XML

La evolución de los datos

- **Binario**.

La evolución de los datos

- **Binario**.
  - Lo mas simple en datos.
  - Eficientes y fácil de procesar para una máquina.
  - Imposible de entender para una persona.

La evolución de los datos

- **Binario**.
  - Lo mas simple en datos.
  - Eficientes y fácil de procesar para una máquina.
  - Imposible de entender para una persona.
- ¿Cuál es la desventaja ?

La evolución de los datos

- **Binario**.
  - Lo mas simple en datos.
  - Eficientes y fácil de procesar para una máquina.
  - Imposible de entender para una persona.
- ¿Cuál es la desventaja ?

La estructura es propietaria, depende de la máquina. Todo lo que se ve en el archivo depende de la máquina.

La evolución de los datos

- **Archivos de texto**.

La evolución de los datos

- **Archivos de texto**.
  - Lo más legible en datos.
  - Eficiente y legible para una persona.
  - Fácil para el intercambio de información.

La evolución de los datos

- **Archivos de texto**.
  - Lo más legible en datos.
  - Eficiente y legible para una persona.
  - Fácil para el intercambio de información.
- ¿Cuál es la desventaja ?

La evolución de los datos

- **Archivos de texto**.
  - Lo más legible en datos.
  - Eficiente y legible para una persona.
  - Fácil para el intercambio de información.
- ¿Cuál es la desventaja ?

No tiene metadatos, es decir, el computador no sabe lo que hay.

La evolución de los datos

- **Bases de datos relacionales**.

La evolución de los datos

- **Bases de datos relacionales**.
  - Lo más estructurado en datos.
  - Fácil de procesar para un computador.
  - Legible por una persona.

La evolución de los datos

- **Bases de datos relacionales**.
  - Lo más estructurado en datos.
  - Fácil de procesar para un computador.
  - Legible por una persona.
- ¿ Cual es la desventaja ?

La evolución de los datos

- **Bases de datos relacionales**.
  - Lo más estructurado en datos.
  - Fácil de procesar para un computador.
  - Legible por una persona.
- ¿ Cual es la desventaja ?

Muy estructurados.

¿ Cuál es el próximo paso ?

¿ Cuál es el próximo paso ?

Necesitamos un formato de datos que tenga:

- Metadatos – estructura legible para el computador.
- Universalidad – que no dependa de la máquina.
- Extensibilidad – que pueda modificar la estructura más fácilmente.

¿ Cuál es el próximo paso ?

Necesitamos un formato de datos que tenga:

- Metadatos – estructura legible para el computador.
- Universalidad – que no dependa de la máquina.
- Extensibilidad – que pueda modificar la estructura más fácilmente.

Necesitamos XML.

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

XML: Un ejemplo

```
<equipos_de_futbol>

   <equipo nombre="Universidad de Chile" >
      <estadio> Estadio Nacional </estadio>
      <entrenador> Sebastián Beccacece </entrenador>
      <jugadores>
         <jugador> Gonzalo Jara </jugador>
         <jugador> Jhonny Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo nombre="Universidad Catolica" ranking="1">
      <entrenador> Mario Salas </entrenador>
      <jugadores> </jugadores>
   </equipo>

</equipos_de_futbol>
```

Cómo se construye un documento XML

Cómo se construye un documento XML

- Sintaxis (simplificada):

```
documento := <tag> nodo </tag>

nodo := nodo nodo | <tag> nodo </tag> | <tag/> |
        PCDATA | ϵ
```

Cómo se construye un documento XML

- Sintaxis (simplificada):

```
documento := <tag> nodo </tag>

nodo := nodo nodo | <tag> nodo </tag> | <tag/> |
        PCDATA | ϵ
```

- Un documento que cumple con esta sintaxis es un:

Documento bien formado.

Documentos bien formados

Un documento bien formado debe tener:

- un nodo **raíz**.
- cada tag debe **cerrarse**.
- los tags son **case sensitive**.
- tags deben estar **correctamente anidados**.

Documentos bien formados

Un documento bien formado debe tener:

- un nodo **raíz**.
- cada tag debe **cerrarse**.
- los tags son **case sensitive**.
- tags deben estar **correctamente anidados**.

