

Antonio Jiménez Sevilla

Tarea AD06

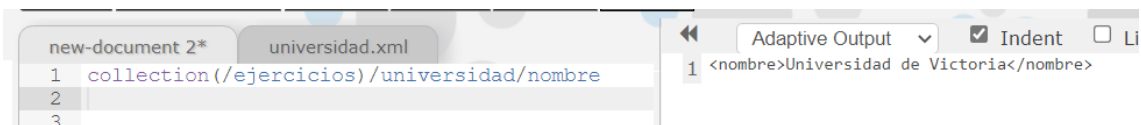
Utilizando la base de datos XML , crear una la colección ejercicios y en ella sube los documentos universidad.xml , libros.xml y librosalmacen.xml.

EJERCICIO 1.- XPATH (universidad.xml)

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

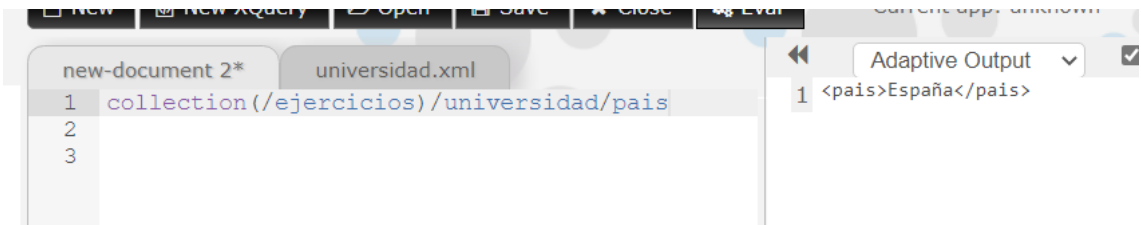
1. Nombre de la Universidad.

`collection(/ejercicios)/universidad/nombre`



2. País de la Universidad.

`collection(/ejercicios)/universidad/país`



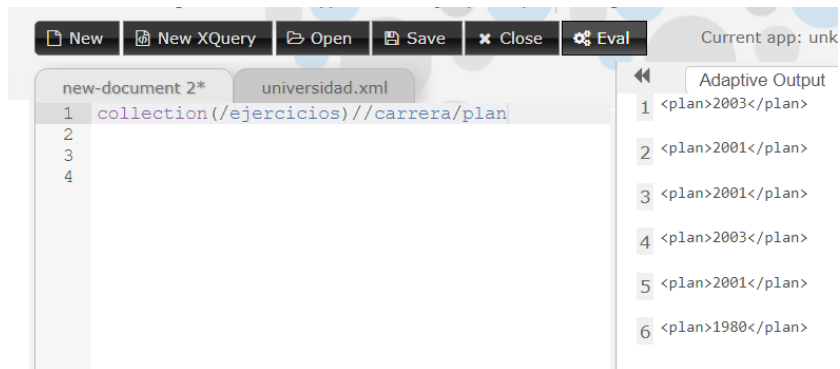
3. Nombres de las Carreras.

`collection(/ejercicios)/universidad/carreras/carrera/nombre`



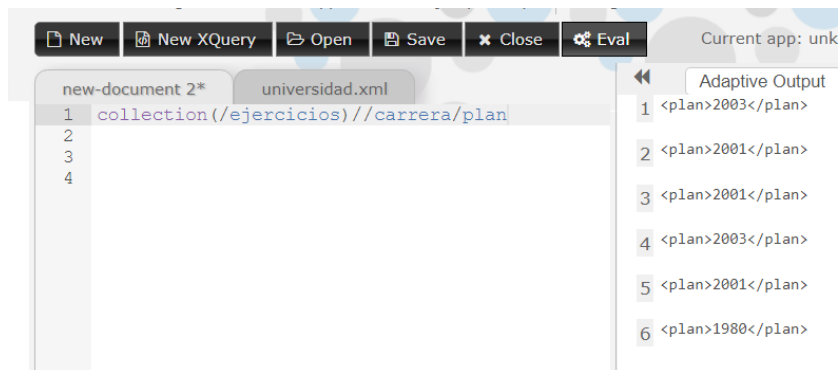
4. Años de plan de estudio de las carreras.

