



TAREA AD06

Módulo de acceso a datos en modalidad a distancia del I.E.S. Augusto
González de Linares.



20 DE ENERO DE 2023
DIEGO GONZÁLEZ GARCÍA

Índice

1. EJERCICIO 1.- XPATH (universidad.xml).....	3
2. EJERCICIO 2. XQUERY (libros.xml y librosalmacen.xml)	7
3. EJERCICIO 3. (utiliza eXist en un programa JAVA)	11
a) Programa principal.....	11
b) Clase <i>BaseDatos_eXist</i>	12

Enunciado

Utilizando la base de datos XML, crear una la colección ejercicios y en ella sube los documentos universidad.xml, libros.xml y librossalmacen.xml. Los recursos necesarios para la elaboración de los ejercicios se encuentran en el siguiente enlace.

The screenshot shows the BaseX 9.3.2 interface. The top window displays the XML data for 'libros.xml', 'universidad.xml', and 'librossalmacen.xml'. The bottom window shows the query result for the query 'db:open("ejercicios", "libros.xml")'. The result is an XML document with three books:

```
<bib>
  <libro año="1994">
    <titulo>TCP/IP Illustrated</titulo>
    <autor>
      <apellido>Stevens</apellido>
      <nombre>W.</nombre>
    </autor>
    <editorial>Addison-Wesley</editorial>
    <precio>65.95</precio>
  </libro>
  <libro año="1992">
    <titulo>Advan Programming for Unix environment</titulo>
    <autor>
      <apellido>Stevens</apellido>
      <nombre>W.</nombre>
    </autor>
    <editorial>Addison-Wesley</editorial>
    <precio>65.95</precio>
  </libro>
  <libro año="2000">
    <titulo>Data on the Web</titulo>
    <autor>
      <apellido>Abiteboul</apellido>
      <nombre>Serge</nombre>
    </autor>
    <autor>
      <apellido>Buneman</apellido>
      <nombre>Peter</nombre>
    </autor>
    <autor>
      <apellido>Suciu</apellido>
      <nombre>Dag</nombre>
    </autor>
  </libro>
</bib>
```

Por los problemas con los que nos estamos encontrando con eXist, he realizado los ejercicios 1 y 2 de la tarea con BaseX, teniendo configurada una base de datos con los 3 archivos .xml requeridos en la tarea.

1. EJERCICIO 1.- XPATH (universidad.xml)

Resuelve las consultas que se plantean y envía el documento de texto con las soluciones:

Para poder mantener un poco de orden entre los ejercicios, estoy guardando las consultas XPath en el directorio *Ejercicio1*, que se encuentra dentro del directorio raíz *Ejercicios*, que es quien contiene los archivos .xml.

1) Nombre de la Universidad.

The screenshot shows an XPath query editor on the left with the following code:

```
1 (: Nombre de la Universidad. :)  
2 /universidad/nombre
```

On the right, the 'Resultado' pane shows the result of the query:

```
<nombre>Universidad de Victoria</nombre>
```

2) País de la Universidad.

The screenshot shows an XPath query editor on the left with the following code:

```
1 (: País de la Universidad. :)  
2 /universidad/pais
```

On the right, the 'Resultado' pane shows the result of the query:

```
<pais>España</pais>
```

3) Nombres de las Carreras.

The screenshot shows an XPath query editor on the left with the following code:

```
1 (: Nombres de las Carreras. :)  
2 /universidad/carreras/carrera/nombre
```

On the right, the 'Resultado' pane shows the results of the query:

```
<nombre>I.T. Informática</nombre>  
<nombre>Dipl. Empresariales</nombre>  
<nombre>Dipl. Relaciones Laborales</nombre>  
<nombre>Lic. Química</nombre>  
<nombre>Lic. Biología</nombre>  
<nombre>Lic. Humanidades</nombre>
```

4) Años de plan de estudio de las carreras.

The screenshot shows an XPath query editor on the left with the following code:

```
1 (: Años de plan de estudio de las carreras. :)  
2 /universidad/carreras/carrera/plan
```

On the right, the 'Resultado' pane shows the results of the query:

```
<plan>2003</plan>  
<plan>2001</plan>  
<plan>2001</plan>  
<plan>2003</plan>  
<plan>2001</plan>  
<plan>1980</plan>
```

5) Nombres de todos los alumnos.

The screenshot shows an XPath query editor on the left with the following code:

```
1 (: Nombres de todos los alumnos. :)  
2 /universidad/alumnos/alumno/nombre
```

On the right, the 'Resultado' pane shows the results of the query:

```
<nombre>Victor Manuel</nombre>  
<nombre>Luisa</nombre>  
<nombre>Fernando</nombre>  
<nombre>María</nombre>
```

6) Identificadores de todas las carreras.

The screenshot shows an XPath query editor on the left with the following code:

```
1 (: Identificadores de todas las carreras. :)  
2 /universidad/carreras/carrera/@id
```

On the right, the 'Resultado' pane shows the results of the query:

```
id="c01"  
id="c02"  
id="c03"  
id="c04"  
id="c05"  
id="c06"
```

7) Datos de la carrera cuyo id es c01.

The screenshot shows the XQuery Editor with the following query:

```
1 (: Datos de la carrera cuyo id es c01. :)
2 /universidad/carreras/carrera[@id='c01']
```

The Result pane shows 1 Resultado, 161 b:

```
<carrera id="c01">
  <nombre>I.T. Informática</nombre>
  <plan>2003</plan>
  <creditos>250</creditos>
  <centro>Escuela de Informática</centro>
</carrera>
```

8) Centro en que se estudia de la carrera cuyo id es c02.