¿ Cómo validar si un documento esta bien formado ?

Documento mal formado

Documento mal formado

```
<equipo>
      <nombre> Universidad de Chile
      <goles> 2 <tarjetas> 1 </goles> </tarjetas>
      <Entrenador> Beccacece </entrenador>
</equipo>

<equipo>
      <nombre> Colo Colo
      <goles> 0 <tarjetas> 3 </goles> <tarjetas>
      <Entrenador> ? </entrenador>
</equipo
```

¿ Por qué documentos bien formados ?

¿ Por qué documentos bien formados ?

- Es un requisito del estandar de XML.

¿ Por qué documentos bien formados ?

- Es un requisito del estandar de XML.
- Y si el documento esta bien formado:

Entonces puede ser modelado como un arbol ordenado.

Arboles: Documento XML

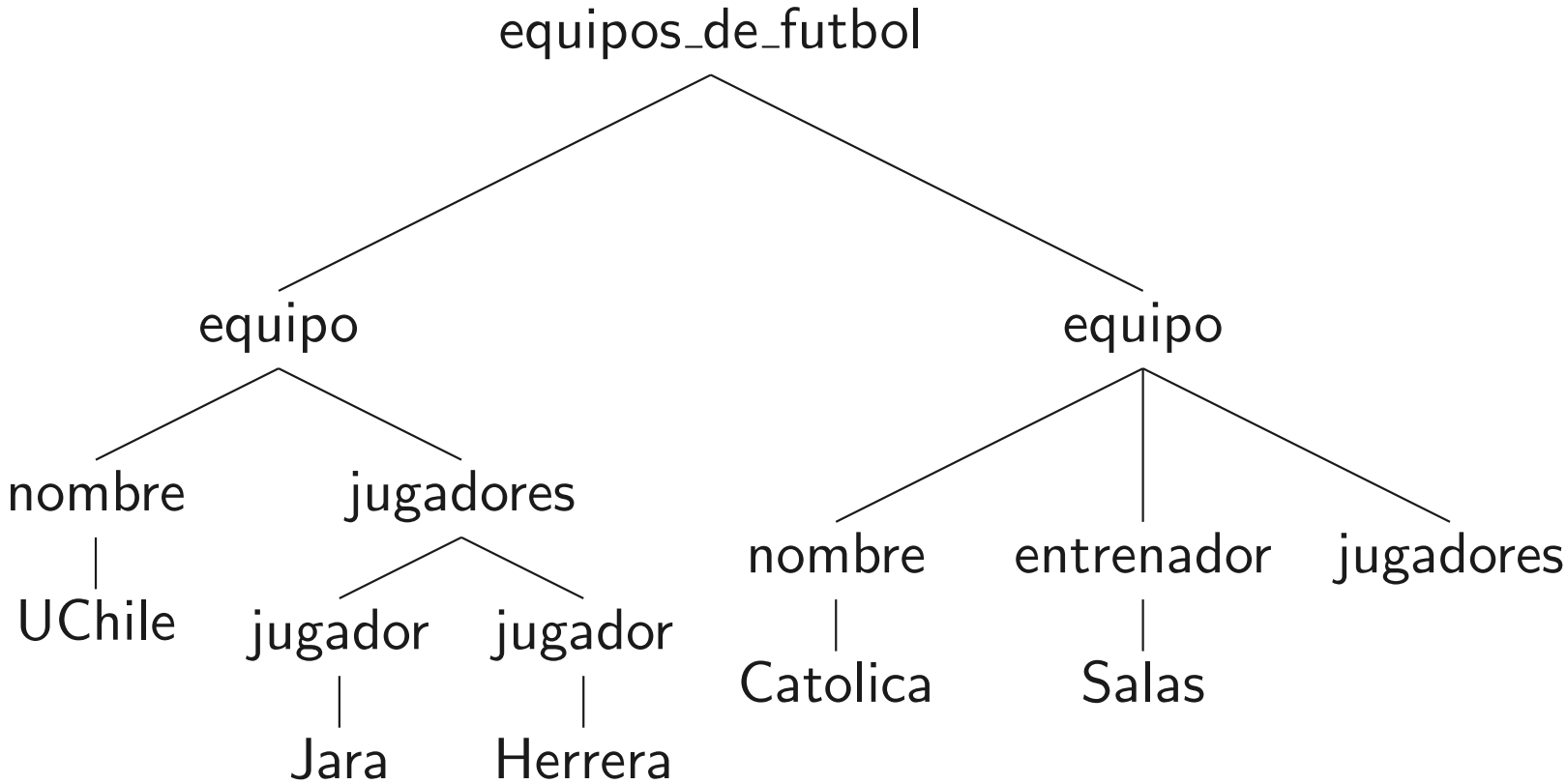
```
<equipos_de_futbol>

   <equipo>
      <nombre> UChile </nombre>
      <jugadores>
         <jugador> Jara </jugador>
         <jugador> Herrera </jugador>
      </jugadores>
   </equipo>

   <equipo>
      <nombre> Católica </entrenador>
      <entrenador> Salas </entrenador>
      <jugadores>
      </jugadores>
   </equipo>

</equipos_de_futbol>
```

