Bloque 5 - XPATH y XQUERY

Introducción

XPath y XQuery son lenguajes de consulta utilizados para seleccionar y extraer información de documentos XML. Ambos lenguajes son desarrollados por el World Wide Web Consortium (W3C).

XPATH

XPath es un lenguaje de selección de nodos utilizado para buscar y seleccionar elementos específicos dentro de un documento XML. Permite especificar rutas de acceso a elementos y atributos utilizando una sintaxis similar a la de una ruta de archivo. XPath también permite realizar cálculos y comparaciones en los valores de los atributos y elementos.

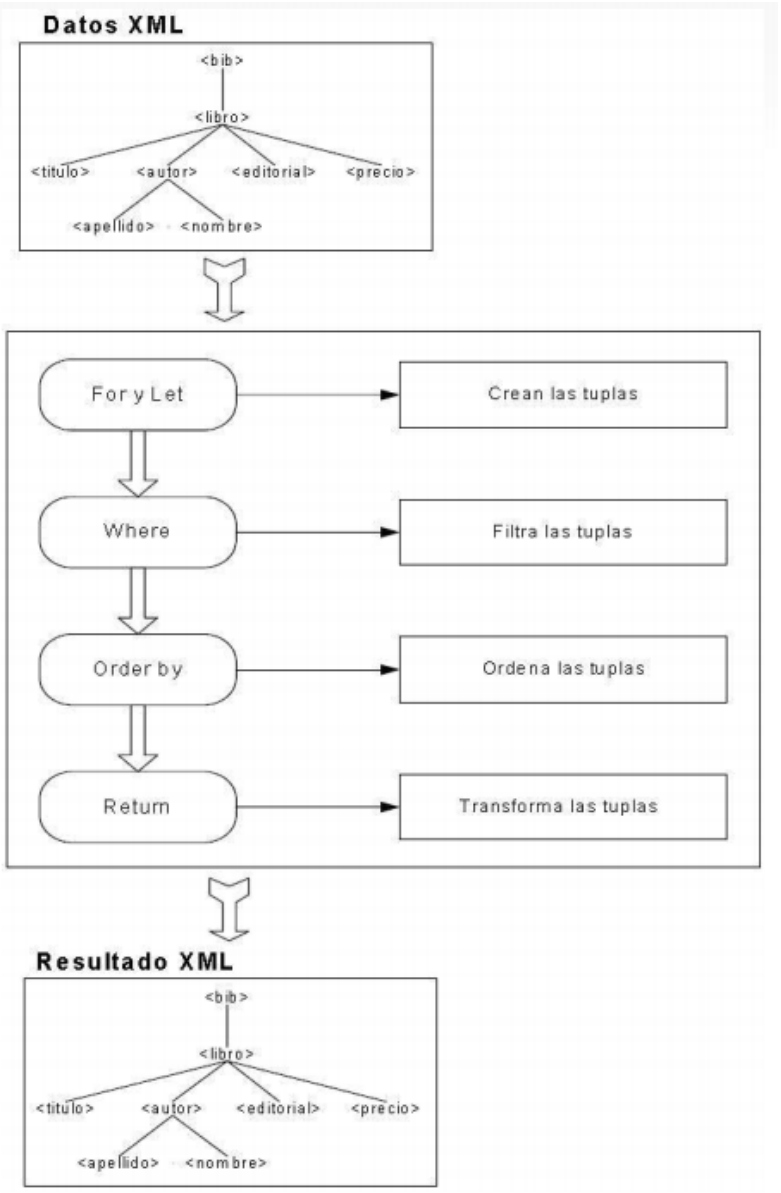
```
/
/company
/company/office
/company/office[2]
/company/office/@location
/company/office[@location = 'Boston']
/*
//*
.
*
//age/text()
/company/office[1]/following-sibling::office/@location
//employee[age > 18 and age < 30]
//attribute()
//element()
//employee[age >= 30][1]/first_name
```

