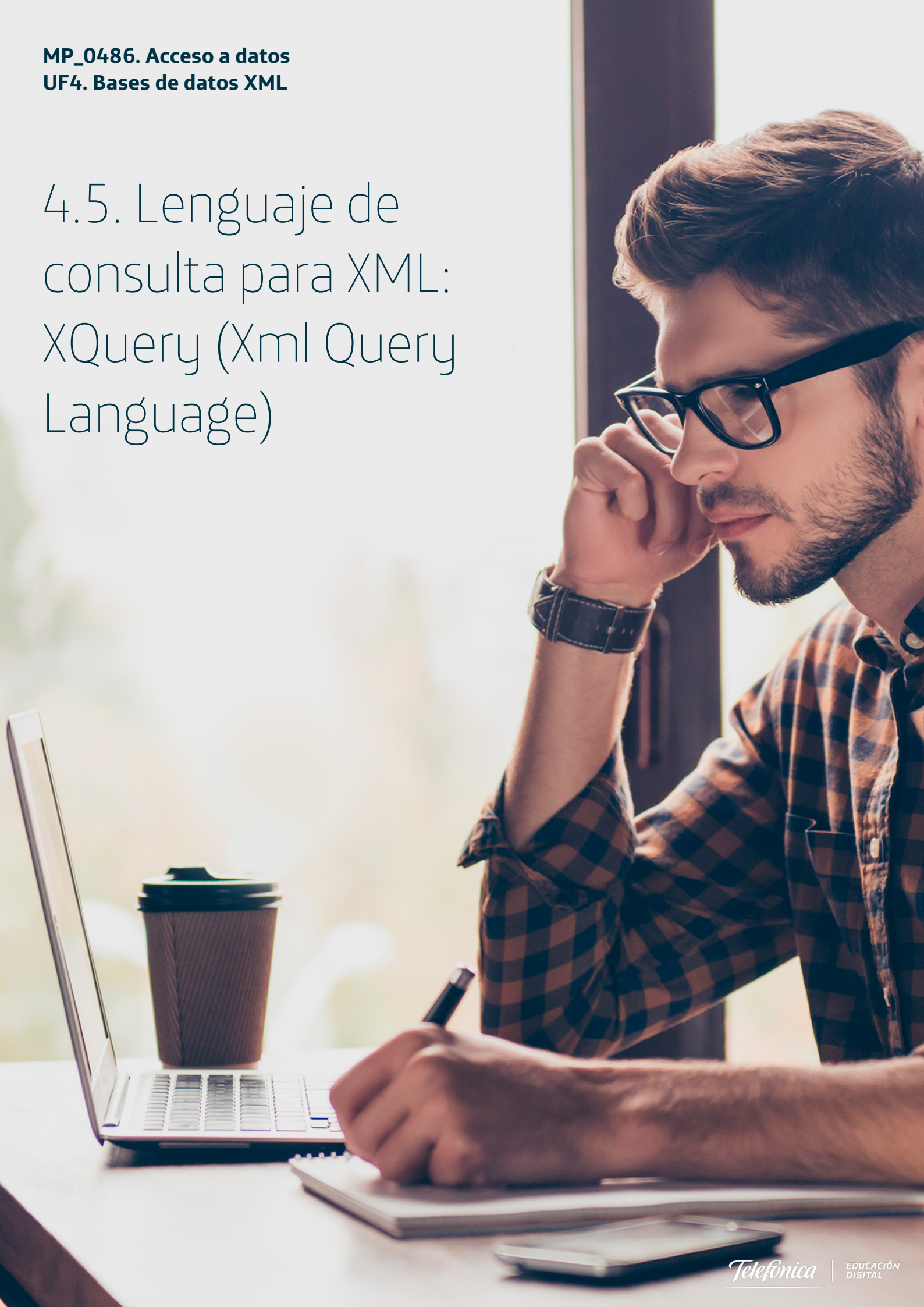
4.5. Lenguaje de consulta para XML: XQuery (Xml Query Language)



Índice

Objetivos

En esta lección perseguimos los siguientes objetivos:

- Utilizar el lenguaje XQuery para formular consultas en bases de datos XML.
- Utilizar la aplicación BaseX para escribir y ejecutar las consultas XQuery.