`collection(/ejercicios)//carrera/plan`



5. Nombres de todos los alumnos.

`collection(/ejercicios)//alumno/nombre`



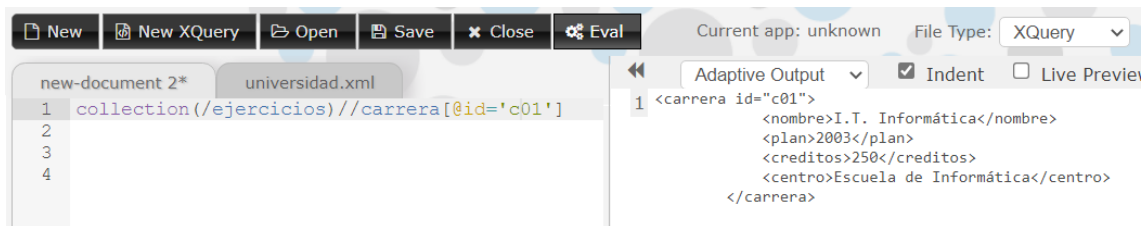
6. Identificadores de todas las carreras.

`collection(/ejercicios)//carrera/@id`



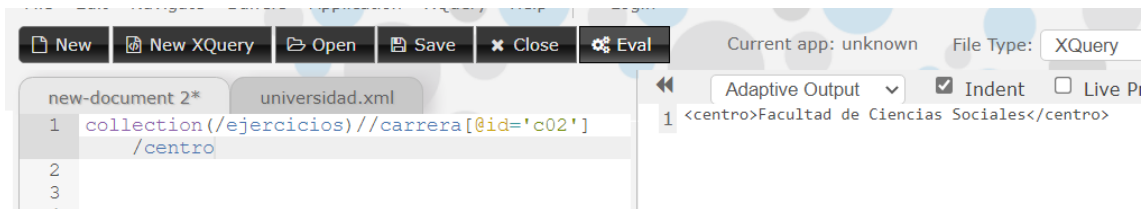
7. Datos de la carrera cuyo id es c0.

`collection(/ejercicios)//carrera[@id='c01']`



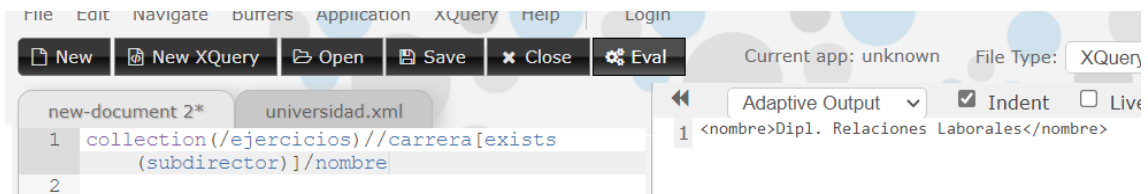
8. Centro en que se estudia la carrera cuyo id es c02.

`collection(/ejercicios)//carrera[@id='c02']/centro`



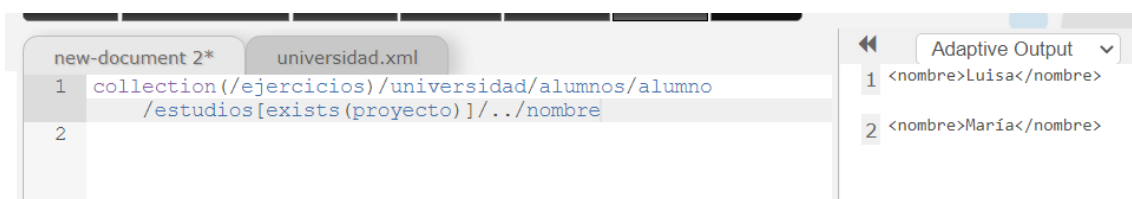
9. Nombre de las carreras que tengan subdirector.

`collection(/ejercicios)//carrera[exists(subdirector)]/nombre`



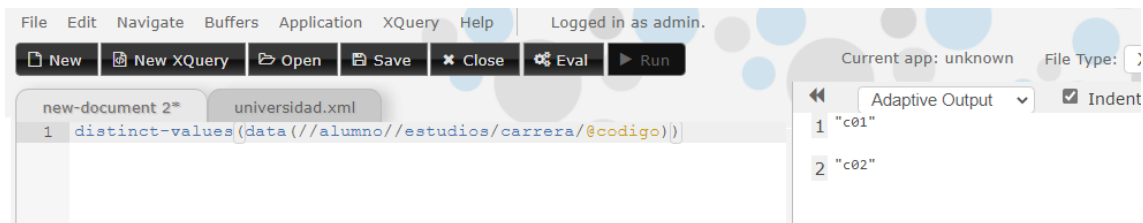
10. Nombre de los alumnos que estén haciendo proyecto.

