

## Tarea para AD06.

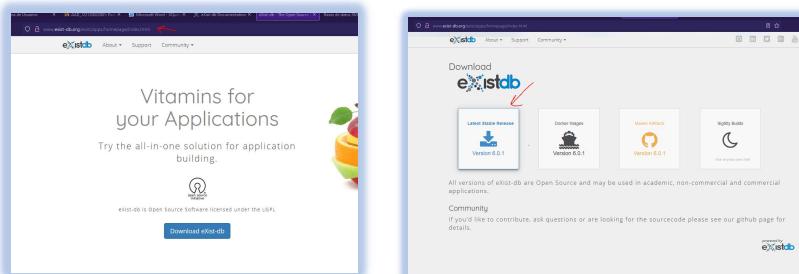
### Enunciado.

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

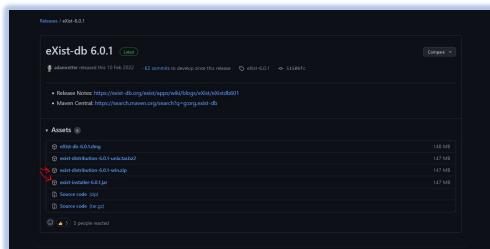
Los recursos necesarios para la elaboración de los ejercicios se encuentran en el enlace facilitado.

### Instalación Exist-DB

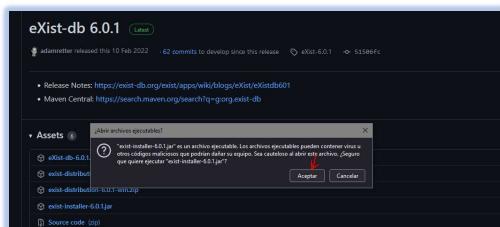
Vamos a la web oficial, donde nos ofrece la descarga para nuestro SO.



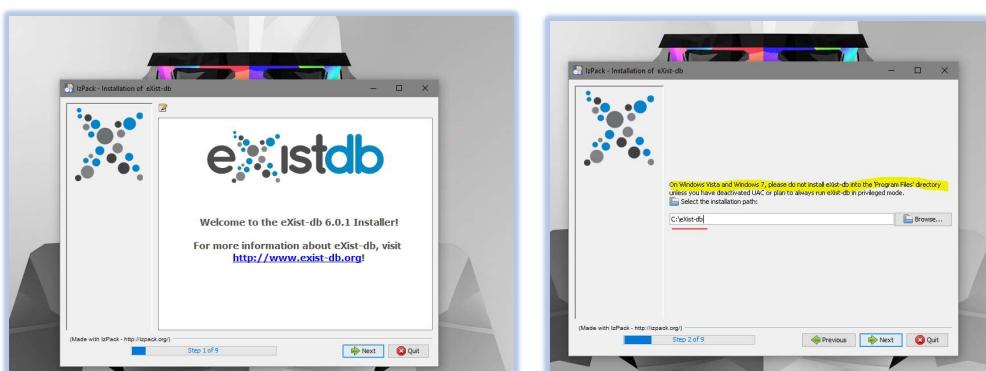
Nos da la posibilidad de descargar un paquete o hacerlo través de un archivo **.jar**, que será lo que usaré por ser como lo marca el temario.