	Ejemplos de rutas XPath

XQUERY

XQuery, también conocido como XML Query, es un lenguaje para buscar y extraer elementos y atributos de documentos XML. La mejor forma de entender este lenguaje es diciendo que XQuery es para XML lo que SQL es para las bases de datos. Es un lenguaje de consulta diseñado para escribir consultas sobre colecciones de datos expresadas en XML. Permite realizar consultas complejas y combinadas, como la selección de elementos y atributos específicos, la ordenación de resultados, la filtración de datos, la agregación de valores y la realización de cálculos.

Abarca desde archivos XML hasta bases de datos relacionales con funciones de conversión de registros a XML.



```
<Profesors>
{for $prof in //Profesor
return
  if ($prof/Mail) then <Prof>{$prof/text()} (mail:
{$prof/Mail/text()})</Prof>
  else <Prof>{$prof/text()} (no tiene mail)</Prof>
}
</Profesors>
```

	Ejemplos XQuery

XQuery utiliza XPath como lenguaje para acceder a los nodos y elementos de los documentos XML y otros tipos de datos estructurados.

Comparten el mismo modelo de datos y soportan las mismas funciones y operadores.

Especificación

- XML Path Language (XPath) 3.1 W3C Recommendation 21 March 2017
- XPath and XQuery Functions and Operators 3.1. W3C Recommendation 21 March 2017
- XQuery 3.1: An XML Query Language

Documentación

- Desarrollo de Aplicaciones Web - UOC (pág 168 y sucesivas)
- Documentación
- Resumen comparado con selectores CSS
- XPath tester online
- XQuery, XPath tester online
- W3C funciones XPath
- Funciones XPATH y XSLT - Mozilla develop
- Resúmenes
- XQuery
- BaseX

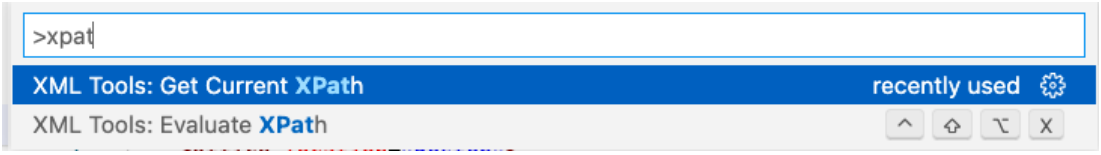
Ejercicios XPATH

1. Crea en el proyecto un fichero xml de nombre `company.xml`[1] con el siguiente código.

```
<company>
  <office location="Boston">
    <employee>
      <first_name>John</first_name>
      <last_name>Smith</last_name>
      <age>25</age>
    </employee>
  </office>
  <office location="Vienna">
    <employee>
      <first_name>Mary</first_name>
      <last_name>Brown</last_name>
      <age>30</age>
    </employee>
    <employee>
      <first_name>Peter</first_name>
      <last_name>Davis</last_name>
      <age>34</age>
    </employee>
  </office>
</company>
```

2. **Opcional** - Utiliza VSCode para **obtener la ruta XPath** de diferentes elementos o atributos del fichero `company.xml`.

   Desde la paleta de comandos:



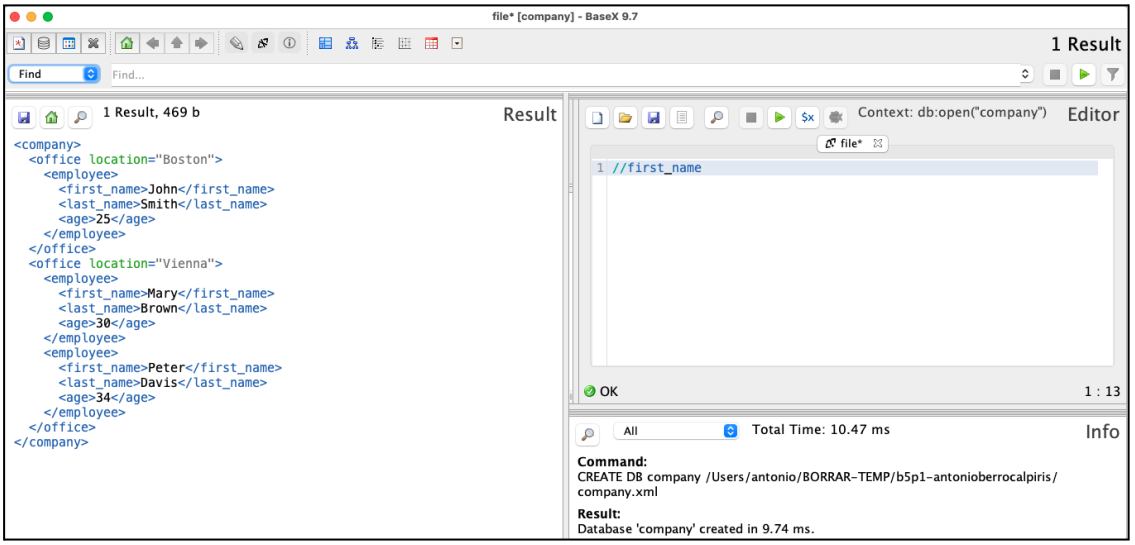
   Mostrará la ruta XPath del elemento en el que se encuentra el cursor. **La opción evaluar sentencias XPath funciona de forma incorrecta.**

[1] Fichero obtenido de https://www.altova.com/training/xpath3/location-path-expressions

```
/company/office[1]/employee/last_name
Press 'Enter' to confirm your input or 'Escape' to cancel
```

```xml
<company>
    <office location="Boston">
        <employee>
            <first_name>John</first_name>
            <last_name>Smith</last_name>
            <age>25</age>
        </employee>
    </office>
    <office location="Vienna">
        <employee>
            <first_name>Mary</first_name>
            <last_name>Brown</last_name>
            <age>30</age>
        </employee>
        <employee>
            <first_name>Peter</first_name>
            <last_name>Davis</last_name>
            <age>34</age>
        </employee>
    </office>
</company>
```

3. Descarga el software BaseX (formato `.jar`) y crea una nueva base de datos utilizando el fichero `company.xml`
   `Menú → Database → New →` buscar el fichero `company.xml` y nombre de la base de datos `company` y `ok`. Al pulsar el icono "`Casa`" se recarga toda la base de datos `company.xml`. En el panel Editor escribimos las rutas XPath.

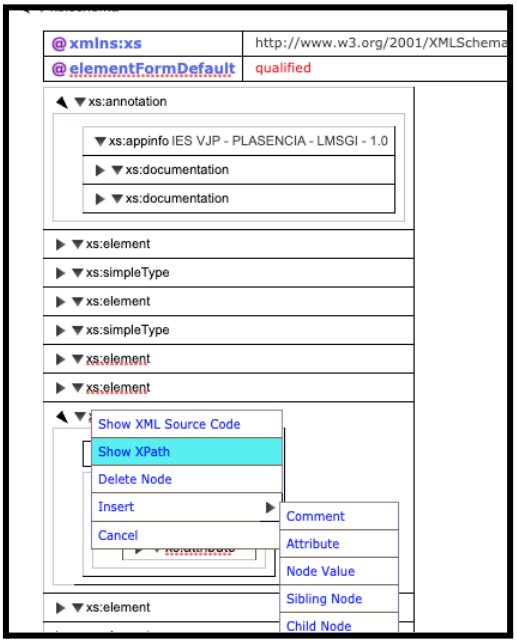


Prueba las siguientes rutas XPath incluidas en el tutorial Location Path Expressions *[En caso de error con BASEX puedes utilizar Online XPath Tester and Evaluator o Free Online XPath Tester / Evaluator - FreeFormatter.com]*