`collection(/ejercicios)//alumno/estudios[exists(proyecto)]/../nombre`



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

`distinct-values(data(//alumno//estudios/carrera/@codigo))`



12. Apellidos y Nombre de los alumnos con beca.

`collection(/ejercicios)//alumno[@beca]/apellido1 | //alumno[@beca]/apellido2
|//alumno[@beca]/nombre`



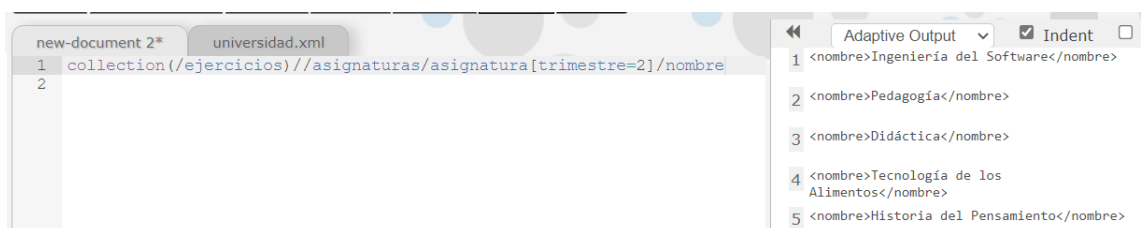
13. Nombre de las asignaturas de la titulación c04.

`collection(/ejercicios)//asignatura[@titulacion="c04"]/nombre`



14. Nombre de las asignaturas de segundo trimestre.

`collection(/ejercicios)//asignatura[trimestre=2]/nombre`



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

`collection(/ejercicios)//asignatura[creditos_teoricos!=4]/nombre`



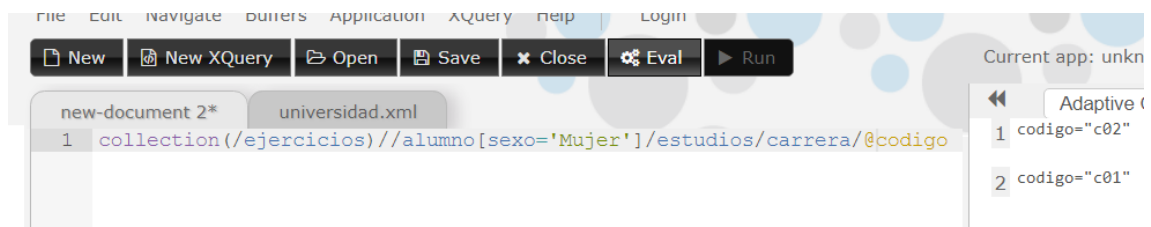
16. Código de la carrera que estudia el último alumno.

`collection(/ejercicios)//alumno[last()]/estudios/carrera/@codigo`



17. Código de las asignaturas que estudian mujeres.

`collection(/ejercicios)//alumno[sexo='Mujer']/estudios/carrera/@codigo`



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

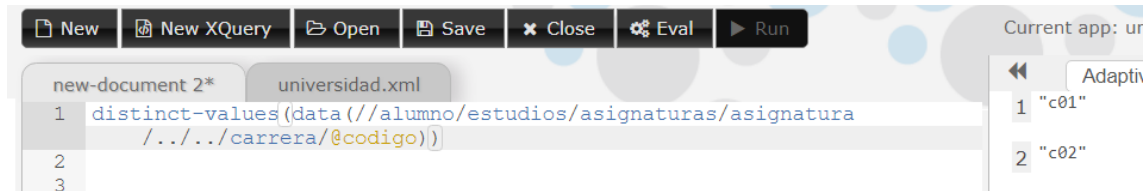
`collection(/ejercicios)//alumno[estudios/asignaturas/asignatura[@codigo='a02']]/nombre`



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

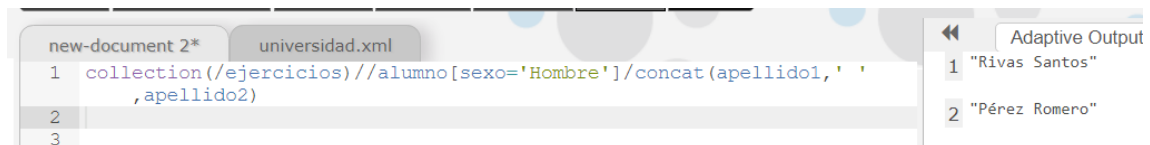
Distinct-

values(data([//alumno/estudios/asignaturas/asignatura/../../carrera/@codigo](#)))