XQuery

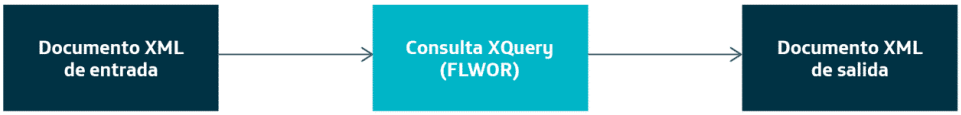
Introducción a XQuery

Desde enero de 2007, **XQuery** es el estándar para la consulta de bases de datos XML recomendado por W3C.

Estas son sus características principales:

- Su sintaxis se parece a SQL, pero incluye algunas características más propias del mundo de la programación.
- Se trata de un lenguaje que nos permite encontrar y extraer elementos dentro de un documento o base de datos XML.
- Cada consulta XQuery lee una secuencia de datos en formato XML y devuelve otra secuencia de datos XML con el resultado.
- XQuery, al igual que SQL, utiliza distintas cláusulas. Estas cláusulas siguen la norma FLWOR (de las iniciales *For*, *Let*, *Where*, *Order by* y *Return*).

La representación gráfica de una consulta XQuery quedaría así:



Representación de una consulta XQuery.

Vamos a analizar las distintas cláusulas de XQuery definidas por la palabra FLWOR. Te aconsejamos que vayas probando los distintos ejemplos que aparecerán, abriendo la aplicación BaseX y la base de datos *cruceros*, tal y como aprendiste en la lección anterior.

- **FOR:** selecciona una secuencia de nodos por medio de una expresión XPath y el resultado lo almacena en una variable de memoria.

  for $esc in /cruceros/crucero/escalas
  return $esc

  $esc es una variable de memoria cuyo contenido es el conjunto de nodos *escalas* obtenido a partir de la expresión XPath "*/cruceros/crucero/escalas*".

- **LET:** realiza un cálculo y asigna el resultado a una variable, que después puede ser utilizada en una condición *where*.

```
for $esc in /cruceros/crucero/escalas
let $c := count($esc//escala)
where $c < 7
return $esc
```

En el anterior ejemplo, para cada uno de los nodos *escalas*, cuenta el número de nodos hijo *escala* y va almacenando el resultado en la variable *$c*, que posteriormente se utiliza en la condición *where*. El resultado final son los nodos *escalas* que tienen menos de 7 nodos hijo *escala*.

- **WHERE:** filtra los nodos.

```
for $esc in /cruceros/crucero/escalas/escala
where $esc/parada="Venecia"
return $esc
```

Obtenemos los nodos *escala* con *parada* en Venecia.

- **ORDER BY:** ordena los nodos.

```
for $cru in /cruceros/crucero
order by $cru/destino
return $cru
```

Obtenemos los nodos *crucero* ordenados por *destino*.

- **RETURN:** en esta cláusula se define el resultado o salida de la consulta XQuery.

```
for $cru in /cruceros/crucero
return $cru/destino
```