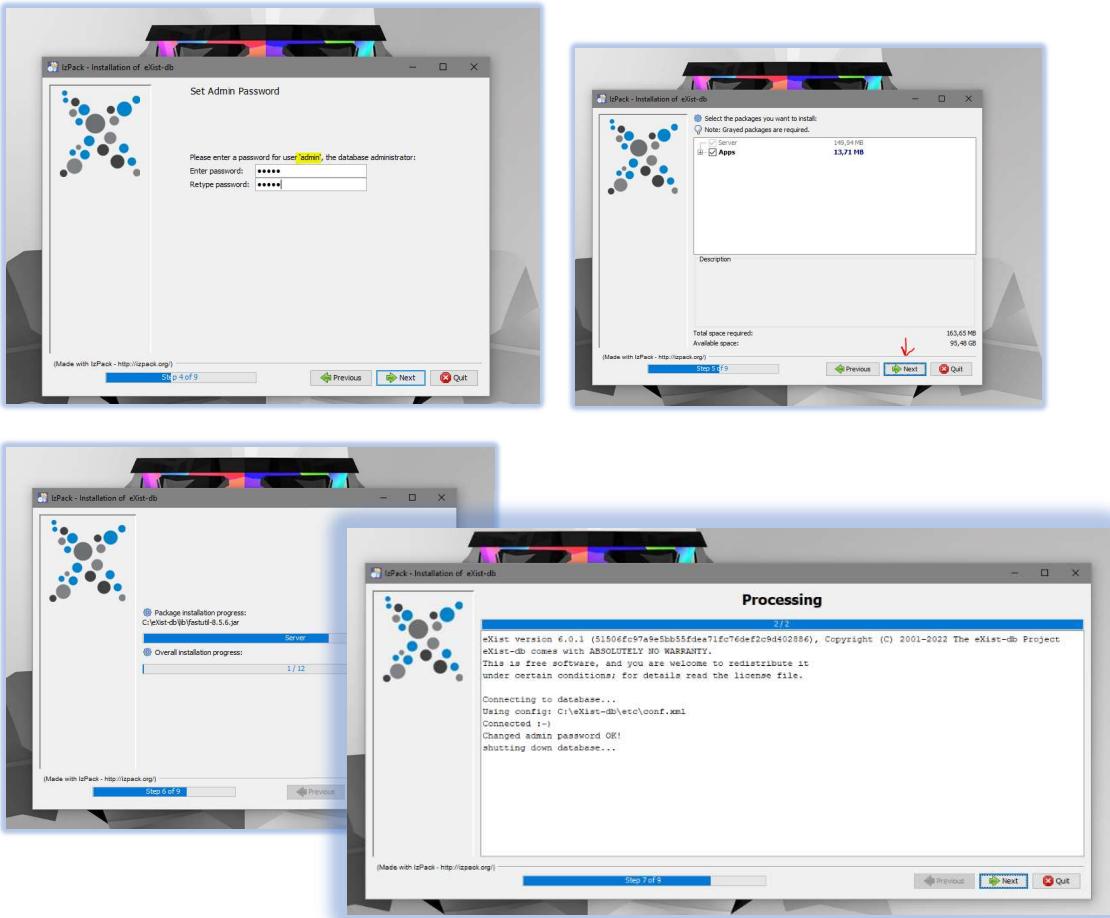
Aceptamos



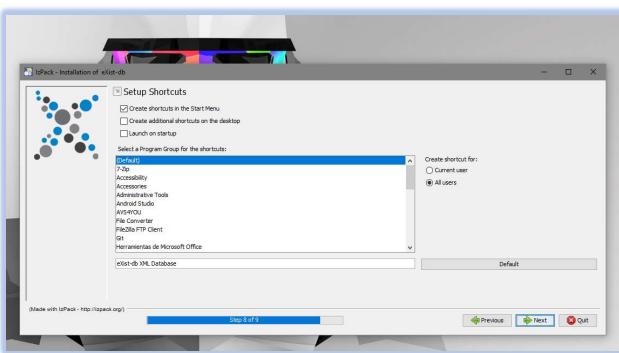
Comienza la instalación



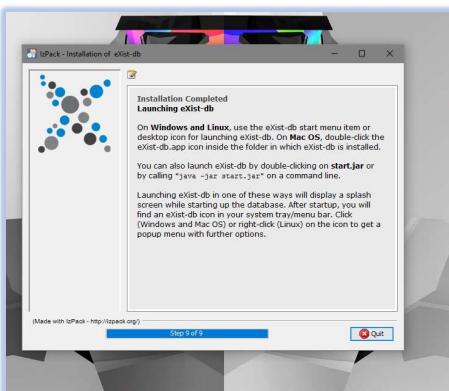
Le añadimos una contraseña e indicamos los paquetes a instalar.



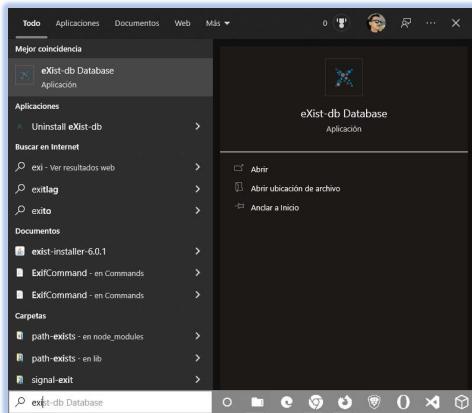
Seguimos personalizando



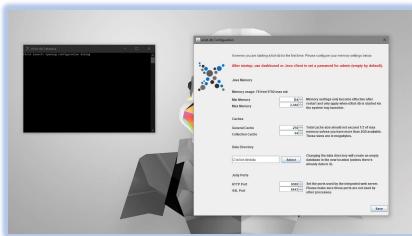
Al terminar, nos indica las fórmulas para acceder a nuestras nuevas herramientas.



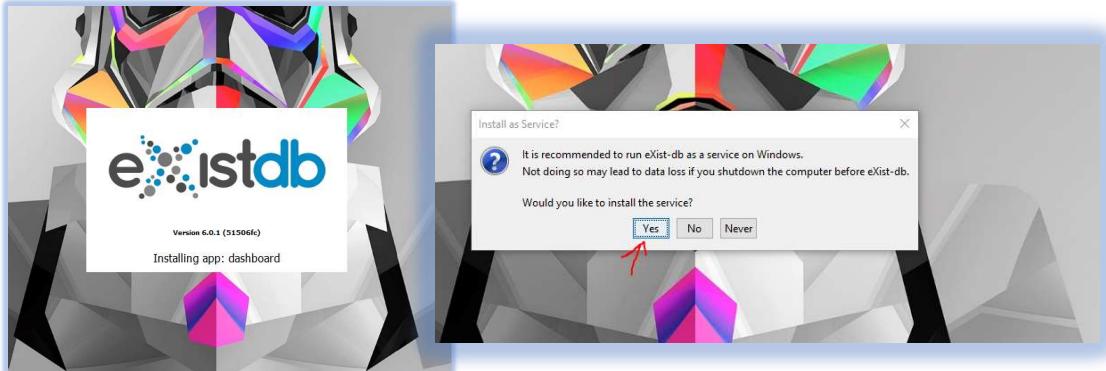
Abrimos desde nuestro escritorio



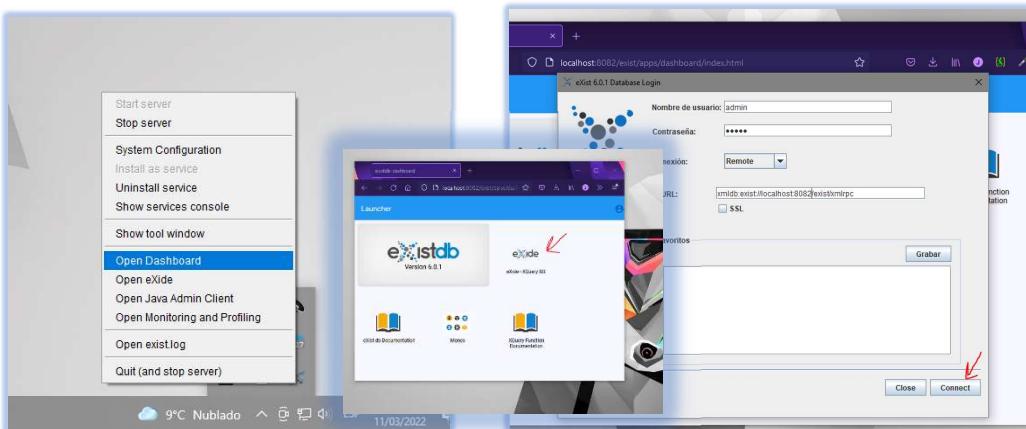
Para evitar conflictos, cambio el puerto del servidor, por ejemplo, a [8082](#)



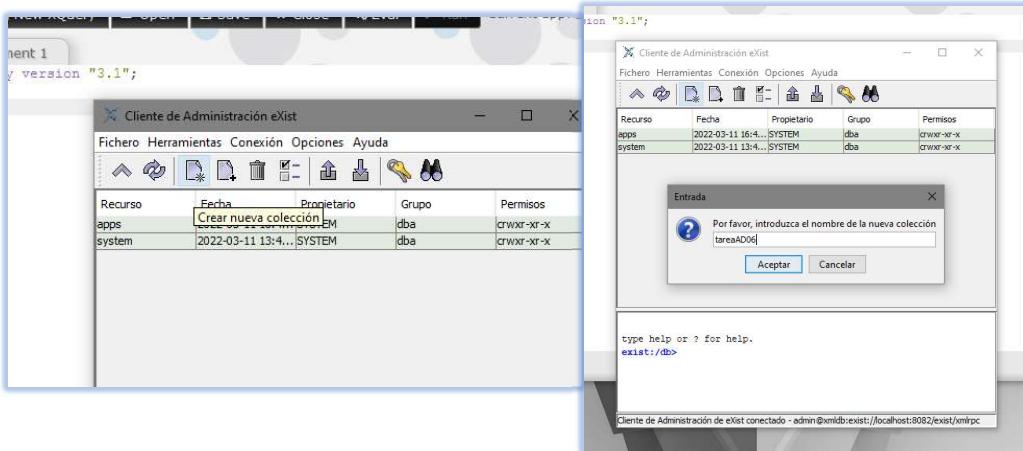
Inicia la instalación de apps y aceptamos que corra como servicio.