20. Apellidos de todos los hombres.

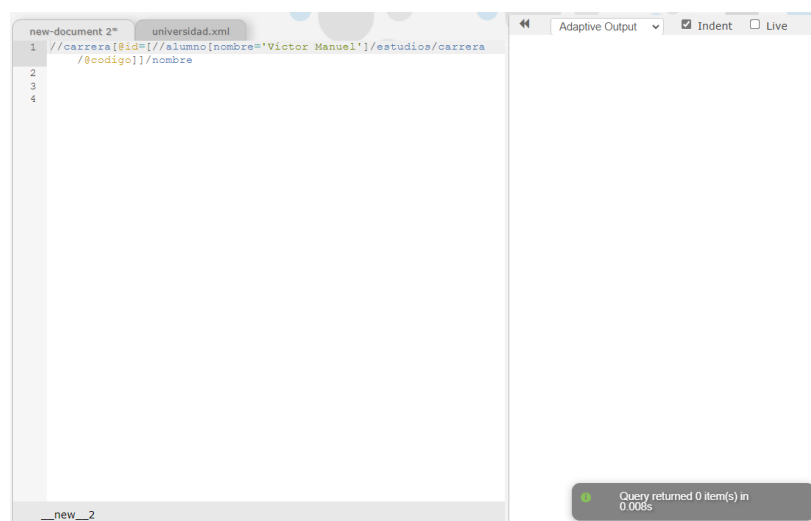
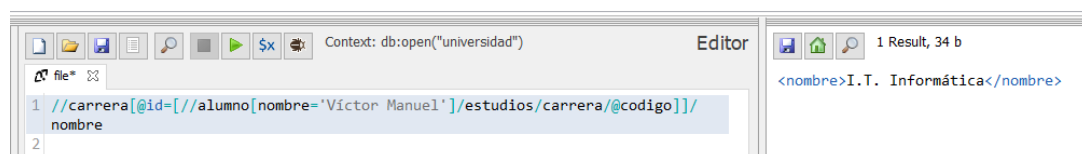
collection(/ejercicios)//alumno[sexo='Hombre']/concat(apellido1,' ',apellido2)



21. Nombre de la carrera que estudia Víctor Manuel.

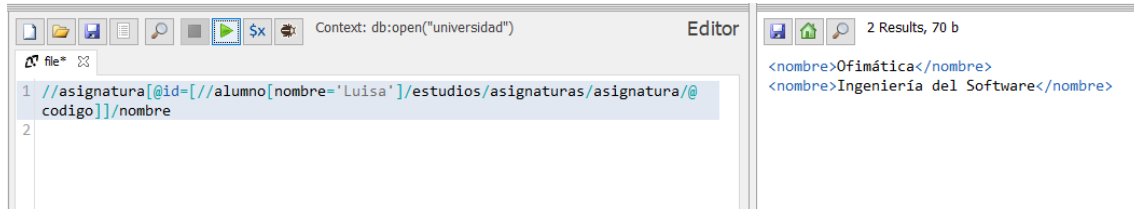
Aquí estoy usando BaseX nose por que no me saca las consultas, te pongo solo esta captura para que no saturar de algo todo el rato igual, no he conseguido que funcione por mucho que lo he intentado.

`//carrera[@id=[//alumno[nombre=' Víctor Manuel']//estudios/carrera/@codigo]]/nombre`



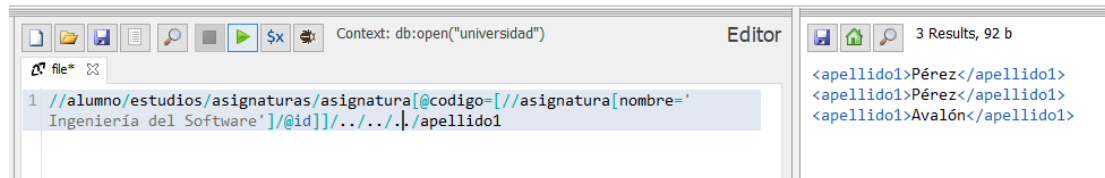
22. Nombre de las asignaturas que estudia Luisa.

//asignatura[@id=//alumno[nombre='Luisa']/estudios/asignaturas/asignatura/@codigo]/nombre



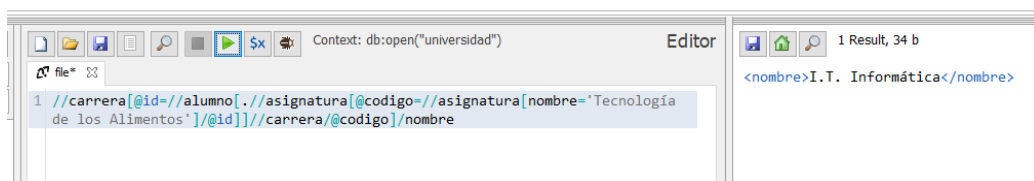
23. Primer apellido de los alumnos matriculados en Ingeniería del Software.

