INFORME XML ESQUEMA

CONTENIDO

INTRODUCCIÓN AL XML ESQUEMA

Este informe recoge todos los cambios que se han realizado sobre el archivo 'circuito.xsd' para utilizar las características de los XML Esquemas. Los DTD solo permiten un tipo de datos 'String' denominado PCDATA en los elementos y CDATA en los atributos. En los XML Esquema se permiten más tipos de datos, restricciones y rangos.

Elemento longitud del circuito

Para empezar, la longitud se expresa con el tipo de datos decimal, lo que permite expresar un valor numérico exacto (proporciona una mayor precisión), sin errores de redondeo.

Sin embargo, suponiendo que será necesario utilizar el tipo de datos decimal en más partes del código, y que debería de ser en todos los casos un número positivo, se decide imponer más restricciones sobre este tipo, para que solo acepte números positivos (mayores que cero).

Otro cambio realizado en esta sección es en relación con las unidades. Mientras que antes era de tipo 'string', ahora se ve más coherente que solo acepte unos valores específicos: 'km' (kilometro) y 'm' (metro). Esto mejora la validación y permite reducir errores.

Para llevar a cabo esto, se crea una variable global "unidades" (pues se utilizará en más elementos) y se especifica en el atributo.

Se adjuntan a continuación las imágenes que representan los cambios realizados.

```
<!-- atributos globales-->
<xs:simpleType name="unidades">
    <xs:restriction base="xs:string">
        <xs:enumeration value="cm" />
        <xs:enumeration value="m" />
        <xs:enumeration value="km" />
    </xs:restriction>
</xs:simpleType>
```

```
<!-- Definición de un decimal positivo -->
<xs:simpleType name="decimalPositivo">
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value="0"/>
    </xs:restriction>
</xs:simpleType>
```

```
<!-- longitud -->
</xs:element>
<xs:element name="nombre" type="xs:string" />
<xs:element name="longitudCircuito">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="decimalPositivo">
                <xs:attribute name="unidades" type="unidades" use="required" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Elemento anchura del circuito

Este elemento es muy similar al anterior. Primero, se cambia el tipo de datos para que únicamente acepte decimales positivos. Para esto, se utiliza el tipo de datos definido en el apartado anterior, denominado "decimalPositivo".

Además, la unidades se deben expresar en metros, por lo que conviene usar la variable global "unidades", que enumera las posibles opciones para este campo.

```
<!-- anchura -->
<xs:element name="anchura">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="decimalPositivo">
                <xs:attribute name="unidades" type="unidades" use="required" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Elemento fecha del circuito

En este elemento se produce un cambio notable. Mientras que en DTD no hay un tipo de datos específico para fechas, en los XML Esquemas sí. Utilizando este tipo de datos denominado "date" se reduce el número de líneas de código considerablemente.

```
<!-- fecha -->
<xs:element name="fecha">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="dia" />
            <xs:element ref="mes" />
            <xs:element ref="año" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="dia" type="xs:string" />
<xs:element name="mes" type="xs:string" />
<xs:element name="año" type="xs:string" />
```

```
<!-- fecha -->
<xs:element name="fecha" type="xs:date"/>
```

Elemento hora del circuito

Este elemento trata de especificar la hora en la que se realiza la carrera en el circuito. Con XML Esquemas se puede utilizar el tipo de datos "time" en vez de "string", que garantiza que la hora especificada sea válida.

En la imagen se puede observar el cambio ya realizado.

```
<!-- hora -->
<xs:element name="hora" type="xs:string" />
```

Elemento vueltas del circuito

Este elemento especifica el número de vueltas que los pilotos deben de dar en el circuito. Como siempre debería ser un número positivo, se impone una restricción al tipo de datos “int”.

En la imagen a continuación se puede ver el cambio ya realizado.

```
<!-- vueltas -->
<xs:element name="vueltas">
    <xs:simpleType>
        <xs:restriction base="xs:int">
            <xs:minInclusive value="0"/> <!-- Permite solo valores 0 o positivos -->
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Elementos localidad y país del circuito

Estos dos elementos no requieren ningún cambio con respecto al tipo de datos, pues ambos deben de ser de tipo “string”.

```
<!-- localidad -->
<xs:element name="localidad" type="xs:string" />

<!-- país -->
<xs:element name="pais" type="xs:string" />
```

Elementos referencias y link del circuito

El elemento referencias sirve para encapsular varios links que se quieren mostrar, por tanto, debe especificar que como mínimo debe contener un link. No hay máximo, podrá encapsular un número indefinido de links.

El elemento link se coloca dentro del elemento referencias y es de tipo “anyURI”. De esta manera, se asegura que los links introducidos sean válidos.

Este último elemento, link, tiene además un atributo de tipo “string” para indicar el texto visible en lugar de la URL del link.

```
<!-- referencias -->
<xs:element name="referencias">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="unbounded" ref="link" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="link">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="linkName" type="xs:string" use="required" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Elementos fototeca e imagen del circuito

Estos elementos permiten recrear una especie de galería, donde se almacenan distintas fotografías. El elemento fototeca contiene una secuencia de elementos imagen, que pueden aparecer un número indefinido de fotos, pero como mínimo una vez.