The screenshot shows the XQuery Editor with the following query:

```
1 (: Centro en que se estudia de la carrera cuyo
id es c02. :)
2 /universidad/carreras/carrera[@id='c02']/
centro
```

The Result pane shows 1 Resultado, 46 b:

```
<centro>Facultad de Ciencias Sociales</centro>
```

9) Nombre de las carreras que tengan subdirector.

The screenshot shows the XQuery Editor with the following query:

```
1 (: Nombre de las carreras que tengan
subdirector. :)
2 /universidad/carreras/carrera[exists(
subdirector)]/nombre
```

The Result pane shows 1 Resultado, 43 b:

```
<nombre>Dipl. Relaciones Laborales</nombre>
```

10) Nombre de los alumnos que estén haciendo proyecto.

The screenshot shows the XQuery Editor with the following query:

```
1 (: Nombre de los alumnos que estén haciendo
proyecto. :)
2 /universidad/alumnos/alumno[exists(estudios/
proyecto)]/nombre
```

The Result pane shows 2 Resultados, 47 b:

```
<nombre>Luisa</nombre>
<nombre>María</nombre>
```

11) Códigos de las carreras en las que hay algún alumno matriculado.

The screenshot shows the XQuery Editor with the following query:

```
1 (: Códigos de las carreras en las que hay
algún alumno matriculado. :)
2 distinct-values(universidad/alumnos/alumno/
estudios/carrera/@codigo)
```

The Result pane shows 2 Resultados, 8 b:

```
c01
c02
```

12) Apellidos y Nombre de los alumnos con beca.

The screenshot shows the XQuery Editor with the following query:

```
1 (: Apellidos y Nombre de los alumnos con beca. :)
2 universidad/alumnos/alumno[@beca='si']/concat(
apellido1, ' ', apellido2, ' ', nombre)
```

The Result pane shows 1 Resultado, 22 b:

```
Pérez Romero Fernando
```

13) Nombre de las asignaturas de la titulación c04.

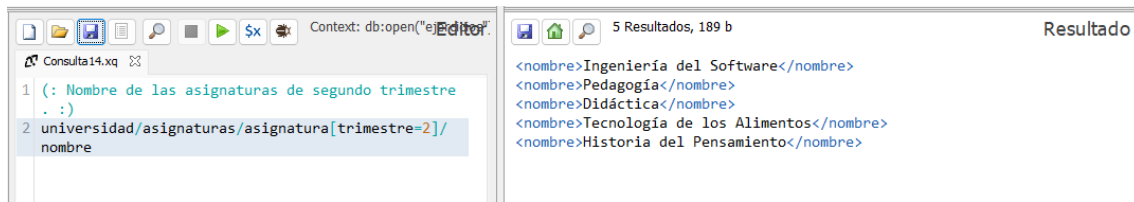
The screenshot shows the XQuery Editor with the following query:

```
1 (: Nombre de las asignaturas de la titulación c04
. :)
2 universidad/asignaturas/asignatura[@titulacion='
c04']/nombre
```

The Result pane shows 2 Resultados, 74 b:

```
<nombre>Pedagogía</nombre>
<nombre>Tecnología de los Alimentos</nombre>
```

14) Nombre de las asignaturas de segundo trimestre.



Context: db:open("e") Editor

Consulta14.xq

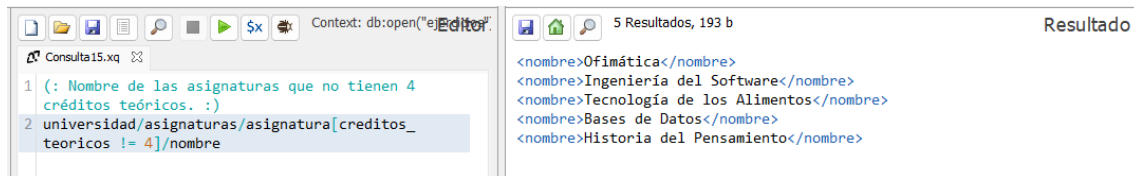
```
1 (: Nombre de las asignaturas de segundo trimestre . :)
2 universidad/asignaturas/asignatura[trimestre=2]/
  nombre
```

5 Resultados, 189 b

Resultado

```
<nombre>Ingeniería del Software</nombre>
<nombre>Pedagogía</nombre>
<nombre>Didáctica</nombre>
<nombre>Tecnología de los Alimentos</nombre>
<nombre>Historia del Pensamiento</nombre>
```

15) Nombre de las asignaturas que no tienen 4 créditos teóricos.



Context: db:open("e") Editor

Consulta15.xq

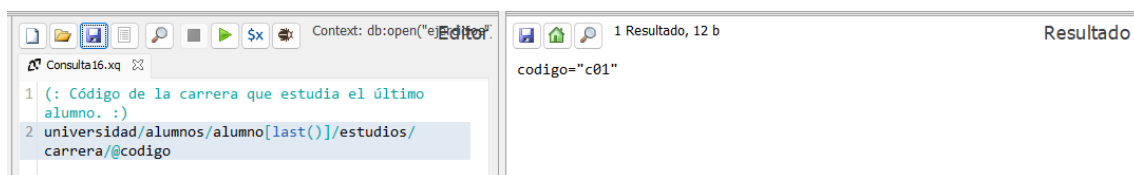
```
1 (: Nombre de las asignaturas que no tienen 4
  créditos teóricos. :)
2 universidad/asignaturas/asignatura[creditos_
  teoricos != 4]/nombre
```

5 Resultados, 193 b

Resultado

```
<nombre>Ofimática</nombre>
<nombre>Ingeniería del Software</nombre>
<nombre>Tecnología de los Alimentos</nombre>
<nombre>Bases de Datos</nombre>
<nombre>Historia del Pensamiento</nombre>
```

16) Código de la carrera que estudia el último alumno.