//alumno/estudios/asignaturas/asignatura[@codigo=//asignatura[nombre='Ingeniería del Software']/@id]/../../apellido1



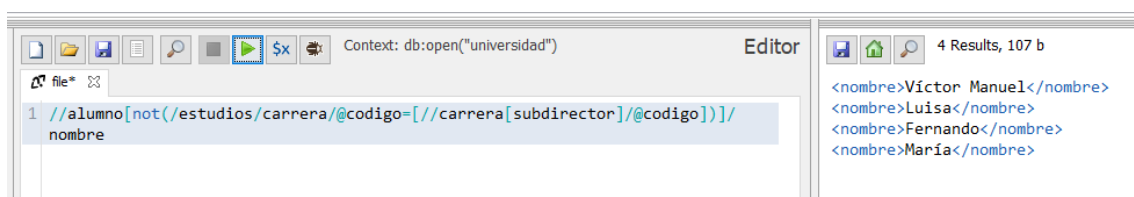
24. Nombre de las carreras que estudian los alumnos matriculados en la asignatura Tecnología de los Alimentos.

//carrera[@id=//alumno[.//asignatura[@codigo=//asignatura[nombre='Tecnología de los Alimentos']/@id]]/carrera/@codigo/nombre



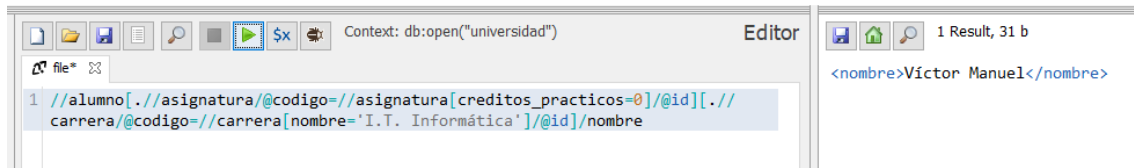
25. Nombre de los alumnos matriculados en carreras que no tienen subdirector.

//alumno[not(//carrera/@codigo=//carrera[subdirector]/@codigo)]/nombre



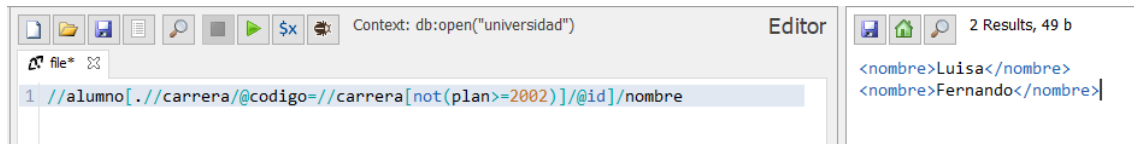
26. Nombre de los alumnos matriculados en asignaturas con 0 créditos prácticos y que estudien la carrera de I.T. Informática.

`//alumno[.//carrera/@codigo=//carrera[nombre='I.T. Informática']/@id][.//asignatura/@codigo=//asignatura[creditos_practicos=0]/@id]/@id]/nombre`



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

`//alumno[.//carrera/@codigo=//carrera[not(plan>=2002)]/@id]/nombre`

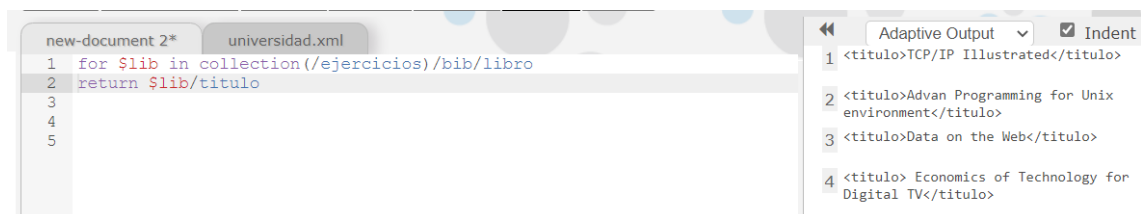


EJERCICIO 2. XQUERY (libros.xml y librosalmacen.xml)

1. Listar el título de todos los libros.

No estoy usando `for $lib in doc("libros.xml")//libro` así no escribo todo el rato ("libros.xml").