El elemento imagen ha cambiado considerablemente. Para empezar, se cree conveniente que sea de tipo "anyURI" para asegurar que los enlaces introducidos de imágenes sean válidos. El mayor cambio realizado es la agregación de dos atributos antes no considerados: formato y descripción. Estos dos atributos antes no se tenían en cuenta, pero ahora se cree oportuno tenerlos en cuenta.

Los atributos formato y descripción son de tipo "string", pues en ellos simplemente se almacena el tipo de recurso y una breve descripción de la imagen. Cabe destacar que son opcionales, no se obliga a especificarlos.

Véase la imagen a continuación.

```
<!-- fototeca -->
<xs:element name="fototeca">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="unbounded" ref="imagen" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="imagen">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="formato" type="xs:string" use="optional" />
                <xs:attribute name="descripcion" type="xs:string" use="optional" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Elementos galería y vídeo del circuito

Estos elementos son similares a los anteriores. Imitan una galería de vídeos, pero no es obligatorio que en el circuito haya una galería. Por esto, se realiza un pequeño cambio al inicio del XML Esquema. En la secuencia de elementos que debe de tener el circuito, en el lugar donde aparece el elemento galería se añade "minOccurs=0".

El resto es similar al elemento imagen anterior. El elemento vídeo es de tipo "anyURI" para asegurar que los enlaces introducidos sean válidos y tiene dos atributos: formato y descripción.

Los atributos formato y descripción son de tipo "string" y permiten especificar el formato del archivo de vídeo y añadir una breve descripción del vídeo.

En la imagen posterior se verán mejor los cambios.

```xml
<xs:element name="circuito">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="nombre" />
            <xs:element ref="longitudCircuito" />
            <xs:element ref="anchura" />
            <xs:element ref="fecha" />
            <xs:element ref="hora" />
            <xs:element ref="vueltas" />
            <xs:element ref="localidad" />
            <xs:element ref="pais" />
            <xs:element ref="referencias" />
            <xs:element ref="fototeca" />
            <xs:element ref="galeria" minOccurs=0 />
            <xs:element ref="coordenadas" />
            <xs:element ref="trazosCircuito" />
        </xs:sequence>
        <xs:attribute name="nombre" type="xs:string" use="required" />
    </xs:complexType>
```

```xml
<xs:element name="video">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="formato" type="xs:string" use="optional" />
                <xs:attribute name="descripcion" type="xs:string" use="optional" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Elementos coordenadas, longitud, latitud y altitud del circuito

Estos elementos representan las coordenadas de un punto del mapa. El elemento coordenadas debe disponer de los elementos longitud, latitud y altitud obligatoriamente para que la coordenada sea válida.

Estos elementos (longitud, latitud y altitud) son de tipo decimal para poder almacenar decimales.

```xml
<!-- coordenadas -->
<xs:element name="coordenadas">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="longitud" />
            <xs:element ref="latitud" />
            <xs:element ref="altitud" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="longitud" type="xs:decimal" />
<xs:element name="latitud" type="xs:decimal" />
<xs:element name="altitud" type="xs:decimal" />
```

Elemento trazos del circuito

Este elemento posiblemente sea el más complejo del XML Esquema. Este elemento quiere representar un trazo del circuito, que además conviene que sea identificable, por lo que contiene elementos sector con un identificador.

Este elemento sector tiene un atributo que simula un ID único (un número identificativo) de tipo “integer” y contiene la información de la distancia del tramo, coordenadas y número de sector.

El elemento distancia del tramo es de tipo “decimalPositivo”, es decir, utiliza la variable global creada en un apartado anterior para asegurar que almacena decimales positivos. Además, tiene un atributo unidades, de tipo “unidades” (creado también anteriormente) para asegurar que se introduzca con una métrica permitida.

El elemento coordenadas es el explicado en el apartado anterior (coordenadas, longitud, latitud y altitud).

El elemento “numSector” indica el número de sector al que pertenece el trazo actual, por lo que es de tipo de datos “integer”.

Se adjunta a continuación una imagen.

```
<!-- trazos circuito -->
<xs:element name="trazosCircuito">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="unbounded" ref="sector" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="sector">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="distanciaTramo" />
            <xs:element ref="coordenadas" />
            <xs:element ref="numSector" />
        </xs:sequence>

        <xs:attribute name="id" type="xs:integer" use="required" />
    </xs:complexType>
</xs:element>

<xs:element name="distanciaTramo">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="decimalPositivo">
                <xs:attribute name="unidades" type="unidades" use="required" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>

<xs:element name="numSector" type="xs:integer" />
```

CONCLUSIÓN

En la mayoría de los casos los cambios realizados son por cuestiones del tipo de datos utilizado. Esto ocurre debido a que los DTD solo tienen soporte para tipos de datos muy básicos, como PCDATA, para definir los elementos. Con XML Esquemas se consigue una gran variedad de tipos de datos, además de tipos de datos complejos y restricciones.

Con los DTD la validación es más limitada, ya que solo verifica que la estructura del .xml coincida con el DTD, pero no valida el contenido en profundidad. XML Esquemas verifica también los valores de los datos.