Arboles: Modelo del Documento XML



Un poco de historia

- El padre de los formatos de marcado es SGML.

SGML: Standard Generalized Markup Language.

Un poco de historia

- El padre de los formatos de marcado es SGML.

SGML: Standard Generalized Markup Language.

- Propuesto por Charles Goldfarb y otros durante los 80.

Un poco de historia

- El padre de los formatos de marcado es SGML.

SGML: Standard Generalized Markup Language.

- Propuesto por Charles Goldfarb y otros durante los 80.
- Un lenguaje para el diseño de tipos de documentos.

Un poco de historia

- El padre de los formatos de marcado es SGML.

SGML: Standard Generalized Markup Language.

- Propuesto por Charles Goldfarb y otros durante los 80.
- Un lenguaje para el diseño de tipos de documentos.
- SGML es un lenguaje complicado y difícil de implementar.

Un poco de historia

- Una de las aplicaciones de SGML fue HTML.

HTML: HyperText Markup Language.

Un poco de historia

- Una de las aplicaciones de SGML fue HTML.

HTML: HyperText Markup Language.

- Propuesto por Tim Berners-Lee en 1991 para la Web.

Un poco de historia

- Una de las aplicaciones de SGML fue HTML.

HTML: HyperText Markup Language.

- Propuesto por Tim Berners-Lee en 1991 para la Web.
- Mucho más simple de implementar comparado con SGML.

Un poco de historia

- Una de las aplicaciones de SGML fue HTML.

HTML: HyperText Markup Language.

- Propuesto por Tim Berners-Lee en 1991 para la Web.
- Mucho más simple de implementar comparado con SGML.
- Creado para desplegar información y referenciar documentos.

Un poco más de historia

- Problemas de SGML y HTML:

Un poco más de historia

- Problemas de SGML y HTML:
  - SGML es un lenguaje muy complicado.

Un poco más de historia

- Problemas de SGML y HTML:
  - SGML es un lenguaje muy complicado.
  - HTML es un lenguaje para desplegar información.

Un poco más de historia

- Problemas de SGML y HTML:
  - SGML es un lenguaje muy complicado.
  - HTML es un lenguaje para desplegar información.

- **XML** surge de tener un lenguaje más **simple** y **general**.

XML: eXtensible Markup Language.

Un poco más de historia

- Problemas de SGML y HTML:
  - SGML es un lenguaje muy complicado.
  - HTML es un lenguaje para desplegar información.

- **XML** surge de tener un lenguaje más **simple** y **general**.

XML: eXtensible Markup Language.

- **XML 1.0** es propuesto en 1998 como W3C Recommendation.

Un poco más de historia

- Problemas de SGML y HTML:
  - SGML es un lenguaje muy complicado.
  - HTML es un lenguaje para desplegar información.

- **XML** surge de tener un lenguaje más **simple** y **general**.

XML: eXtensible Markup Language.

- **XML 1.0** es propuesto en 1998 como W3C Recommendation.