`for $lib in collection(/ejercicios)/bib/libro return lib/titulo`



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

```
for $lib in collection(/ejercicios)/bib/libro
order by $lib/año
return <libro>{$lib/@año} {$lib/titulo}</libro>
```



3. Listar los libros cuyo precio sea 65.95

Esta la tenía hecha de otra manera, las he dejado para que la veas, he cambiado casi todas, mucho más fácil como nos explicaste.

```
for $lib in collection(/ejercicios)/bib/libro[precio = 65.95] return $lib
```



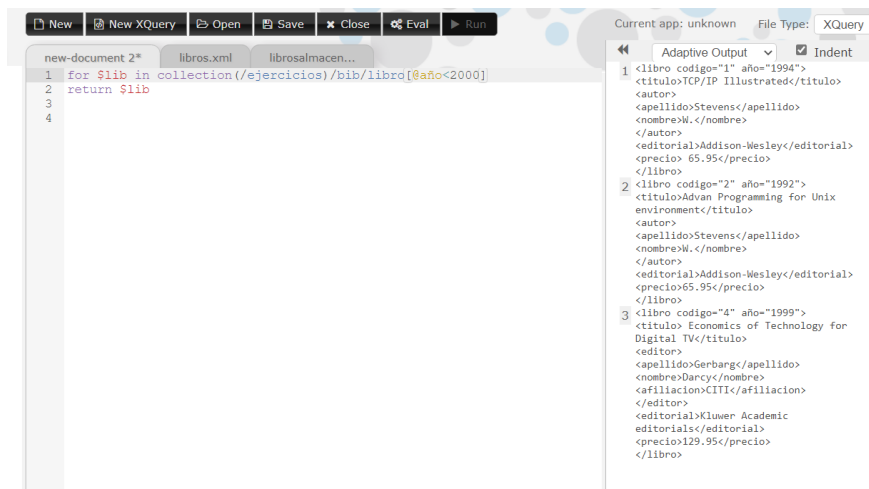
Esta es como la tenía antes

```
for $lib in collection(/ejercicios)/bib/libro where $lib/precio = 65.95 return $lib
```



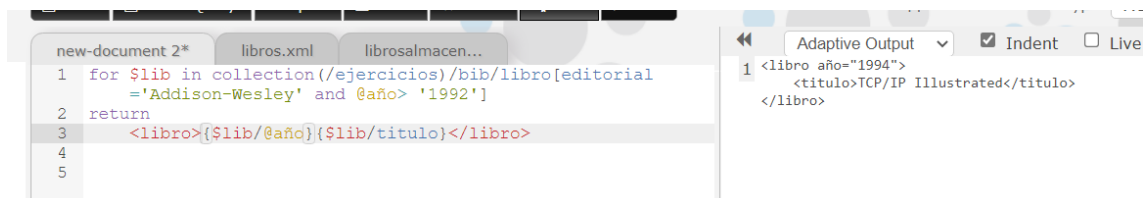
4. Listar los libros publicados antes del año 2000

for \$lib in collection(/ejercicios) /bib/libro[@año<2000] return \$lib



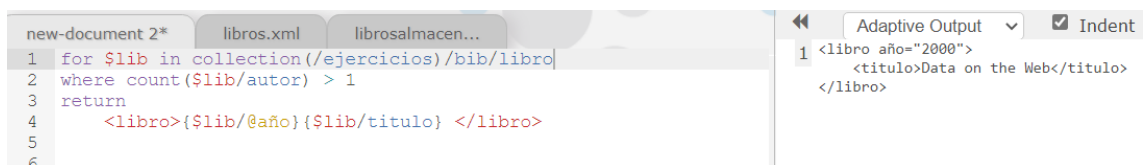
5. Listar año y título de los libros publicados por Addison-Wesley después del año 1992.

for \$lib in collection(/ejercicios) /bib/libro[editorial='Addison-Wesley' and
\$lib/@año>1992]
return <libro>{\$lib/@año}{\$lib/titulo}</libro>



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

for \$lib in collection(/ejercicios) /bib/libro where count(\$lib/autor) > 1
return <libro>{\$lib/@año}{\$lib/titulo} </libro>