Context: db:open("e") Editor

Consulta16.xq

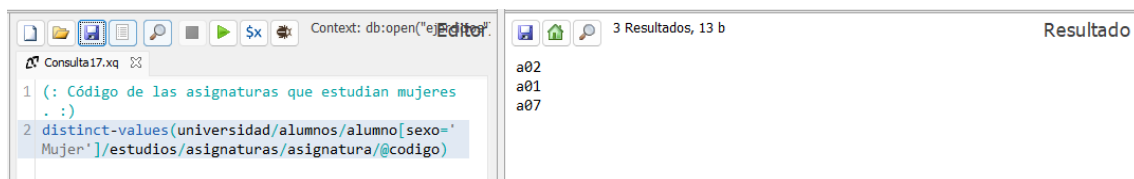
```
1 (: Código de la carrera que estudia el último
  alumno. :)
2 universidad/alumnos/alumno[last()]/estudios/
  carrera/@codigo
```

1 Resultado, 12 b

Resultado

```
codigo="c01"
```

17) Código de las asignaturas que estudian mujeres.



Context: db:open("e") Editor

Consulta17.xq

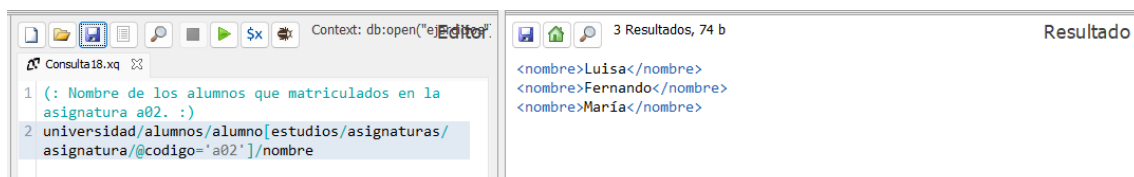
```
1 (: Código de las asignaturas que estudian mujeres
  . :)
2 distinct-values(universidad/alumnos/alumno[sexo='
  Mujer']/estudios/asignaturas/asignatura/@codigo)
```

3 Resultados, 13 b

Resultado

```
a02
a01
a07
```

18) Nombre de los alumnos que matriculados en la asignatura a02.



Context: db:open("e") Editor

Consulta18.xq

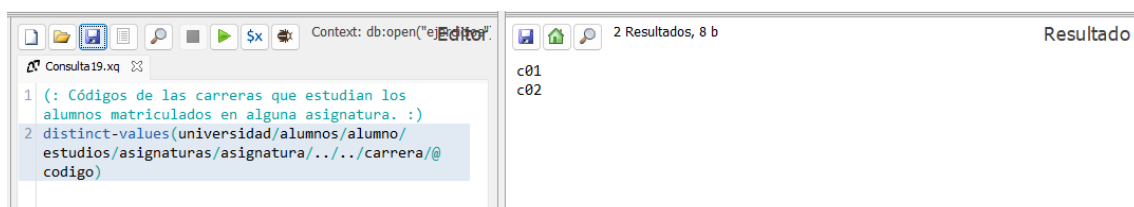
```
1 (: Nombre de los alumnos que matriculados en la
  asignatura a02. :)
2 universidad/alumnos/alumno[estudios/asignaturas/
  asignatura/@codigo='a02']/nombre
```

3 Resultados, 74 b

Resultado

```
<nombre>Luisa</nombre>
<nombre>Fernando</nombre>
<nombre>María</nombre>
```

19) Códigos de las carreras que estudian los alumnos matriculados en alguna asignatura.



Context: db:open("e") Editor

Consulta19.xq

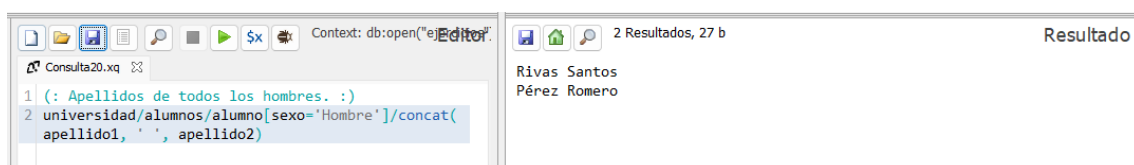
```
1 (: Códigos de las carreras que estudian los
  alumnos matriculados en alguna asignatura. :)
2 distinct-values(universidad/alumnos/alumno[
  estudios/asignaturas/asignatura/../../carrera/@
  codigo])
```

2 Resultados, 8 b

Resultado

```
c01
c02
```

20) Apellidos de todos los hombres.



Context: db:open("e") Editor

Consulta20.xq

```
1 (: Apellidos de todos los hombres. :)
2 universidad/alumnos/alumno[sexo='Hombre']/concat(
  apellido1, ' ', apellido2)
```

2 Resultados, 27 b

Resultado

```
Rivas Santos
Pérez Romero
```

21) Nombre de la carrera que estudia Víctor Manuel.

Context: db:open("ejemplo")

Consulta21.xq

```
1 (: Nombre de la carrera que estudia Víctor Manuel . :)
2 universidad/carreras/carrera[@id= [/universidad/
alumnos/alumno[nombre='Víctor Manuel']/estudios/
carrera/@codigo]]/nombre
```

1 Resultado, 34 b

Resultado

```
<nombre>I.T. Informática</nombre>
```

22) Nombre de las asignaturas que estudia Luisa.