- Es una versión simplificada de SGML.

Un poco más de historia

- Problemas de SGML y HTML:
  - SGML es un lenguaje muy complicado.
  - HTML es un lenguaje para desplegar información.

- **XML** surge de tener un lenguaje más **simple** y **general**.

XML: eXtensible Markup Language.

- **XML 1.0** es propuesto en 1998 como W3C Recommendation.

- Es una versión simplificada de SGML.

- Todo documento XML es compatible con SGML.

Ventajas de XML

- Es un estándar internacional aceptado.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.
- Independiente de la plataforma o software.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.
- Independiente de la plataforma o software.
- Contiene estructura y datos a la vez.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.
- Independiente de la plataforma o software.
- Contiene estructura y datos a la vez.
- Es extensible, puede ser actualizado de forma incremental.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.
- Independiente de la plataforma o software.
- Contiene estructura y datos a la vez.
- Es extensible, puede ser actualizado de forma incremental.
- Puede representar la mayoría de los dominios.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.
- Independiente de la plataforma o software.
- Contiene estructura y datos a la vez.
- Es extensible, puede ser actualizado de forma incremental.
- Puede representar la mayoría de los dominios.
- Fácil de implementar y procesar.

Ventajas de XML

- Es un estándar internacional aceptado.
- Legible por cualquier maquina o sistema operativo.
- Independiente de la plataforma o software.
- Contiene estructura y datos a la vez.
- Es extensible, puede ser actualizado de forma incremental.
- Puede representar la mayoría de los dominios.
- Fácil de implementar y procesar.

Desventajas de XML

- Sintaxis es redundante.

Desventajas de XML

- Sintaxis es redundante.
- Aumenta el costo al enviar, guardar o procesar datos.

Desventajas de XML

- Sintaxis es redundante.
- Aumenta el costo al enviar, guardar o procesar datos.
- Sintaxis es verbosa.

Desventajas de XML

- Sintaxis es redundante.
- Aumenta el costo al enviar, guardar o procesar datos.
- Sintaxis es verbosa.
- No representa la semantica de los datos.

Desventajas de XML

- Sintaxis es redundante.
- Aumenta el costo al enviar, guardar o procesar datos.
- Sintaxis es verbosa.
- No representa la semantica de los datos.
- Para algunos dominios no es ni simple ni eficiente.

XML Hoy

Principales aplicaciones

- Información en la Web
  - HTML5 es en realidad un tipo de XML
  - Envío de datos.
  - Estándares: Web Services, SOAP

XML Hoy

Principales aplicaciones

- Información en la Web
  - HTML5 es en realidad un tipo de XML
  - Envío de datos.
  - Estándares: Web Services, SOAP

XML Hoy

Principales aplicaciones

- Información en la Web
  - HTML5 es en realidad un tipo de XML
  - Envío de datos.
  - Estándares: Web Services, SOAP
- Documentos e imagenes
  - OpenDocument (ODF): documentos tipo office.
  - RSS: newsfeeds.
  - Scalable Vector Graphics (SVG): grafica de vectores.

Ejemplo de XML para documentos

```
<document title = "Defensa de la Tierra" año = "1973">

<author> Luis Oyarzún </author>

<capitulo titulo = "Santiago">
</paginas inicio = "45" fin = "57">
<texto> Casi todos los días, en Santiago el habitante
sufre, consciente o <italic>inconscientemente</italic>,
la falta de paisaje, la asfixiante plenitud atmosférica
y urbana...
...
</capitulo>
<capitulo titulo = ...>
....
```

Detalles de XML

- XML esta basado en **Unicode**.