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

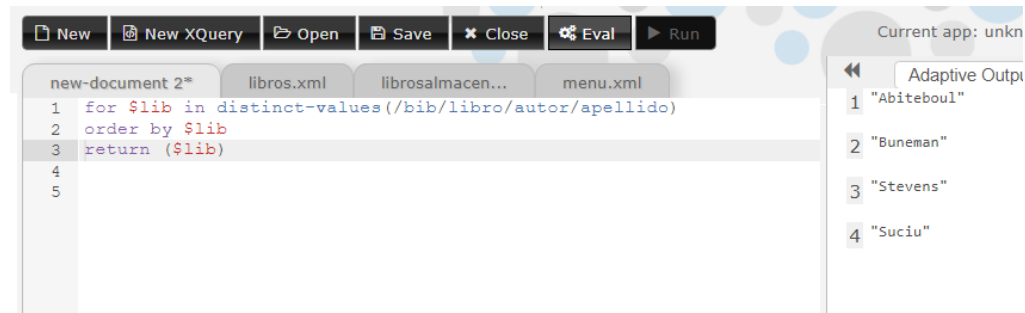
```
for $lib in collection(/ejercicios)/bib/lib where count($lib/autor) = 0  
return <libro>{$lib/@año}{$lib/titulo} </lib>
```

importante el = 0 no poner símbolos de mayor o menor resultados diferentes.



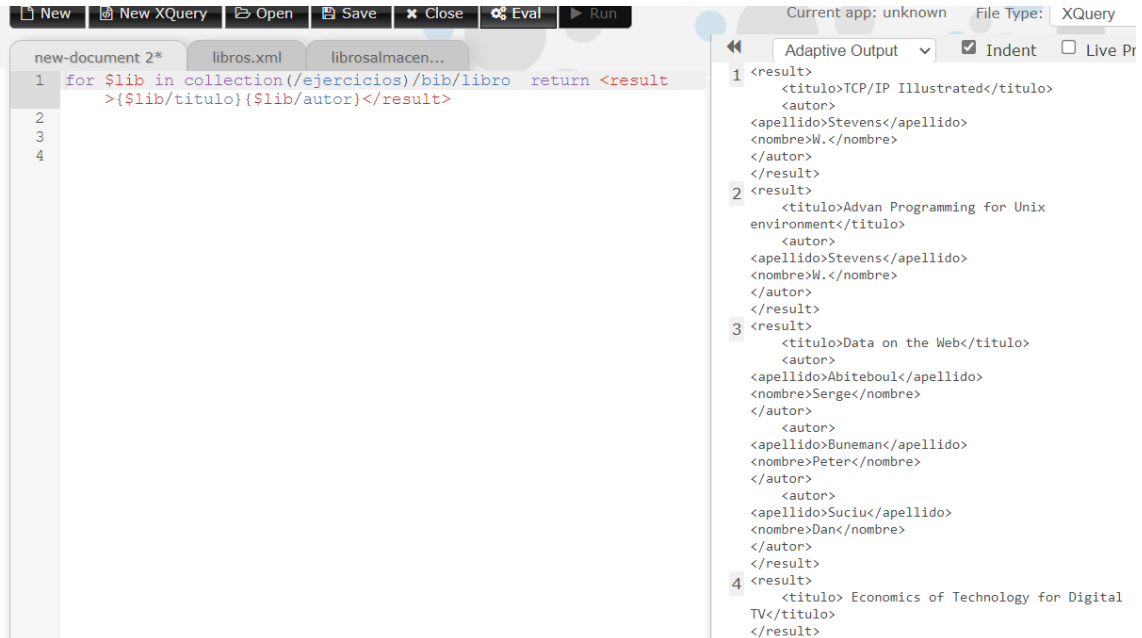
8. Mostrar los apellidos de los autores que aparecen en el documento, sin repeticiones, ordenados alfabéticamente.

```
for $lib in distinct-values(/bib/libro/autor/apellido)  
order by $lib  
return ($lib)
```



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

```
for $lib in collection(/ejercicios) /bib/libro
return <result>{$lib/titulo}{$lib/autor}</result>
```



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

```
for $lib in collection(/ejercicios) /bib/libro
return
<libro>
{$lib/titulo}
<numero_autores>{$lib/count(autor)}</numero_autores>
</libro>
```



11. Una lista ordenada alfabéticamente de títulos de libros comprados.

Modificamos el xml y colocamos los códigos para poder hacer la tarea. No consigo relacionarlas por mucho que intento, y hacer lo que explicaste.

```
for $lib in collection(/ejercicios) /bib/libro[codigo <= 2]/titulo
order by $lib
return $lib
```



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

```
Let $lib := collection(/ejercicios)/bib/libro[@codigo>=3]
```

```
return <total>{sum($lib/precio)}</total>
```