Context: db:open("ejemplo")

Consulta22.xq

```
1 (: Nombre de las asignaturas que estudia Luisa. :)
2 universidad/asignaturas/asignatura[@id= [/
universidad/alumnos/alumno[nombre='Luisa']/
estudios/asignaturas/asignatura/@codigo]]/nombre
```

2 Resultados, 70 b

Resultado

```
<nombre>Ofimática</nombre>
<nombre>Ingeniería del Software</nombre>
```

23) Primer apellido de los alumnos matriculados en Ingeniería del Software.

Context: db:open("ejemplo")

Consulta23.xq

```
1 (: Primer apellido de los alumnos matriculados en
Ingeniería del Software. :)
2 universidad/alumnos/alumno[estudios/asignaturas/
asignatura/@codigo= [/universidad/asignaturas/
asignatura[nombre='Ingeniería del Software']/@id]]/
apellido1
```

3 Resultados, 92 b

Resultado

```
<apellido1>Pérez</apellido1>
<apellido1>Pérez</apellido1>
<apellido1>Avalón</apellido1>
```

24) Nombre de las carreras que estudian los alumnos matriculados en la asignatura Tecnología de los Alimentos.

Context: db:open("ejemplo")

Consulta24.xq

```
1 (: Nombre de las carreras que estudian los
alumnos matriculados en la asignatura Tecnología
de los Alimentos. :)
2 universidad/carreras/carrera[@id= [/universidad/
alumnos/alumno[estudios/asignaturas/asignatura/@
codigo= [/universidad/asignaturas/asignatura[
nombre='Tecnología de los Alimentos']/@id]]/
estudios/carrera/@codigo]]/nombre
```

1 Resultado, 34 b

Resultado

```
<nombre>I.T. Informática</nombre>
```

25) Nombre de los alumnos matriculados en carreras que no tienen subdirector.

Context: db:open("ejemplo")

Consulta25.xq

```
1 (: Nombre de los alumnos matriculados en carreras
que no tienen subdirector. :)
2 universidad/alumnos/alumno[not( /universidad/
carreras/carrera/@codigo= [/universidad/carreras/
carrera[subdirector]/@codigo]]/nombre
```

4 Resultados, 107 b

Resultado

```
<nombre>Víctor Manuel</nombre>
<nombre>Luisa</nombre>
<nombre>Fernando</nombre>
<nombre>María</nombre>
```

26) Nombre de los alumnos matriculados en asignaturas con 0 créditos prácticos y que estudien la carrera de I.T. Informática.

Context: db:open("ejemplo")

Consulta26.xq

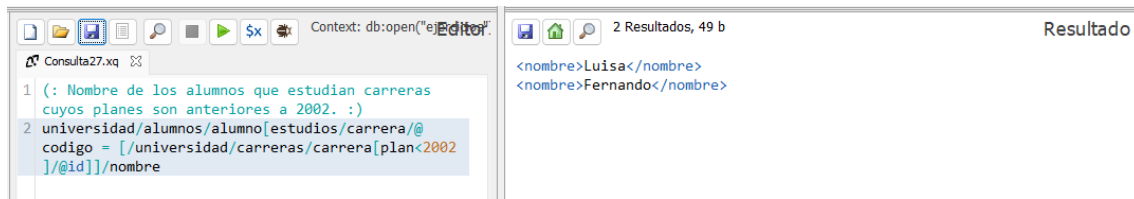
```
1 (: Nombre de los alumnos matriculados en
asignaturas con 0 créditos prácticos y que
estudien la carrera de I.T. Informática. :)
2 universidad/alumnos/alumno[estudios/asignaturas/
asignatura/@codigo = [/universidad/asignaturas/
asignatura[creditos_practicos=0]/@id]] [estudios/
carrera/@codigo = [/universidad/carreras/carrera[
nombre='I.T. Informática']/@id]]/nombre
```

1 Resultado, 31 b

Resultado

```
<nombre>Víctor Manuel</nombre>
```

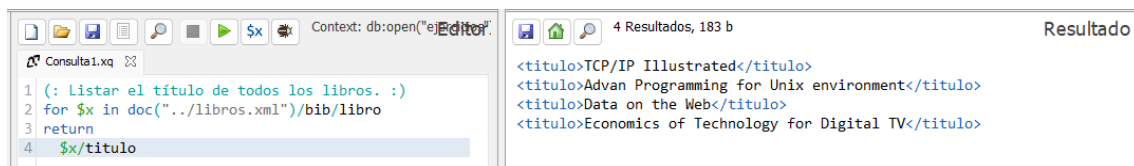
27) Nombre de los alumnos que estudian carreras cuyos planes son anteriores a 2002.



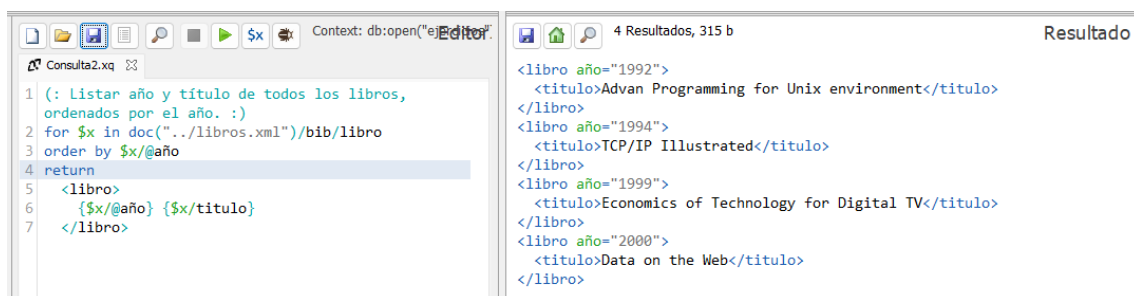
2. EJERCICIO 2. XQUERY (libros.xml y librosalmacen.xml)

Para poder mantener un poco de orden entre los ejercicios, estoy guardando las consultas XQuery en el directorio *Ejercicio2*, que se encuentra dentro del directorio raíz *Ejercicios*, que es quien contiene los archivos .xml.