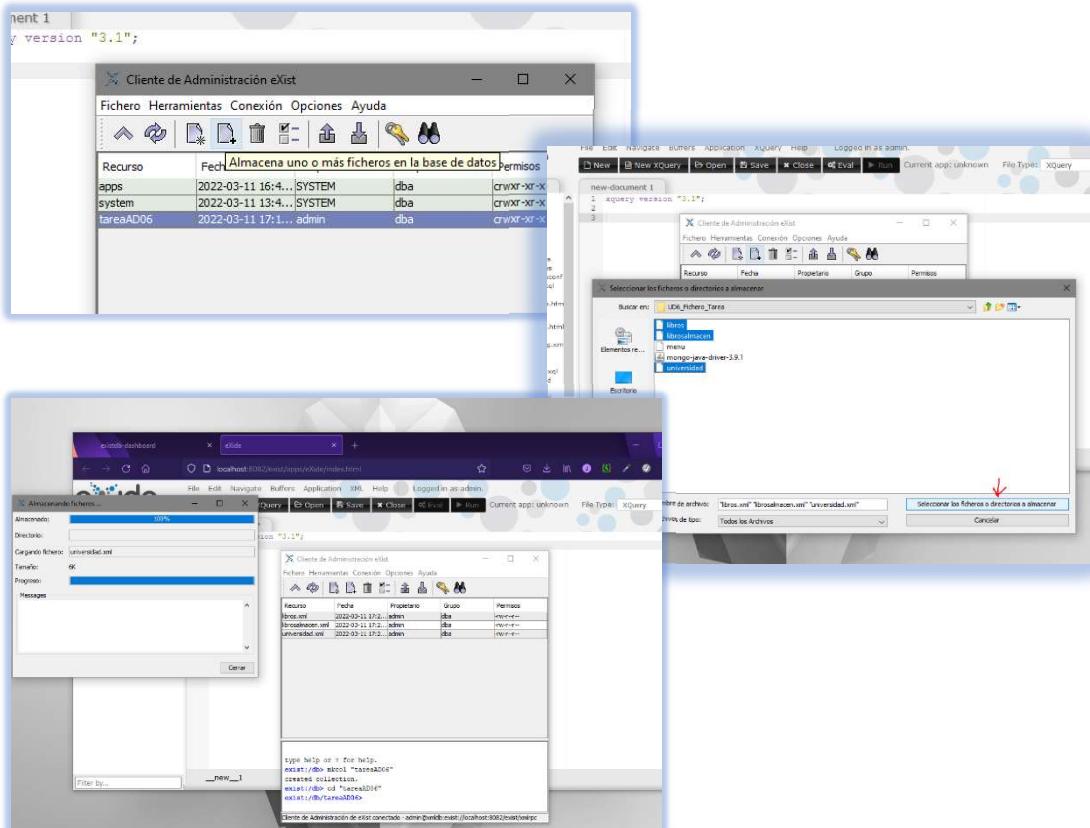
Desde la barra de herramientas podemos ver el [launcher](#) activo. Con el menú secundario, abrimos nuestro [dashboard](#). Con el mismo método abrimos desde el menú secundario el cliente de Administración.



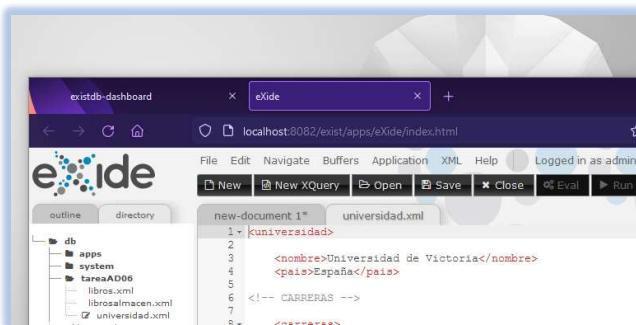
Creamos una nueva colección como se pide en el enunciado.



Haciendo doble click sobre la colección, se accede a ella, y se pueden añadir los ficheros.



Podemos verlos ya añadidos tanto desde el mismo cliente como abriendo la aplicación eXide.



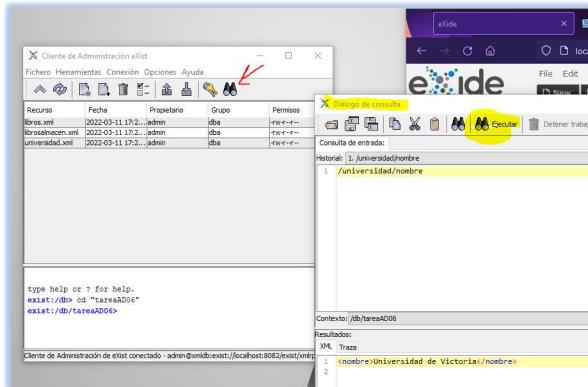
Paso, entonces, al ejercicio 1.

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

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

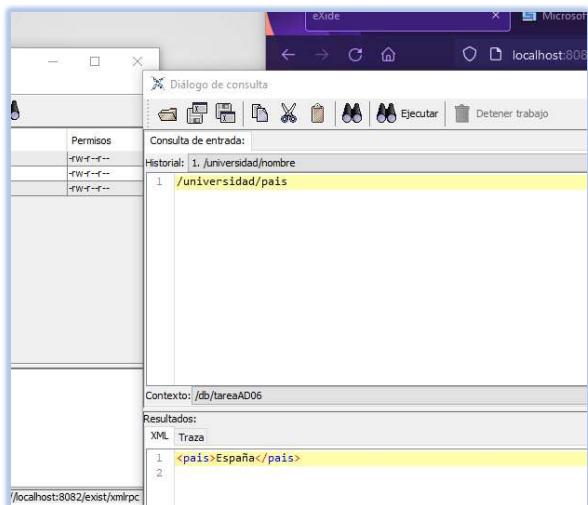
1. Nombre de la Universidad.

/universidad/nombre



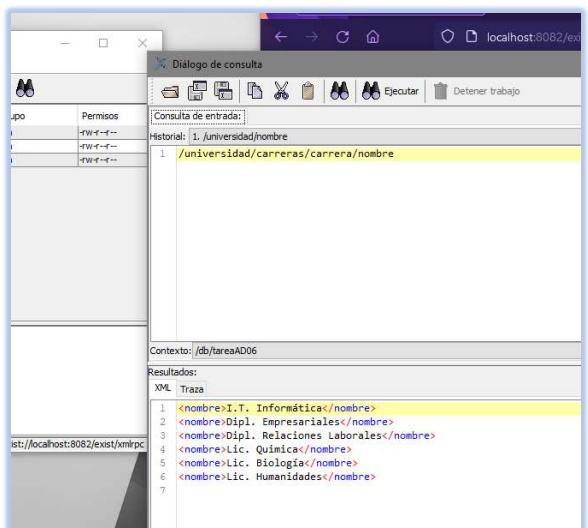
2. País de la Universidad.