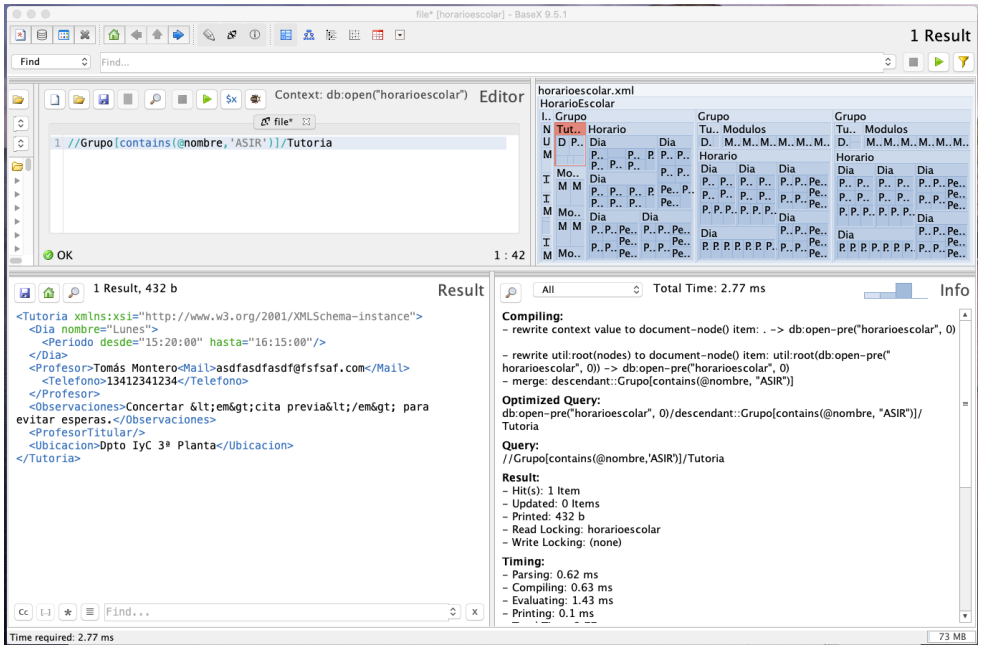
```
/
/company
/company/office
/company/office[2]
/company/office/@location
/company/office[@location = 'Boston']
/*
//*
.
*
//age/text()
/company/office[1]/following-sibling::office/@location
//employee[age > 18 and age < 30]
//attribute()
//element()
//employee[age >= 3][01]/first_name
```

4. Observa los ejemplos incluidos en https://www.freeformatter.com/xpath-tester.html

5. **A realizar por el profesor** - Utiliza https://xmlgrid.net/ para recorrer el fichero XML de los horarios disponible en el repositorio. Puede generar la ruta XPath a los distintos elementos y atributos. *Recuerda: esta opción también la tenemos disponible en VSCode.*



6. Utiliza BASEX para obtener información del documento XML de los horarios utilizando XPath 3.



Ejecución de expresión XPath en BASEX

- El elemento raíz
  `/`

- Devuelve el curso
  `/HorarioEscolar/@curso`

- Similar al anterior, pero devuelve el texto únicamente
  `/HorarioEscolar/@curso/string()`

- Devuelve todos los Profesors
  `//Profesor`

- Devuelve en formato texto todos los Profesors, incluido el texto de sus subelementos si los hay (text() solo el del elemento)
  `//Profesor/string()`

- Similar al anterior pero filtrando para que solo devuelva los distintos
  `distinct-values(//Profesor/string())`

- Igual al anterior, pero solo indica el número total de Profesors distintos
  `count(distinct-values(//Profesor/string()))`

- Devuelve el día de la tutoría del grupo 1º ASIR
  ```
  /HorarioEscolar/Grupo[@nombre="1º ASIR"]/Tutoria/Dia/@nombre/string()
  ```