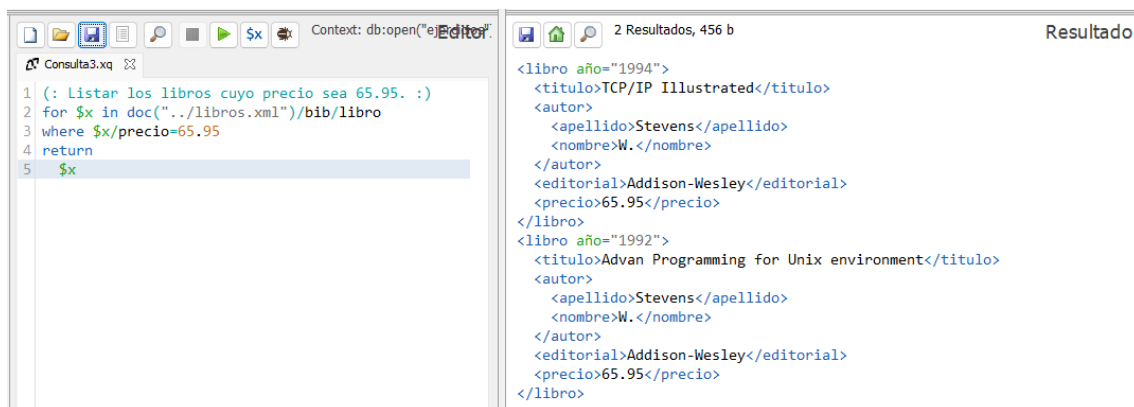
- 1) Listar el título de todos los libros.



- 2) Listar año y título de todos los libros, ordenados por el año.



- 3) Listar los libros cuyo precio sea 65.95.



- 4) Listar los libros publicados antes del año 2000.

The screenshot shows the XQuery Editor with a query named 'Consulta4.xq'. The query is designed to list books published before the year 2000. The results pane shows three XML entries for books published in 1994, 1992, and 1999.

```

Context: db:open("ejercicio")
Consulta4.xq
1 (: Listar los libros publicados antes del año 2000. :)
2 for $x in doc("../libros.xml")/bib/libro
3 where $x/@año < 2000
4 return
5   $x
  
```

3 Resultados, 748 b

```

<libro año="1994">
  <titulo>TCP/IP Illustrated</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="1992">
  <titulo>Advan Programming for Unix environment</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="1999">
  <titulo>Economics of Technology for Digital TV</titulo>
  <editor>
    <apellido>Gerbang</apellido>
    <nombre>Darcy</nombre>
    <afiliacion>CITI</afiliacion>
  </editor>
  <editorial>Kluwer Academic editorials</editorial>
  <precio>129.95</precio>
</libro>
  
```

5) Listar año y título de los libros publicados por Addison-Wesley después del año 1992.

The screenshot shows the XQuery Editor with a query named 'Consulta5.xq'. The query filters for books published by Addison-Wesley after the year 1992. The results pane shows one XML entry for a book published in 1994.

```

Context: db:open("ejercicio")
Consulta5.xq
1 (: Listar año y título de los libros publicados por Addison-Wesley después del año 1992. :)
2 for $x in doc("../libros.xml")/bib/libro
3 where $x/editorial='Addison-Wesley' and $x/@año > 1992
4 return
5   <libro>
6     { $x/@año }
7     { $x/titulo }
8   </libro>
  
```

1 Resultado, 68 b

```

<libro año="1994">
  <titulo>TCP/IP Illustrated</titulo>
</libro>
  
```

6) Listar año y título de los libros que tienen más de un autor.

The screenshot shows the XQuery Editor with a query named 'Consulta6.xq'. The query filters for books that have more than one author. The results pane shows one XML entry for a book published in 2000.

```

Context: db:open("ejercicio")
Consulta6.xq
1 (: Listar año y título de los libros que tienen más de un autor. :)
2 for $x in doc("../libros.xml")/bib/libro
3 where count($x/autor) > 1
4 return
5   <libro>
6     { $x/@año }
7     { $x/titulo }
8   </libro>
  
```

1 Resultado, 65 b

```

<libro año="2000">
  <titulo>Data on the Web</titulo>
</libro>
  
```

7) Listar año y título de los libros que no tienen autor.

The screenshot shows the XQuery Editor with a query named 'Consulta7.xq'. The query filters for books that do not have an author. The results pane shows one XML entry for a book published in 1999.

```

Context: db:open("ejercicio")
Consulta7.xq
1 (: Listar año y título de los libros que no tienen autor. :)
2 for $x in doc("../libros.xml")/bib/libro
3 where count($x/autor) = 0
4 return
5   <libro>
6     { $x/@año }
7     { $x/titulo }
8   </libro>
  
```

1 Resultado, 88 b

```

<libro año="1999">
  <titulo>Economics of Technology for Digital TV</titulo>
</libro>
  
```

8) Mostrar los apellidos de los autores que aparecen en el documento, sin repeticiones, ordenados alfabéticamente.