/universidad/país



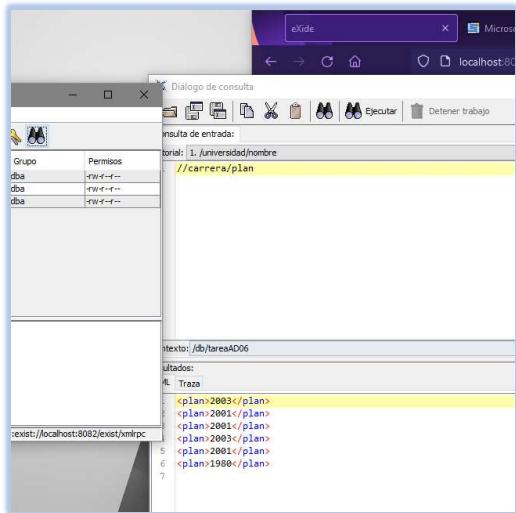
3. Nombres de las Carreras.

/universidad/carreras/carrera/nombre



#### 4. Años de plan de estudio de las carreras.

La misma consulta del punto anterior, pero pidiendo el plan, o  
//carrera/plan

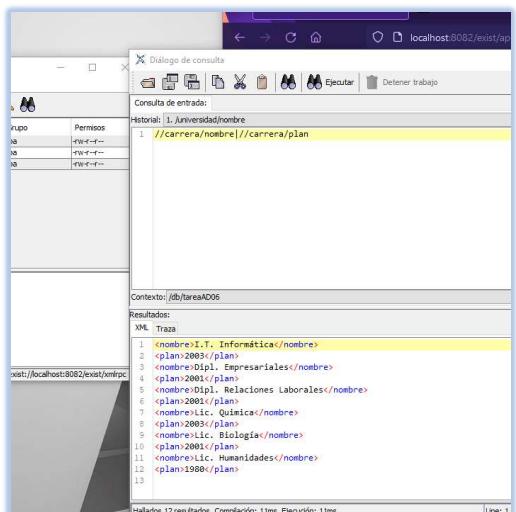


The screenshot shows the eXide IDE interface with the following details:

- Consulta de entrada:** Historial: 1. Universidad/nombre  
//carrera/plan
- Contexto:** /db/tareaAD06
- Resultados:** XML, Traza
- Output (Traza):**

```
1 <plan>2003</plan>
2 <plan>2001</plan>
3 <plan>2001</plan>
4 <plan>2003</plan>
5 <plan>2001</plan>
6 <plan>1998</plan>
```

Si queremos mostrar los datos con mayor claridad:  
//carrera/nombre||//carrera/plan



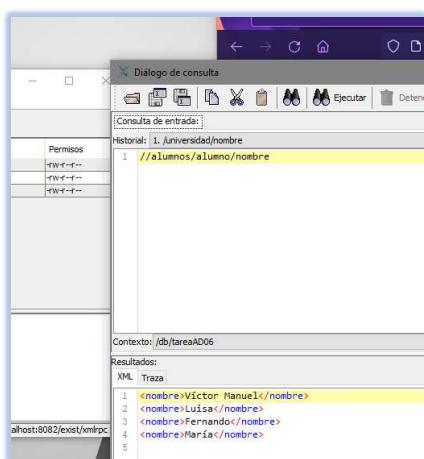
The screenshot shows the eXide IDE interface with the following details:

- Consulta de entrada:** Historial: 1. Universidad/nombre  
//carrera/nombre||//carrera/plan
- Contexto:** /db/tareaAD06
- Resultados:** XML, Traza
- Output (Traza):**

```
1 <nombre>I.T. Informática</nombre>
2 <plan>2003</plan>
3 <nombre>Dipl. Empresariales</nombre>
4 <plan>2001</plan>
5 <nombre>Dipl. Relaciones Laborales</nombre>
6 <plan>2001</plan>
7 <nombre>Lic. Químicas</nombre>
8 <plan>2003</plan>
9 <nombre>Lic. Biología</nombre>
10 <plan>2001</plan>
11 <nombre>Lic. Humanidades</nombre>
12 <plan>1998</plan>
```

#### 5. Nombres de todos los alumnos.

//alumnos/alumno/nombre



The screenshot shows the eXide IDE interface with the following details:

- Consulta de entrada:** Historial: 1. Universidad/nombre  
//alumnos/alumno/nombre
- Contexto:** /db/tareaAD06
- Resultados:** XML, Traza
- Output (Traza):**

```
1 <nombre>Victor Manuel</nombre>
2 <nombre>Luisa</nombre>
3 <nombre>Fernando</nombre>
4 <nombre>María</nombre>
```

## 6. Identificadores de todas las carreras.

/universidad/carreras/carrera/@id/string()

The screenshot shows the Oracle SQL Developer XQuery dialog. The 'Consulta de entrada' field contains the query `/universidad/carreras/carrera/@id/string()`. The 'Resultados' pane shows the results in XML format, specifically the trace, which lists six items: c01, c02, c03, c04, c05, and c06.

## 7. Datos de la carrera cuyo id es c0.

No hay ninguna carrera con ese id como puede verse en la consulta anterior y que muestro en este detalle

The screenshot shows the Oracle SQL Developer XQuery dialog. The 'Consulta de entrada' field contains the query `8. /universidad/carreras/carrera/@id/string`. The 'Resultados' pane shows the results in XML format, specifically the trace, which lists two items: c01 and c02. A red arrow points to the 'c01' entry.

Por ello, he pensado en hacer la consulta con el primer id disponible, ya que si lo hago con el cero, el resultado se mostrará vacío.

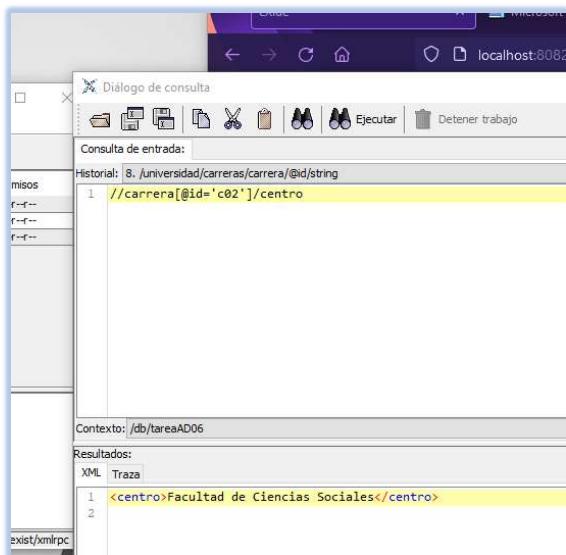
`/universidad/carreras/carrera[@id='c01']`