- Devuelve el día de la tutoría del grupo 1º ASIR, la hora de inicio y la hora de fin.
  ```
  /HorarioEscolar/Grupo[@nombre="1º ASIR"]/Tutoria/Dia/(@nombre/string(),./Periodo/@desde/string(),./Periodo/@hasta/string())
  ```

DAW POR AQUÍ

- Devuelve el Profesor del módulo LMSGI
  ```
  //Modulo[@abbr = “LMSGI”]/Profesor
  ```

- Devuelve los Profesor del grupo 1º ASIR
  ```
  //Grupo[@nombre=”1º ASIR”]//Profesor
  ```

- Devuelve los atributos abbr y nombre de todos los Modulo del Grupo 1º DAW
  ```
  //Grupo[@nombre="1º DAW"]//Modulo/(@abbr,@nombre)
  ```

- Devuelve los atributos abbr y nombre de todos los Modulo del grupo 1º DAW y el Profesor en modo nodo de texto
  ```
  //Grupo[@nombre="1º DAW"]//Modulo/(Profesor/text(),@abbr,@nombre)
  ```

- Todas los elementos de Tutoria
  ```
  //Tutoria/*
  ```

- Todos los atributos de Tutoria, al no haberlos devuelve 0
  ```
  //Tutoria/@*
  ```

- Devuelve todos los atributos de todos los elementos descendientes de Tutoria
  ```
  //Tutoria//@*
  ```

- Devuelve todos los atributos de todos los elementos descendientes de Tutoria y el nombre del Profesor en modo nodo de texto
  ```
  //Tutoria//(@*,./Profesor/text())
  ```

- Igual al anterior añadiendo el nombre del Grupo
  ```
  //Tutoria/(.//@*,./Profesor/text(),../@nombre)
  ```

- Modulo que tienen un valor superior a 200 en el atributo horas

```
//Modulo[@horas > 200]
```

- Modulo con atributo horas con valor entre 90 y 200
  ```
  //Modulo[@horas > 90 and @horas < 200]
  ```

- Devuelve el nombre del grupo, la abreviatura del módulo y el número de horas de los módulos 224 horas
  ```
  //Modulo[@horas = 224]/(ancestor::Grupo/@nombre, @abbr, @horas)
  ```

- Devuelve los grupos cuyo nombre contiene DA
  ```
  //Grupo[contains(@nombre,"DA")]
  ```

- Devuelve el primer grupo
  ```
  //Grupo[1]
  ```

- Devuelve el penúltimo grupo
  ```
  //Grupo[last()-1]
  ```

- Devuelve el nombre del elemento padre que encierra el penúltimo grupo, es decir, devuelve el texto HorarioEscolar
  ```
  //Grupo[last()-1]/../name()
  ```

- Devuelve el número de grupos
  ```
  count(//Grupo)
  ```

- Devuelve el número de elementos en todo el árbol
  ```
  count(//*)
  ```

- Devuelve el número de elementos raíz, siempre será 1
  ```
  count(/*)
  ```

- Devuelve las abreviaturas de todos los módulos, sin repeticiones
  ```
  distinct-values(//@abbr)
  ```

- Atributos y elementos de Modulo (Union |)
  ```
  //Modulo/@*|//Modulo/node()
  ```

- Idiomas en los que se atienden consultas por mail o teléfono
  ```
  //(IES,Profesor)/(Telefono, Mail)/@idiomas
  ```