The screenshot shows the XQuery Editor with a query to extract author surnames from an XML document. The context is 'db:open("ejercicio8.xml")'. The query is as follows:

```

1 (: Mostrar los apellidos de los autores que aparecen
2   en el documento, sin repeticiones, ordenados
3   alfabéticamente. :)
4 for $x in distinct-values(doc("../libros.xml")/bib/
5   libro/autor/apellido)
6 order by $x
7 return
8   $x

```

The Result window shows 4 results: Abiteboul, Buneman, Stevens, and Suciu.

9) Por cada libro, listar agrupado en un elemento <result> su título y autores.

The screenshot shows the XQuery Editor with a query to group book titles and authors into <result> elements. The context is 'db:open("ejercicio9.xml")'. The query is as follows:

```

1 (: Por cada libro, listar agrupado en un elemento <
2   result> su título y autores. :)
3 for $x in doc("../libros.xml")/bib/libro
4 return
5   <result>
6     {$x/titulo}
7     {$x/autor}
8   </result>

```

The Result window shows 4 results, each containing a title and a list of authors. The results are:

- <result>
 <titulo>TCP/IP Illustrated</titulo>
 <autor>
 <apellido>Stevens</apellido>
 <nombre>W.</nombre>
 </autor>
 </result>
- <result>
 <titulo>Advan Programming for Unix environment</titulo>
 <autor>
 <apellido>Stevens</apellido>
 <nombre>W.</nombre>
 </autor>
 </result>
- <result>
 <titulo>Data on the Web</titulo>
 <autor>
 <apellido>Abiteboul</apellido>
 <nombre>Serge</nombre>
 </autor>
 <autor>
 <apellido>Buneman</apellido>
 <nombre>Peter</nombre>
 </autor>
 <autor>
 <apellido>Suciu</apellido>
 <nombre>Dan</nombre>
 </autor>
 </result>
- <result>
 <titulo>Economics of Technology for Digital TV</titulo>
 </result>

10) Por cada libro, obtener su título y el número de autores, agrupados en un elemento <libro>.

The screenshot shows the XQuery Editor with a query to group book titles and the number of authors into <libro> elements. The context is 'db:open("ejercicio10.xml")'. The query is as follows:

```

1 (: Por cada libro, obtener su título y el número de
2   autores, agrupados en un elemento <libro>. :)
3 for $x in doc("../libros.xml")/bib/libro
4 return
5   <libro>
6     {$x/titulo}
7     <n_autores>
8       {count($x/autor)}
9     </n_autores>
10  </libro>

```

The Result window shows 4 results, each containing a title and the number of authors. The results are:

- <libro>
 <titulo>TCP/IP Illustrated</titulo>
 <n_autores>1</n_autores>
 </libro>
- <libro>
 <titulo>Advan Programming for Unix environment</titulo>
 <n_autores>1</n_autores>
 </libro>
- <libro>
 <titulo>Data on the Web</titulo>
 <n_autores>3</n_autores>
 </libro>
- <libro>
 <titulo>Economics of Technology for Digital TV</titulo>
 <n_autores>0</n_autores>
 </libro>

11) Una lista ordenada alfabéticamente de libros comprados.

The screenshot shows a database editor window with a query in the left pane and its results in the right pane. The query is a Prolog-style query that iterates over books and their prices, returning the title and price. The results pane shows two rows of data, each represented as an XML-like structure with fields for year, code, title, author, name, editorial, and price.

```
Context: db:open("ejercicios") Editor
```

```
Consulta11.xq
1 (: Una lista ordenada alfabéticamente de libros
2 comprados. :)
3 for $x in doc("../libros.xml")/bib/libro
4 for $y in doc("../librosalmacen.xml")/almacen
5 where $x/@codigo = $y/comprados/codigo
6 order by $x/titulo
7 return
8 $x
```

```
2 Resultados, 478 b Resultado
```

```
<libro año="1992" código="2">
  <titulo>Advan Programming for Unix environment</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="1994" código="1">
  <titulo>TCP/IP Illustrated</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
```

12) Obtener la suma del importe de todos los libros que están pendientes.

The screenshot shows a database editor window with a query in the left pane and its result in the right pane. The query is a Prolog-style query that calculates the sum of the prices of all pending books. The result pane shows a single numerical value representing the sum.

```
Context: db:open("ejercicios") Editor
```

```
Consulta12.xq
1 (: Obtener la suma del importe de todos los libros que están
2 pendientes. :)
3 sum(
4   for $x in doc("../libros.xml")/bib/libro
5   for $y in doc("../librosalmacen.xml")/almacen/pendientes
6   where $x/@codigo = $y/codigo
7   return
8   $x/precio)
```

```
1 Resultado, 18 b Resultado
```

```
169.89999999999998
```

13) Una lista ordenada de autores que tengan libros pendientes. La última línea contendrá una línea que tenga el total de autores.

The screenshot shows a database editor window with a query in the left pane and its result in the right pane. The query is a Prolog-style query that returns a list of authors with pending books, ordered by the number of pending books. The last line of the result is a summary line showing the total number of authors.

```
Context: db:open("ejercicios") Editor
```