The screenshot shows the Oracle SQL Developer XQuery dialog. The 'Consulta de entrada' field contains the query `1. /universidad/carreras/carrera[@id='c01']`. The 'Resultados' pane shows the results in XML format, specifically the trace, which displays the XML structure of the first career, identified by id 'c01'. The XML output is as follows:

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

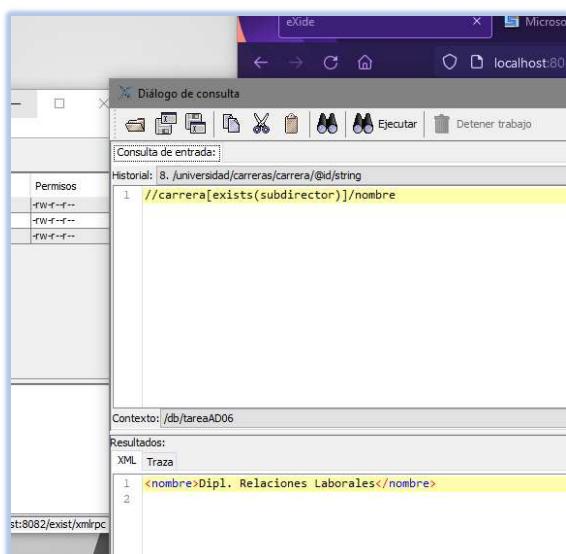
8. Centro en que se estudia de la carrera cuyo id es c02.

```
//carrera[@id='c02']/centro
```



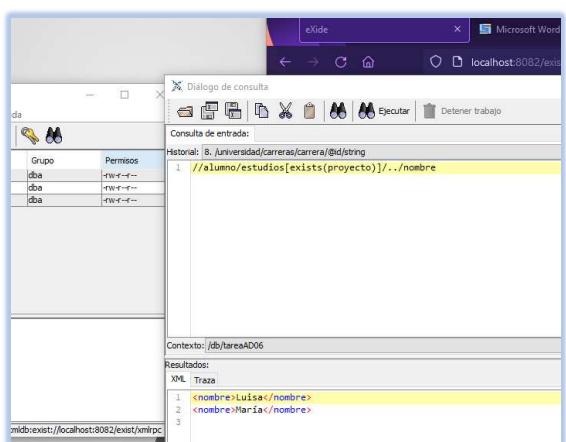
9. Nombre de las carreras que tengan subdirector.

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



10. Nombre de los alumnos que estén haciendo proyecto.

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



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

Desde **xpather** (no desde el cliente de **eXist**) nos deja con la siguiente consulta:  
`//carrera[@id=/alumno/estudios/carrera/@codigo]/@id`

The screenshot shows the xpather interface with the following details:

- Consulta de entrada:** `//carrera[@id=/alumno/estudios/carrera/@codigo]/@id`
- Contexto:** /db
- Resultados:** XML | Traza
- Historial:** 1. //carrera[@id=/alumno/estudios/carrera/@codigo]/@id
- Elementos:** 1. c01  
2. c02
- XML mode:** Format | Save

The XML output pane shows the following structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<universidad>
  <nOMBRE>Universidad de Victoria</nOMBRE>
  <pAIS>España</pAIS>
  <!-- CARRERAS -->
  <carreras>
```

## 12. Apellidos y Nombre de los alumnos con beca.

`//alumno[@beca]/apellido1 | //alumno[@beca]/apellido2 | //alumno[@beca]/nombre`

The screenshot shows the eXide interface with the following details:

- Consulta de entrada:** `//alumno[@beca]/apellido1 | //alumno[@beca]/apellido2 | //alumno[@beca]/nombre`
- Contexto:** /db/tareaAD06
- Resultados:** XML | Traza
- Historial:** 8. /universidad/carreras/carrera/@id/string
- Elementos:** 1. <apellido1>Pérez</apellido1>  
2. <apellido2>Romero</apellido2>  
3. <nombre>Fernando</nombre>  
4.
- alhost:8082/exist/xmlepc**

## 13. Nombre de las asignaturas de la titulación c04.

`//asignatura[@titulacion="c04"]/nombre`

The screenshot shows the eXide interface with the following details:

- Consulta de entrada:** `//asignatura[@titulacion="c04"]/nombre`
- Contexto:** /db/tareaAD06
- Resultados:** XML | Traza
- Historial:** 8. /universidad/carreras/carrera/@id/string
- Elementos:** 1. <nombre>Pedagogía</nombre>  
2. <nombre>Tecnología de los Alimentos</nombre>  
3.
- alhost:8082/exist/xmlepc**

14. Nombre de las asignaturas de segundo trimestre.

```
//asignatura[trimestre=2]/nombre
```

The screenshot shows the eXide interface with the following details:

- Consulta de entrada:** Historial: 8. /universidad/carreras/carrera/@id/string  
1 //asignatura[trimestre=2]/nombre
- Contexto:** /db/tareaAD06
- Resultados:** XML Traza  
1 <nombre>Ingeniería del Software</nombre>  
2 <nombre>Pedagogía</nombre>  
3 <nombre>Didáctica</nombre>  
4 <nombre>Tecnología de los Alimentos</nombre>  
5 <nombre>Historia del Pensamiento</nombre>

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

```
//asignatura[creditos_teoricos!=4]/nombre
```

The screenshot shows the eXide interface with the following details:

- Consulta de entrada:** Historial: 8. /universidad/carreras/carrera/@id/string  
1 //asignatura[creditos\_teoricos!=4]/nombre
- Contexto:** /db/tareaAD06
- Resultados:** XML Traza  
1 <nombre>Ofimática</nombre>  
2 <nombre>Ingeniería del Software</nombre>  
3 <nombre>Tecnología de los Alimentos</nombre>  
4 <nombre>Bases de Datos</nombre>  
5 <nombre>Historia del Pensamiento</nombre>

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

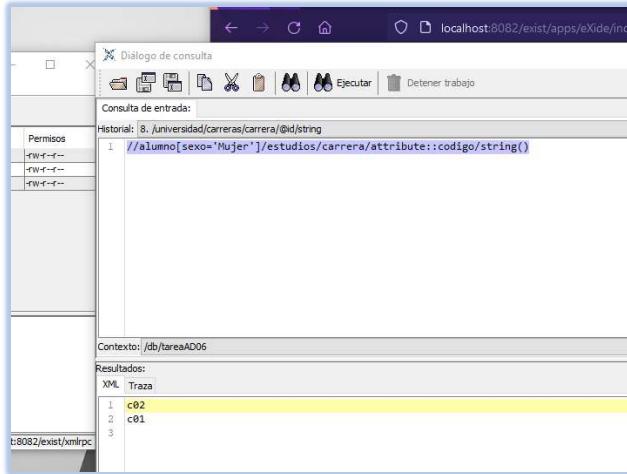
```
//alumno[last()]/estudios/carrera/attribute::codigo/string()
```

The screenshot shows the eXide interface with the following details:

- Consulta de entrada:** Historial: 8. /universidad/carreras/carrera/@id/string  
1 //alumno[last()]/estudios/carrera/attribute::codigo/string()
- Contexto:** /db/tareaAD06
- Resultados:** XML Traza  
1 c01  
2

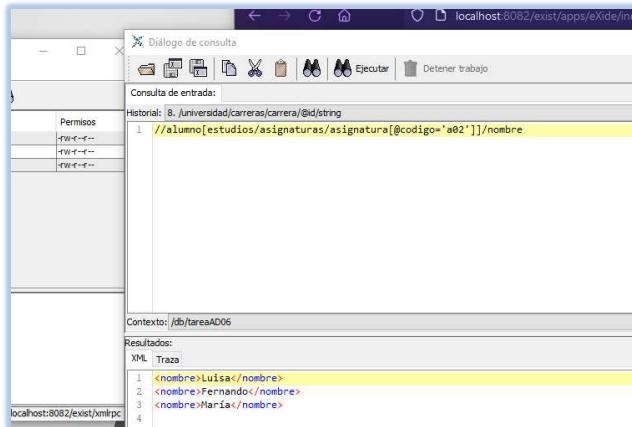
### 17. Código de las asignaturas que estudian mujeres.