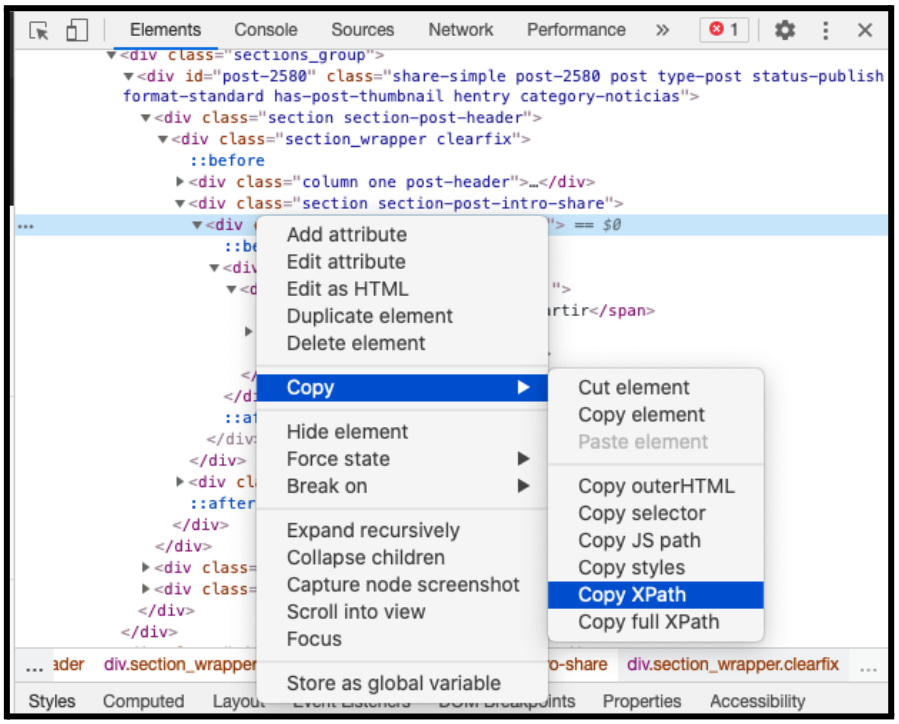
- Teléfonos y mails en los que se atienden consultas en inglés
  ```
  //(IES,Profesor)/(Telefono, Mail)[contains(@idiomas,"en")]
  ```

- Teléfono, mail y nombre de los Profesors que atienden consultas en inglés

```
//Profesor/(Telefono,
Mail)[contains(@idiomas,"en")]/../(text(),./Telefono/text(),./
Mail/text())
```

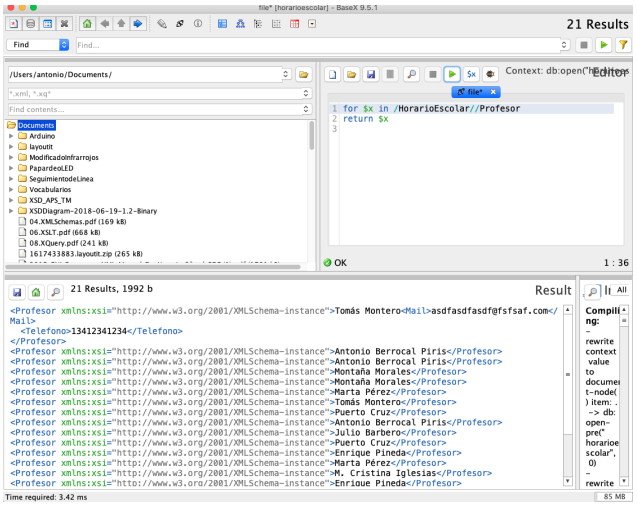
Nota: para que XPath tenga en cuenta el schema XSD (por ejemplo obtener valores por defecto) se necesita un software "schema-aware processor", como http://saxon.sourceforge.net/

7. Muestra una página Web en Google Chrome y desde las herramientas del desarrollador muestra la rutas XPath de algunos elementos



Ejercicios XQUERY y XPATH

Realiza los siguientes ejercicios sobre el documento XML de los horarios, utilizando XQuery en el software BASEX



8. Ejecuta la siguientes expresiones XQuery y observa el resultado:

for $Profesor in //Profesor return $Profesor	**let** $Profesor:= //Profesor return $Profesor