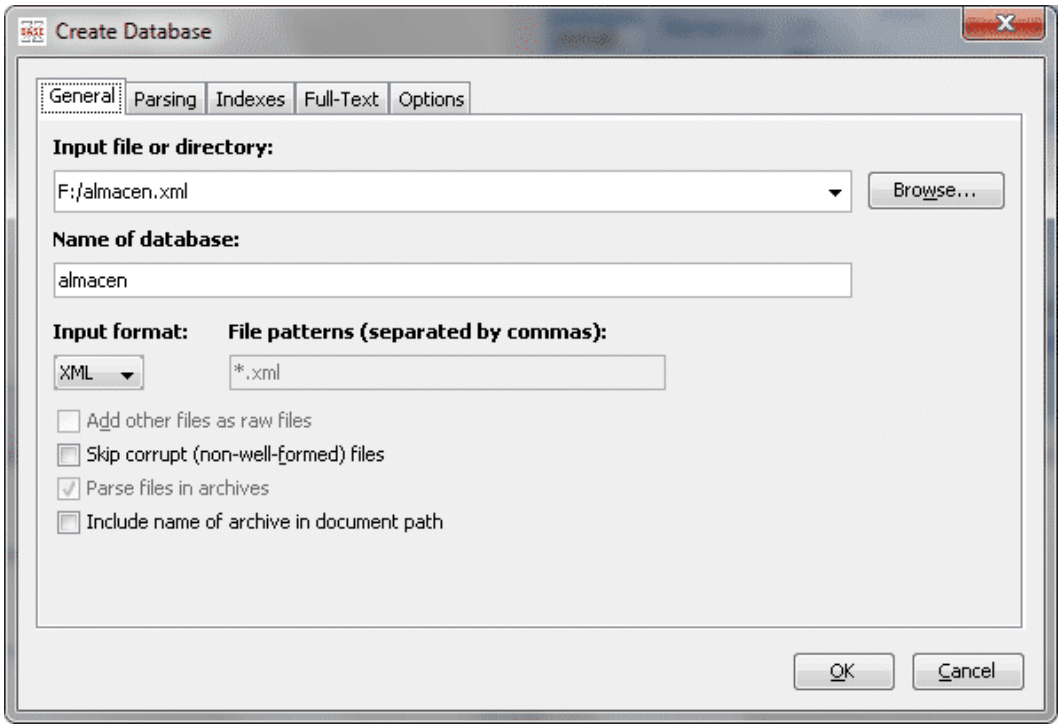
La variable *$cru* contiene todos los nodos *crucero*, pero como salida de la consulta XQuery se devuelven sólo los nodos *destino*.

Crear la base de datos ALMACEN

A continuación vamos a trabajar con una **nueva base de datos, ALMACEN,** que representará un almacén de productos de alimentación.

Para realizar este ejemplo, puedes descargarte el archivo *"almacen.xml"* en la versión online de esta lección. Si ya tienes este archivo, sigue estos pasos para crear la nueva base de datos desde BaseX y familiarizarte con ella:

1. Abre la aplicación BaseX y selecciona *Database / New* en la barra de menús.



2. Utiliza el botón *Browse* para localizar el archivo *almacen.xml* en la ubicación que lo hayas guardado. Luego, escribe *almacen* en al campo *Name of database*.

Por último, pulsa el botón *OK*.

3. Dedica un rato a examinar los datos y familiarizarte con la estructura del nuevo documento. Comprobarás que se trata de un documento XML que contiene los datos de un almacén de productos de alimentación. Tiene un nodo raíz, denominado *almacen*, que consta de un conjunto de nodos *categoria*, cada uno de ellos compuesto por un conjunto de nodos *producto*.

You're the master of your life, the captain of your ship. Steer it with intention. Will you skirt the coast from one safe harbor to the next? Or will you sail into the vast open blue? Every day you get to decide anew what course to chart.

Ya conoces la estructura del documento XML. Manos a la obra y a escribir consultas XQuery.

XPath

XQuery utiliza XPath para seleccionar los nodos requeridos en la consulta.

Por eso, utiliza la aplicación BaseX para ejecutar cada uno de los ejemplos que te iremos mostrando.

1. Comencemos por un sencillo ejemplo que mostrará todos los nombre de las categorías:

```
for $cat in /almacen/categoria/nombreCategoria
return $cat
```

El fragmento */almacen/categoria/nombreCategoria* es una expresión XPath. Hemos indicado toda la ruta dentro de la estructura del documento, hasta llegar al nodo *nombreCategoria*. Podemos simplificar la expresión del siguiente modo:

```
for $cat in //nombreCategoria
return $cat
```

Recuerda que el símbolo // selecciona el nodo especificado, independientemente del nivel en que se encuentre dentro de la estructura XML.