```
//alumno[sexo='Mujer']/estudios/carrera/attribute::codigo/string()
```



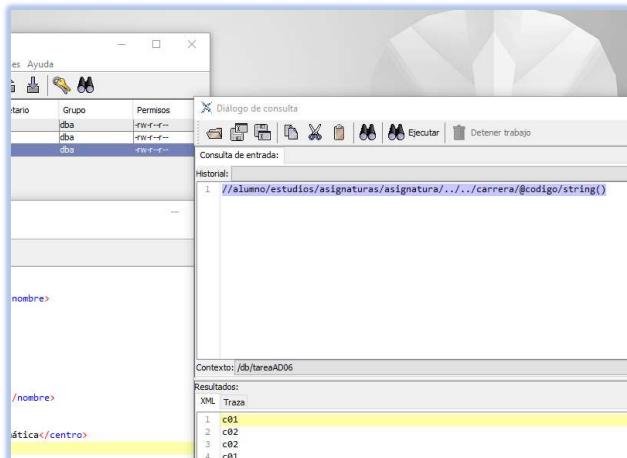
### 18. Nombre de los alumnos que matriculados en la asignatura a02.

```
//alumno/estudios/asignaturas/asignatura[@codigo='a02']/nombre
```



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

```
//alumno/estudios/asignaturas/asignatura/../../carrera/@codigo/string()
```



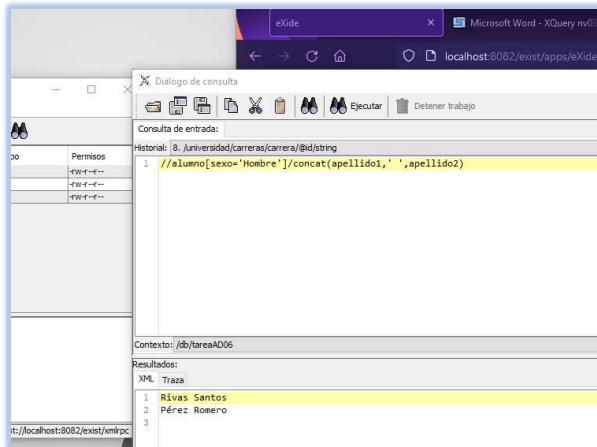
Revisando el material de apoyo, al ver que tengo valores repetidos, veo que la consulta puede mejorarse con:

**distinct-**

```
values(data//alumno/estudios/asignaturas/asignatura/../../carrera/@codigo))
```

## 20. Apellidos de todos los hombres.

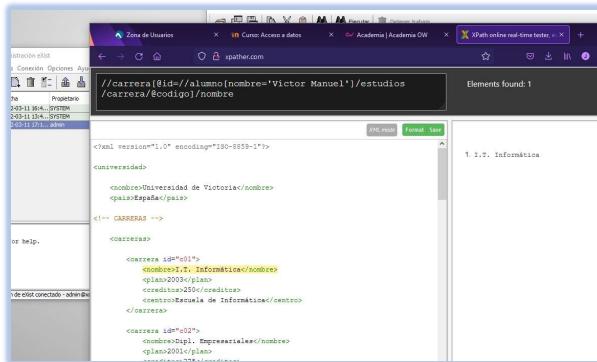
```
//alumno[sexo='Hombre']/concat(apellido1, ' ', apellido2)
```



## 21. Nombre de la carrera que estudia Víctor Manuel.

Desde [xpather](#) (no desde el cliente de [eXist](#)) nos deja con la siguiente consulta:

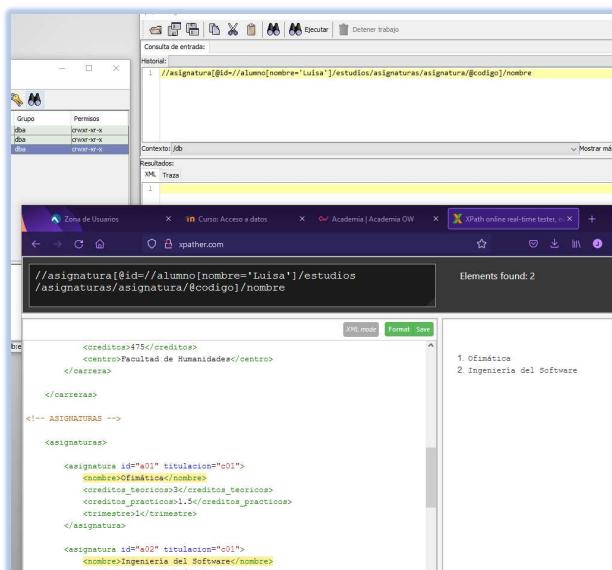
```
//carrera[@id=//alumno[nombre='Víctor  
Manuel']/estudios/carrera/@codigo]/nombre
```



## 22. Nombre de las asignaturas que estudia Luisa.

Aquí la consulta vuelve solamente a funcionar desde [xpather](#):

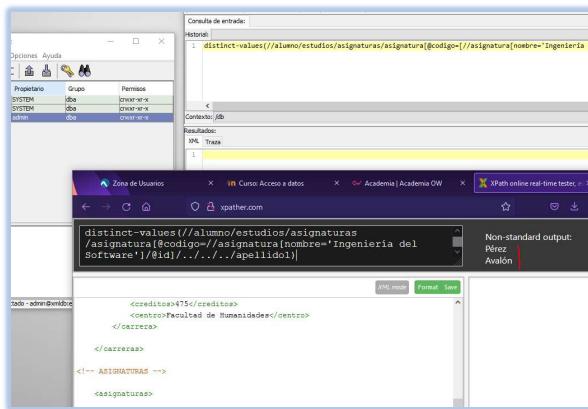
```
//asignatura[@id=//alumno[nombre='Luisa']/estudios/asignaturas/asignatura/@co  
digo]/nombre
```



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

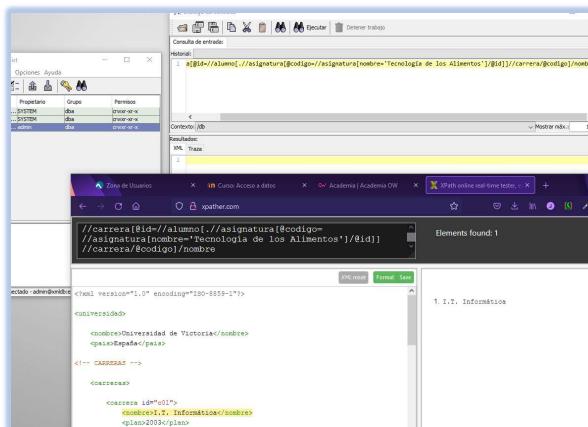
Misma pauta en las consultas, a diferencia que en este caso, al incluir una cláusula `distinct-values`, lo muestra fuera de la ventana `output standard`.