Devuelve los elementos Profesor con todo sus subelementos y atributos.

for $prof in //Profesor return <Profesores>$prof</Profesores>	let $prof := //Profesor return <Profesores>$prof</Profesores>

No devuelve ningún nodo. **Al añadir etiquetas los datos tienen que encerrarse entre llaves** {$Profesor}
Observa la diferencia entre for y let. **Con let el return se ejecuta una sola vez, con for una vez por cada elemento.**

for $prof in //Profesor return <Profesores>{$prof}</Profesores>	let $prof:=//Profesor return <Profesores>{$prof}</Profesores>

Devuelve los elementos Profesor. En el caso de for, cada uno de ellos dentro de un nuevo elemento de nombre Profesores. En el caso de let, todos dentro de un nuevo elemento de nombre Profesores.
Observa que los datos están encerrados entre llaves, es obligatorio al añadir etiquetas.

<Profesores> {for $prof in //Profesor return $prof} </Profesores>	let $prof:=//Profesor return <Profesores>{$prof}</Profesores>

Devuelve los elementos Profesor, encerrados en un elemento nuevo Profesores.
Observa las llaves.

```
<Profesors>
{for $prof in //Profesor
return
<Prof>{$prof/text()}</Prof>}
</Profesors>
```

```
<Profesors>
{let $prof:=//Profesor
return <Prof>{$prof/text()}</Prof>}
</Profesors>
```

En el caso de for, devuelve el nombre de cada Profesor encerrados en un nuevo elemento de nombre Prof dentro de otro de nombre Profesores.
En el caso de let, devuelve todos los nombres concatenados dentro de un solo elemento Prof hijo de Profesores.
Observa las llaves, después de <Profesores> y de <Prof>

```
<Profesores>
{for $prof in
distinct-values(//Profesor/text())
return <Prof>{$prof}</Prof>}
</Profesores>
```

```
<Profesores>
{let $prof :=
distinct-values(//Profesor/text())
return <Prof>{$prof}</Prof>}
</Profesores>
```

Igual a la anterior pero sin repeticiones.

```
<Profesors>
{for $prof in //Profesor
return
  if ($prof/Mail) then
<Prof>{$prof/text()} (mail:
{$prof/Mail/text()})</Prof>
  else <Prof>{$prof/text()} (no
tiene mail)</Prof>
}
</Profesors>
```

```
<Profesors>
{let $prof := //Profesor
return
  if ($prof/Mail) then
<Prof>{$prof/text()} (mail:
{$prof/Mail/text()})</Prof>
  else <Prof>{$prof/text()} (no tiene
mail)</Prof>
}
</Profesors>
```

Añade el texto mail: y el mail del Profesor que lo tenga, y no tiene mail al resto.
En el caso de let los datos salen mezclados, no sería correcto.
Observa el uso de if

9. Devuelve el resultado de las expresiones XPath del ejercicio 6 utilizando FOR

Solución:
Las expresiones XPATH pueden también indicarse en el return y utilizar funciones.

```
for $x in expresionXPATH
return $x
```

Si crea un nuevo elemento los datos tienen que ir entre llaves

```
for $x in expresionXPATH
return <nombreElemento>{$x}</nombreElemento>
```

También podemos poner la etiqueta encerrando el for, no olvidar llaves.

```
<Profesors1asir>
{
for $x in //Grupo[@nombre="1º ASIR"]//Profesor
return $x
}
</Profesorss1asir>
```

10. Devuelve el resultado de las expresiones XPath del ejercicio 6 utilizando LET

Solución:
Las expresiones XPATH pueden también indicarse en el return y utilizar funciones.
En el caso de for el return se ejecuta para cada uno, en el de let solo se ejecuta una vez, por lo que el resultado devuelto suele ser diferente.

```
let $x:= expresionXPATH
return $x
```