2. Ahora, vamos a seleccionar todos los nodos *productos* que quedarán almacenamos en la variable *$pro.*

```
for $pro in /almacen/categoria/productos/producto
return $pro
```

También podemos volver a utilizar XPath para seleccionar de nuevo elementos dentro del contenido de la variable *$pro.*

```
for $pro in /almacen/categoria/productos/producto
return $pro/nombreProducto
```

Where

Ahora vamos a completar nuestras consultas XQuery incluyendo la cláusula **WHERE.**

1. Comenzaremos por seleccionar aquellos productos cuyo precio sea superior a 30 euros.

```
for $pro in /almacen/categoria/productos/producto
where $pro/precio > 30
return $pro
```

Observa que la expresión condicional que va después de la cláusula *where* se expresa con la variable que contiene los datos previamente seleccionados, en este caso, *$pro*. En la expresión condicional volvemos a utilizar XPath para acceder a los nodos deseados dentro de cada nodo *producto*.

2. Podemos utilizar los operadores lógicos *and*, *or* y *not* para expresar criterios más complejos. En el siguiente ejemplo vamos a obtener los productos cuyo precio esté en un rango entre 30 y 40 euros.

```
for $pro in /almacen/categoria/productos/producto
where $pro/precio >= 30 and $pro/precio <= 40
return $pro
```

3. El siguiente ejemplo devuelve los productos de la categoría *Bebidas*.

```
for $cat in /almacen/categoria
where $cat/nombreCategoria = "Bebidas"
return $cat/productos/producto
```

4. Ahora, vamos a obtener todos los artículos de las categorías *bebidas* o *condimentos*.

```
for $cat in /almacen/categoria
where $cat/nombreCategoria = "Bebidas" or $cat/nombreCategoria = "Condimentos"
return $cat/productos/producto
```

5. Y, por último, mostraremos todos los artículos que no sean ni *bebidas* ni *condimentos*.

```
for $cat in /almacen/categoria
where not($cat/nombreCategoria = "Bebidas" or $cat/nombreCategoria =
"Condimentos")
return $cat/productos/producto
```

Let

En ocasiones nos interesa filtrar los resultados según el valor de una expresión calculada.

Aquí es donde juega su papel la cláusula **Let**.

1. Si multiplicamos para cada producto el stock por el precio, nos dará como resultado una cantidad que corresponde a lo que podríamos recaudar por la venta de todo el stock de dicho producto.

Imagina que te piden obtener los productos cuya recaudación posible sea mayor a 2.000 euros; lo podrías resolver con la siguiente consulta que emplea la claúsula *Let*:

```
for $pro in /almacen/categoria/productos/producto
let $recaudacion := $pro/precio*$pro/stock
where $recaudacion > 2000
return $pro
```

2. También podemos utilizar **funciones de agregado**. El siguiente ejemplo muestra el nombre de todas las categorías que cuentan con más de 10 artículos distintos.

```
for $cat in /almacen/categoria
let $c := count($cat/productos/producto)
where $c > 10
return $cat/nombreCategoria
```

Order by

Ahora utilizaremos la cláusula **ORDER BY** para ordenar los resultados obtenidos por una consulta XQuery.

1. La siguiente consulta obtiene todos los **productos ordenados por nombre**.

```
for $pro in /almacen/categoria/productos/producto
order by $pro/nombreProducto
return $pro
```

2. En este otro ejemplo, ordenamos los artículos **en orden de menor a mayor importe** que podemos recaudar por sus ventas.

```
for $pro in /almacen/categoria/productos/producto
let $recaudacion := $pro/precio*$pro/stock
order by $recaudacion
return $pro
```

3. Y ahora, vamos a realizar la misma consulta, pero esta vez **en orden descendente**, es decir, de mayor a menor recaudación.

```
for $pro in /almacen/categoria/productos/producto
let $recaudacion := $pro/precio*$pro/stock
order by $recaudacion descending
return $pro
```

Despedida

Resumen

Has terminado la lección, repasemos los puntos más importantes que hemos tratado.

- Desde enero de 2007, **XQuery es el estándar para la consulta de bases de datos XML** recomendado por W3C.
- XQuery, al igual que SQL, **utiliza distintas cláusulas**. Estas cláusulas siguen la norma FLWOR (de las iniciales *For*, *Let*, *Where*, *Order by* y *Return*).