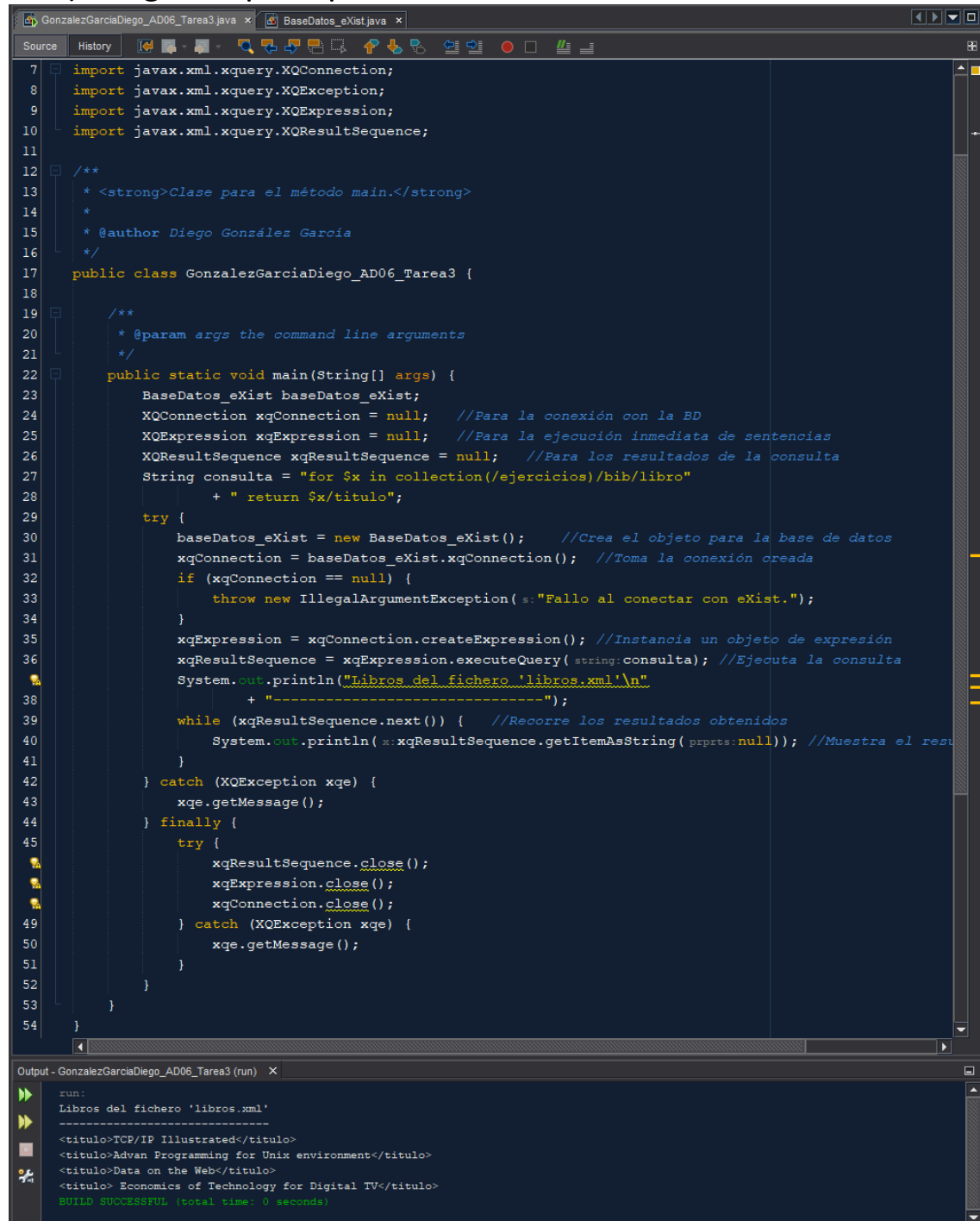
```
Consulta13.xq
1 (: Una lista ordenada de autores que tengan libros
2 pendientes. La última línea contendrá una línea que tenga
3 el total de autores. :)
4 for $x in doc("../libros.xml")/bib/libro
5 for $y in doc("../librosalmacen.xml")/almacen
6 where $x/@codigo = $y/pendientes/codigo and $x/autor
7 order by $x
8 return
9 <autores_libros_pendientes>
10 { $x/autor }
11 <total_autores>{count($x/autor)}</total_autores>
12 </autores_libros_pendientes>
```

```
1 Resultado, 346 b Resultado
```

```
<autores_libros_pendientes>
  <autor>
    <apellido>Abiteboul</apellido>
    <nombre>Serge</nombre>
  </autor>
  <autor>
    <apellido>Buneman</apellido>
    <nombre>Peter</nombre>
  </autor>
  <autor>
    <apellido>Suciu</apellido>
    <nombre>Dan</nombre>
  </autor>
  <total_autores>3</total_autores>
</autores_libros_pendientes>
```