Si crea un nuevo elemento los datos tienen que ir entre llaves

```
let $x:= expresionXPATH
return <nombreElemento>{$x}</nombreElemento>
```

También podemos poner la etiqueta encerrando el let, no olvidar llaves.

```
<Profesors1asir>
{
let $x:= //Grupo[@nombre="1º ASIR"]//Profesor
return $x
}
</Profesors1asir>
```

11. Devuelve el listado de módulos ordenados en primer lugar por el nombre del Profesor de forma descendente y en segundo por el nombre del módulo también descendente.

```
Solución:
for $Profesor in //Modulo/Profesor
order by $Profesor descending,$Profesor/../@abbr descending
return $Profesor/..
```

12. Devuelve un nuevo elemento de nombre `resumenGrupos` con cuatro atributos: número de módulos, número de horas totales, número de Profesors y en el último el nombre del grupo.

```
Solución: observa cómo podemos utilizar la variable del for $grupos
para crear otras expresiones XPath

for $grupos in //Grupo
let $nmodulos := count($grupos/Modulos/Modulo)
let $horastotales := sum($grupos/Modulos/Modulo/@horas)
let                              $nprof                              :=
count(distinct-values($grupos/Modulos/Modulo/Profesor/text()))
return             <resumengrupos             nmodulos="{$nmodulos}"
horastotales="{$horastotales}"
nProfesors="{$nprof}">{$grupos/@nombre}</resumengrupos>
```

13. Repite el anterior, pero el nombre del grupo no lo devuelve como atributo, sino como texto del elemento.

```
Solución: observa que en vez de devolver el nodo atributo devuelve
el texto que contiene.
for $grupos in //Grupo
let $nmodulos := count($grupos/Modulos/Modulo)
let $horastotales := sum($grupos/Modulos/Modulo/@horas)
let                              $nprof                              :=
count(distinct-values($grupos/Modulos/Modulo/Profesor/text()))
return             <resumengrupos             nmodulos="{$nmodulos}"
horastotales="{$horastotales}"
nProfesors="{$nprof}">{$grupos/@nombre/string()}</resumengrupos>

Otra opción es con data()

for $grupos in //Grupo
let $nmodulos := count($grupos/Modulos/Modulo)
let $horastotales := sum($grupos/Modulos/Modulo/@horas)
let                              $nprof                              :=
count(distinct-values($grupos/Modulos/Modulo/Profesor/text()))
```

```
return <resumengrupos nmodulos="{$nmodulos}"
horastotales="{$horastotales}"
nProfesors="{$nprof}">{data($grupos/@nombre)}</resumengrupos>
```

14. Devuelva los módulos impartidos por Antonio Berrocal Piris, indicando el grupo al que pertenecen. Utiliza `where`

```
Solución:
<Modulos Profesor="Antonio Berrocal Piris">
{
 for $modulo in //Modulo
 where $modulo/Profesor[contains(.,"Antonio Berrocal")]
 return <Modulo grupo="{$modulo/../../@nombre}"
abbr="{$modulo/@abbr}" nombre="{$modulo/@nombre}"
horas="{$modulo/@horas}" />
}</Modulos>
```

15. Repite el ejercicio anterior sustituyendo el `where` por un filtro XPath

```
Solución
<Modulos Profesor="Antonio Berrocal Piris">
{
  for $modulo in //Modulo[contains(Profesor,"Antonio Berrocal")]
  return <Modulo grupo="{$modulo/../../@nombre}"
abbr="{$modulo/@abbr}" nombre="{$modulo/@nombre}"
horas="{$modulo/@horas}" />
}</Modulos>
```

Ejercicios Opcionales

16. Realiza consultas sobre el documentos XML de ofertas de empleo.

17. XQuery: Joins de documentos, anidación de for, every, some, funciones, operadores, comentarios, inserción, borrado, modificación, etc. → comparar con SQL