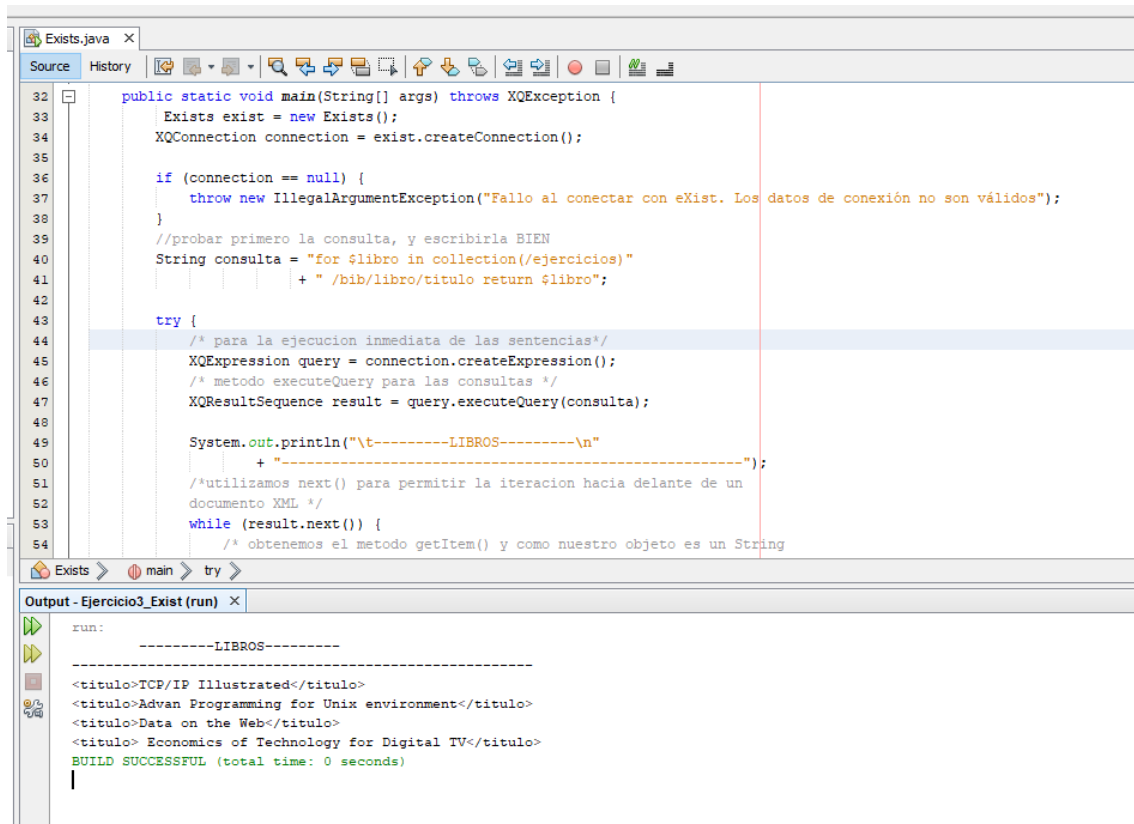


13. Una lista ordenada de autores que tengan libros pendientes. La última línea contendrá una línea que tenga el total de autores.

```
for $li in collection(/ejercicios) /bib/libros[@codigo>=3]
let $lib := count(/lib/libro/autor)
order by $lib
return <salida> {&lib}{ $li/autor} </salida>
```



Ejercicio 3



The screenshot shows an IDE with a Java file named `Exists.java`. The code is as follows:

```
32 public static void main(String[] args) throws XQException {
33     Exists exist = new Exists();
34     XQConnection connection = exist.createConnection();
35
36     if (connection == null) {
37         throw new IllegalArgumentException("Fallo al conectar con eXist. Los datos de conexión no son válidos");
38     }
39     //probar primero la consulta, y escribirla BIEN
40     String consulta = "for $libro in collection(/ejercicios)"
41                     + " /bib/libro/titulo return $libro";
42
43     try {
44         /* para la ejecucion inmediata de las sentencias*/
45         XQExpression query = connection.createExpression();
46         /* metodo executeQuery para las consultas */
47         XQResultSetSequence result = query.executeQuery(consulta);
48
49         System.out.println("\t-----LIBROS-----\n"
50                         + "-----");
51         /*utilizamos next() para permitir la iteracion hacia delante de un
52         documento XML */
53         while (result.next()) {
54             /* obtenemos el metodo getItem() y como nuestro objeto es un String
```

The IDE's breadcrumb shows the path: `Exists > main > try`. Below the code editor, the `Output - Ejercicio3_Exist (run)` window displays the following output:

```
run:
-----LIBROS-----
-----
<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)
```