```
distinct-
values(//alumno/estudios/asignaturas/asignatura[@codigo=//asignatura[nombre='
Ingeniería del Software']/@id]/.../.../apellido1)
```



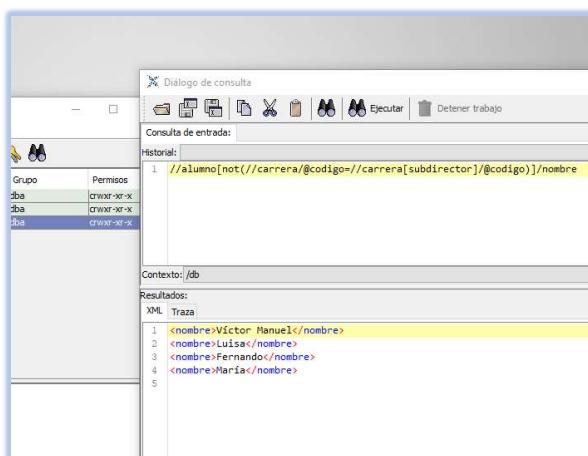
### 24. Nombre de las carreras que estudian los alumnos matriculados en la asignatura Tecnología de los Alimentos.

```
//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]/nombre
```

The screenshot shows the Xpather tool interface. In the 'Consulta de entrada' (Input Query) field, the query is displayed: `//alumno[./carrera/@codigo=//carrera[nombre='I.T. Informática']/@id][./asignatura/@codigo=//asignatura[creditos_practicos=0]/@id]/nombre`. The 'Resultados' (Results) pane shows the output: 'Elements found: 1' and '1. Victor Manuel'.

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

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

The screenshot shows the Xpather tool interface. In the 'Consulta de entrada' (Input Query) field, the query is displayed: `//alumno[./carrera/@codigo=//carrera[not(plan>=2002)]/@id]/nombre`. The 'Resultados' (Results) pane shows the output: 'Elements found: 2' and '1. Luisa' and '2. Pernando'.

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

**1. Listar el título de todos los libros.**

```
for $libro in /bib/libro return $libro/titulo
```

The screenshot shows the Xpather tool interface. In the 'Consulta de entrada' (Input Query) field, the query is displayed: `for $libro in /bib/libro return $libro/titulo`. The 'Resultados' (Results) pane shows the output: 'Elements found: 4' and the titles: '1. <titulo>TCP/IP Illustrated</titulo>', '2. <titulo>Advan Programming for Unix environment</titulo>', '3. <titulo>Data on the Web</titulo>', and '4. <titulo>Economics of Technology for Digital TV</titulo>'.

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

```
for $libro in /bib/libro order by $libro/año return <libro>{$libro/@año}
{$libro/titulo}</libro>
```

The screenshot shows the Oxygen XML Editor interface. On the left, there's a file tree labeled 'libros.xml' containing XML elements like 'bib', 'libro', 'año', 'titulo', 'autor', 'apellido', 'nombre', 'editorial', and 'precio'. The main area is titled 'Dialogo de consulta' with the query:
`for $libro in /bib/libro order by $libro/año return <libro>{$libro/@año}
{$libro/titulo}</libro>`

The results pane shows the output in XML format, listing books from 1992 to 1999, ordered by year.

## 3. Listar los libros cuyo precio sea 65.95

```
for $libro in /bib/libro where $libro/precio = 65.95 return $libro
```

The screenshot shows the Oxygen XML Editor interface. On the left, there's a file tree labeled 'libros.xml' containing XML elements like 'bib', 'libro', 'año', 'titulo', 'autor', 'apellido', 'nombre', 'editorial', and 'precio'. The main area is titled 'Dialogo de consulta' with the query:
`for $libro in /bib/libro where $libro/precio = 65.95 return $libro`

The results pane shows the output in XML format, listing books with a price of 65.95.

## 4. Listar los libros publicados antes del año 2000

```
for $libro in /bib/libro where $libro/@año < 2000 order by $libro/@año return $libro
```

The screenshot shows the Oxygen XML Editor interface. On the left, there's a file tree labeled 'libros.xml' containing XML elements like 'bib', 'libro', 'año', 'titulo', 'autor', 'apellido', 'nombre', 'editorial', and 'precio'. The main area is titled 'Dialogo de consulta' with the query:
`for $libro in /bib/libro where $libro/@año < 2000 order by $libro/@año return $libro`

The results pane shows the output in XML format, listing books published before 2000, ordered by year.

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

```
for $libro in /bib/libro
where $libro/editorial='Addison-Wesley' and $libro/@año>1992
order by $libro/@año
return <libro>{$libro/@año}{$libro/titulo}</libro>
```

The screenshot shows the XDK interface with the following details:

- File Explorer:** Shows the XML file 'libros.xml' with the query results highlighted in yellow.
- Console:** Displays the XQuery code for question 5.
- Results:** Shows the XML output for the query, which includes a single book entry for 'TCP/IP Illustrated' published in 1994.

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

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

The screenshot shows the XDK interface with the following details:

- File Explorer:** Shows the XML file 'libros.xml' with the query results highlighted in yellow.
- Console:** Displays the XQuery code for question 6.
- Results:** Shows the XML output for the query, which includes a single book entry for 'Data on the Web' published in 2000, which has multiple authors.

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

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

The screenshot shows the XDK interface with the following details:

- File Explorer:** Shows the XML file 'libros.xml' with the query results highlighted in yellow.
- Console:** Displays the XQuery code for question 7.
- Results:** Shows the XML output for the query, which includes a single book entry for 'Economics of Technology for Digital TV' published in 1999, which has no authors listed.

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

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

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

Resultados:

XML Traza

- 1 Abiteboul
- 2 Bumeman
- 3 Stevens
- 4 Suciu

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

