>Título técnico</tecn:titulo>
</pelicula>
```

- `xmlns:info="..."` define el espacio de nombres con prefijo `info`.
- `xmlns:tecn="..."` define otro espacio con prefijo `tecn`.
- Cada etiqueta `<info:titulo>` y `<tecn:titulo>` pertenece a un espacio distinto.

Ventajas

- Evita ambigüedades al combinar vocabularios o esquemas diferentes.
- Permite que un documento sea extensible y interoperable.
- Es fundamental en estándares XML complejos (SOAP, XHTML, SVG, etc.).

¿Cuándo usar namespaces?

- Cuando trabajas con documentos XML que combinan varios esquemas o estándares.
- Cuando necesitas evitar colisiones en nombres de etiquetas.
- En documentos simples, no siempre es obligatorio, pero es buena práctica conocerlo.

¿Cómo se declara un namespace?

- Se declara en la etiqueta raíz (o en cualquier etiqueta) con `xmlns:prefijo="URI"`
- Luego se usa el prefijo para calificar las etiquetas que pertenecen a ese espacio.

Ejemplo extendido de un fragmento con namespace:

```
<peliculas xmlns:info="http://www.ejemplo.com/info"
xmlns:desc="http://www.ejemplo.com/desc">
  <pelicula>
    <info:titulo>Matrix</info:titulo>
    <desc:sinopsis>Una realidad simulada...</desc:sinopsis>
  </pelicula>
</peliculas>
```

Wikipedia: Espacio de nombres XML

📝 Tareas Unidad Didáctica 1: Introducción a los lenguajes de marcas

Índice

📝 UD01_A01. Análisis de fragmentos (I)

Actividad: Analiza fragmentos de código y rellena la siguiente tabla.

Fragmento de código	¿Qué lenguaje es?	¿Qué representa?	¿Qué ventajas ves?	¿Dónde lo usarías?
<title>Batman</title>				
{ "autor": "Orwell" }				
<alumno> <nombre>Ana</nombre> </alumno>				

📝UD01_A02. Análisis de fragmentos (II)

Por cada uno de los siguientes fragmentos, contesta las siguientes preguntas:

1. ¿Qué representa el documento?
2. ¿Cómo está estructurado?
3. ¿Qué elementos o símbolos llaman la atención?

Fragmento 1 – HTML

```
<body>
  <h2>Contacto</h2>
  <ul>
    <li>Email: info@ejemplo.com</li>
    <li>Teléfono: 123456789</li>
  </ul>
</body>
```

Fragmento 2 – JSON

```
{
  "usuario": "juan87",
  "activo": true,
  "rol": "editor"
}
```

Fragmento 3 – XML

```
<pedido>
  <producto>Ratón inalámbrico</producto>
  <cantidad>2</cantidad>
</pedido>
```

📝UD01_A03. ¿Qué lenguaje usarías para...?

- Indicar qué lenguaje usaría en cada caso (HTML, XML, JSON).
- Justificar brevemente por qué ese lenguaje es el adecuado.

Nº	Situación	Lenguaje	Justificación breve
1	Un videojuego guarda tu partida y luego la recupera		
2	Una empresa factura a otra empresa pública		

Nº	Situación	Lenguaje	Justificación breve
3	Una web muestra noticias con titulares, imágenes y enlaces		
4	Una app meteorológica consulta la temperatura y la lluvia esperada para mañana		
5	Una app necesita almacenar preferencias del usuario entre sesiones		
6	Una API de películas devuelve título, director, año, géneros y puntuación		
7	Una administración pública crea un modelo de documento interoperable		

📝UD01_A04. Ejemplos de lenguajes de marcas en el mundo real

Instrucciones:

1. **HTML real**: Abrir la consola de desarrollador en una web (ej. https://ejemplo.com) y localizar la sección `<nav>` o el `<header>`.
2. **JSON en consola**: Ejecutar un `curl` o usar las DevTools → pestaña "Network" para ver una respuesta JSON de una API pública (p. ej. https://pokeapi.co).
3. **XML en un feed**: Suscribirse al RSS de un blog (p. ej. https://blog.ejemplo.com/rss) y abrirlo en un navegador o editor.

📝UD01_A05. Detectar y corregir errores en fragmentos XML y JSON

1. Leer el fragmento
2. Detectar errores sintácticos
3. Reescribir el fragmento corregido
4. Justificar el error y su corrección

Ejemplo de actividad:

XML con errores:

```
<curso>
  <nombre>LMISGI</nombre>
  <horas>70
  <modulo>0373</modulo>
</curso>
```

Corrección esperada:

```
<curso>
  <nombre>LMISGI</nombre>
  <horas>70</horas>
  <modulo>0373</modulo>
</curso>
```

📝UD01_A06. Práctica final

Enunciado de la actividad:

A partir del catálogo de películas facilitado por el profesor realizar la siguiente tarea.

Representar esa información con:

- **HTML**, orientado a la visualización.
- **XML**, como representación estructurada.
- **JSON**, para intercambio de datos..

Ejemplo simplificado HTML:

```
<article>
  <h2>Matrix</h2>
  <p><strong>Director:</strong> Wachowski</p>
  <p><strong>Año:</strong> 1999</p>
  <p><strong>Género:</strong> Ciencia ficción</p>
</article>
```

Ejemplo simplificado JSON:

```
{
  "pelicula": {
    "titulo": "Matrix",
    "director": "Wachowski",
    "año": 1999,
    "genero": "Ciencia ficción"
  }
}
```

Ejemplo simplificado XML:

```xml
<pelicula>
  <titulo>Matrix</titulo>
  <director>Wachowski</director>
  <año>1999</año>
  <genero>Ciencia ficción</genero>
</pelicula>
```

Material proporcionado:

- Documento de texto con la descripción de 3-4 películas.
- Guía de ejemplo para cada formato.
- Plantilla HTML vacía, archivo `.json` y `.xml` en blanco.

Indicaciones:

- Cuidar la estructura.
- Validar cada formato con herramientas en línea:
  - HTML: validator.w3.org
  - JSON: jsonlint.com
  - XML: xmlvalidation.com

Documentos para la práctica:

- catalogo_peliculas.txt
- ejemplo.json
- ejemplo.xml
- ejemplo.html

Recordatoios:

```
HTML:
- Usa etiquetas para mostrar contenido visualmente.
- Cualquier dato debe ir dentro del <body>.
- Usa <article> o <section> para separar cada película.

JSON:
- Es sensible a las comillas y comas.
```

```
- La clave debe ir entre comillas dobles.
- Valida tu JSON en https://jsonlint.com/

XML:
- Las etiquetas deben abrirse y cerrarse correctamente.
- El archivo debe tener una única raíz.
- Valida tu XML en https://www.xmlvalidation.com/
```