3. EJERCICIO 3. (utiliza eXist en un programa JAVA)

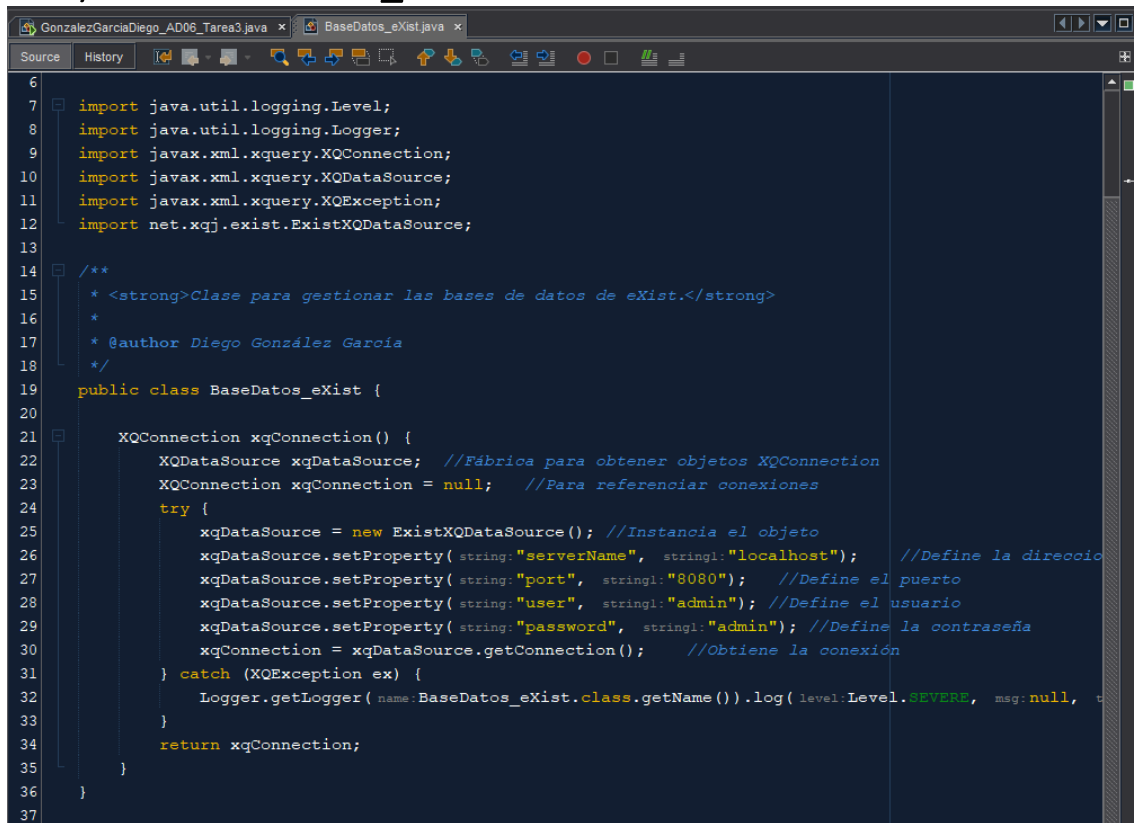
a) Programa principal.



```
7 import javax.xml.xquery.XQConnection;
8 import javax.xml.xquery.XQException;
9 import javax.xml.xquery.XQExpression;
10 import javax.xml.xquery.XQResultSequence;
11
12 /**
13  * <strong>Clase para el método main.</strong>
14  *
15  * @author Diego González García
16  */
17 public class GonzalezGarciaDiego_AD06_Tarea3 {
18
19     /**
20      * @param args the command line arguments
21      */
22     public static void main(String[] args) {
23         BaseDatos_eXist baseDatos_eXist;
24         XQConnection xqConnection = null; //Para la conexión con la BD
25         XQExpression xqExpression = null; //Para la ejecución inmediata de sentencias
26         XQResultSequence xqResultSequence = null; //Para los resultados de la consulta
27         String consulta = "for $x in collection(/ejercicios)/bib/libro"
28             + " return $x/titulo";
29
30         try {
31             baseDatos_eXist = new BaseDatos_eXist(); //Crea el objeto para la base de datos
32             xqConnection = baseDatos_eXist.xqConnection(); //Toma la conexión creada
33             if (xqConnection == null) {
34                 throw new IllegalArgumentException("Fallo al conectar con eXist.");
35             }
36             xqExpression = xqConnection.createExpression(); //Instancia un objeto de expresión
37             xqResultSequence = xqExpression.executeQuery(string:consulta); //Ejecuta la consulta
38             System.out.println("Libros del fichero 'libros.xml'\n"
39                 + "-----");
40             while (xqResultSequence.next()) { //Recorre los resultados obtenidos
41                 System.out.println(xqResultSequence.getItemAsString(prpts:null)); //Muestra el resultado
42             }
43         } catch (XQException xqe) {
44             xqe.getMessage();
45         } finally {
46             try {
47                 xqResultSequence.close();
48                 xqExpression.close();
49                 xqConnection.close();
50             } catch (XQException xqe) {
51                 xqe.getMessage();
52             }
53         }
54     }
55 }
```

Output - GonzalezGarciaDiego_AD06_Tarea3 (run)

```
run:
Libros del fichero 'libros.xml'
-----
<titulo>TCP/IP Illustrated</titulo>
<titulo>Advan Programming for Unix environment</titulo>
<titulo>Data on the Web</titulo>
<titulo>Economics of Technology for Digital TV</titulo>
BUILD SUCCESSFUL (total time: 0 seconds)
```

b) Clase *BaseDatos_eXist*.

```
6
7 import java.util.logging.Level;
8 import java.util.logging.Logger;
9 import javax.xml.xquery.XQConnection;
10 import javax.xml.xquery.XQDataSource;
11 import javax.xml.xquery.XQException;
12 import net.xqj.exist.ExistXQDataSource;
13
14 /**
15  * <strong>Clase para gestionar las bases de datos de eXist.</strong>
16  *
17  * @author Diego González García
18  */
19 public class BaseDatos_eXist {
20
21     XQConnection xqConnection() {
22         XQDataSource xqDataSource; //Fábrica para obtener objetos XQConnection
23         XQConnection xqConnection = null; //Para referenciar conexiones
24         try {
25             xqDataSource = new ExistXQDataSource(); //Instancia el objeto
26             xqDataSource.setProperty( string:"serverName", stringl:"localhost"); //Define la direccio
27             xqDataSource.setProperty( string:"port", stringl:"8080"); //Define el puerto
28             xqDataSource.setProperty( string:"user", stringl:"admin"); //Define el usuario
29             xqDataSource.setProperty( string:"password", stringl:"admin"); //Define la contraseña
30             xqConnection = xqDataSource.getConnection(); //Obtiene la conexión
31         } catch (XQException ex) {
32             Logger.getLogger( name:BaseDatos_eXist.class.getName() ).log( level:Level.SEVERE, msg:null, t
33         }
34         return xqConnection;
35     }
36 }
37
```