Detalles de XML

- XML esta basado en **Unicode**.
- La versión y codificación va en el encabezado:

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

Detalles de XML

- XML esta basado en **Unicode**.
- La versión y codificación va en el encabezado:

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

- Por ejemplo, un documento XML sería:

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

<palabras_complicadas>
   <palabra> ñandú </palabra>
   <palabra> ... <palabra>
     ...
</palabras_complicadas>
```

Detalles de XML

- **Tags** pueden tener cualquier tipo de nombre.
- Ejemplos:

  `<html>`, `<Equipos_De_Fútbol>`, `<miClave1234>`, ...

Detalles de XML

- **Tags** pueden tener cualquier tipo de nombre.
- Ejemplos:

  `<html>, <Equipos_De_Fútbol>, <miClave1234>, ...`

- No pueden empezar con número o carácter de puntuacón.
- No pueden contener espacios.
- Los nombres de tags son case sensitive.

Detalles de XML

- **Tags** pueden tener cualquier tipo de nombre.
- Ejemplos:

  `<html>`, `<Equipos_De_Fútbol>`, `<miClave1234>`, ...

- No pueden empezar con número o carácter de puntuacón.
- No pueden contener espacios.
- Los nombres de tags son case sensitive.
- Es importante una buena selección de los nombres.

Atributos

- Cada tag puede contener **atributos**, información adicional.
- Ejemplo:

```
<equipo nombre="Católica" ranking="48">
```

Atributos

- Cada tag puede contener **atributos**, información adicional.
- Ejemplo:

```
<equipo nombre="Católica" ranking="48">
```

- Sintaxis: `atributo="valor"` o `atributo='valor'`.

Atributos

- Cada tag puede contener **atributos**, información adicional.
- Ejemplo:

```
<equipo nombre="Católica" ranking="48">
```

- Sintaxis: `atributo="valor"` o `atributo='valor'`.
- Para usar comillas simples: `atributo="simples'"`.
- Para usar comillas dobles: `atributo='dobles"'`.

Atributos

Representan información adicional:

- El orden NO es importante.

Atributos

Representan información adicional:

- El orden NO es importante.
- Los siguientes tag son equivalentes:

```
<equipo nombre="Católica" ranking="48">
```

```
<equipo ranking="48" nombre="Católica">
```

Atributos

Representan información adicional:

- El orden NO es importante.
- Los siguientes tag son equivalentes:

```
<equipo nombre="Católica" ranking="48">
<equipo ranking="48" nombre="Católica">
```

- No pueden repetirse en un mismo tag.

¿ Son los atributos necesarios en XML ?

¿ Son los atributos necesarios en XML ?

- Es posible emular atributos con tags. ¿Como?

¿ Son los atributos necesarios en XML ?

- Es posible emular atributos con tags. ¿Como?
- Problemas con los atributos:
  - No pueden contener estructura.
  - No son extensibles.
  - Son difíciles de leer de a muchos.

¿ Son los atributos necesarios en XML ?

- Es posible emular atributos con tags. ¿Como?
- Problemas con los atributos:
  - No pueden contener estructura.
  - No son extensibles.
  - Son difíciles de leer de a muchos.

¿ Son los atributos necesarios en XML ?

- Es posible emular atributos con tags. ¿Como?
- Problemas con los atributos:
  - No pueden contener estructura.
  - No son extensibles.
  - Son difíciles de leer de a muchos.
- Ventajas de los atributos:
  - Simples.
  - Sucintos.

Caracteres especiales

- Los siguientes caracteres NO se pueden usar en XML:

< > & " '

Caracteres especiales

- Los siguientes caracteres NO se pueden usar en XML:

  < > & " '

- Para ocuparlos es necesario controles especiales:

  < → <
  > → >
  & → &
  " → '
  ' → "

Caracteres especiales

- Los siguientes caracteres NO se pueden usar en XML:

  < > & " ’

- Para ocuparlos es necesario controles especiales:

<	→	<
>	→	>
&	→	&
"	→	'
’	→	"

- Por ejemplo:
  `<comparacion> ColoColo < UChile </comparacion>`

Caracteres especiales

- Los siguientes caracteres NO se pueden usar en XML:

  < > & " ’

- Para ocuparlos es necesario controles especiales:

<	→	<
>	→	>
&	→	&
"	→	'
’	→	"

- Por ejemplo:

  `<comparacion> ColoColo < UChile </comparacion>` ×

  `<comparacion> ColoColo < UChile </comparacion>` ✓

Comentarios

- Para hacer comentarios en XML usar:

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

Comentarios

- Para hacer comentarios en XML usar:

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

- No son considerados al procesar el documento.