```
for $libro in /bib/libro
return <result>{$libro/titulo}{$libro/autor}</result>
```

```
for $libro in /bib/libro
return <result>{$libro/titulo}{$libro/autor}</result>
```

Resultados:

XML Traza

- 1 <libro><titulo>TCP/IP Illustrated</titulo><autor>
- 2 <apellido>Stevens</apellido>
- 3 <nombre>W.</nombre>
- 4 </autor></libro>
- 5 <result>
- 6 <libro><titulo>TCP/IP Illustrated</titulo><autor>
- 7 <apellido>Bumeman</apellido>
- 8 <nombre>Serge</nombre>
- 9 </autor></libro>
- 10 <result>
- 11 <libro><titulo>TCP/IP Data on the Web</titulo><autor>
- 12 <apellido>Peter</apellido>
- 13 <nombre>Peter</nombre>
- 14 </autor></libro>
- 15 <result>
- 16 <libro><titulo>Economics of Technology for Digital TV</titulo></libro>
- 17 </result>

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

```
for $libro in /bib/libro
return
<libro>{$libro/titulo}<num_autor>{$libro/count(autor)}</num_autor></libro>
```

```
for $libro in /bib/libro
return
<libro>{$libro/titulo}<num_autor>{$libro/count(autor)}</num_autor></libro>
```

Resultados:

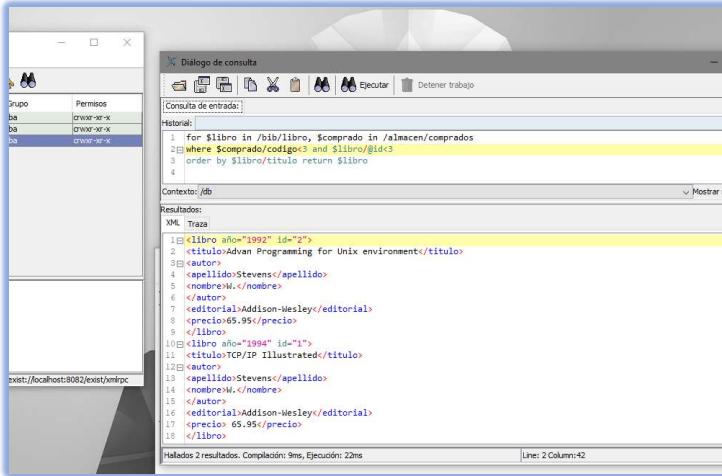
XML Traza

- 1 <libro><titulo>TCP/IP Illustrated</titulo><num\_autor>1</num\_autor></libro>
- 2 <libro><titulo>TCP/IP Data on the Web</titulo><num\_autor>1</num\_autor></libro>
- 3 <libro><titulo>Economics of Technology for Digital TV</titulo><num\_autor>0</num\_autor></libro>

## 11. Una lista ordenada alfabéticamente de categorías de libros comprados.

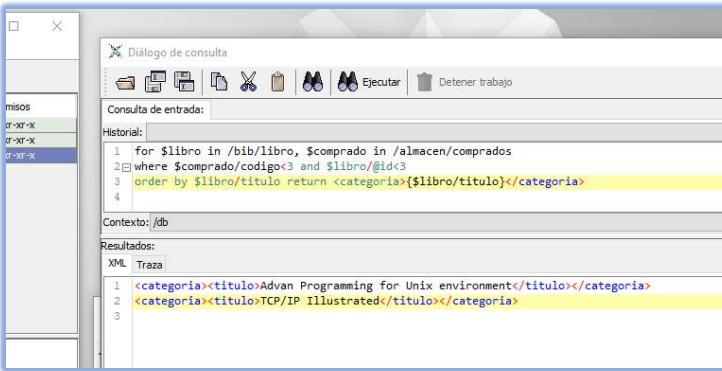
Me encuentro con que los libros, no tienen categorías. Por ello, asumo la premisa de ordenar la consulta por los títulos de los libros comprados, ya que comparten editorial, y por ello no tendría mucho sentido ordenarlos de forma alfabética.

```
for $libro in /bib/libro, $comprado in /almacen/comprados
where $comprado/codigo<3 and $libro/@id<3
order by $libro/titulo return $libro
```



Otra opción posible al enunciado, pero que muestra un resultado menos claro, puede ser indicando que nos los muestre en un elemento <categoría>

```
for $libro in /bib/libro, $comprado in /almacen/comprados
where $comprado/codigo<3 and $libro/@id<3
order by $libro/titulo return <categoría>{$libro/titulo}</categoría>
```



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

El resultado, maneja de forma extraña los datos, pues aunque la petición de precios, la recoge de forma aislada y correcta, luego el trato standard que da a los decimales, corrompe el resultado y no he conseguido ajustarlo más.

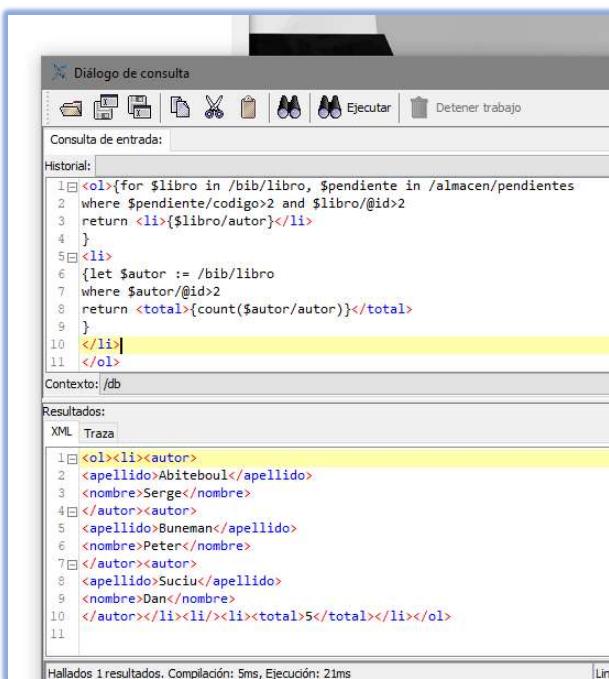
```
sum(for $libro in /bib/libro, $pendiente in /almacen/pendientes
where $pendiente/codigo>2 and $libro/@id>2
order by $libro/precio return $libro/precio)
```



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

El resultado, vuelve a manejar de forma extraña los datos, pero no he conseguido ajustarlo más.

```
<ol>{for $libro in /bib/libro, $pendiente in /almacen/pendientes
where $pendiente/codigo>2 and $libro/@id>2
return <li>{$libro/autor}</li>
}
<li>
{let $autor := /bib/libro
where $autor/@id>2
return <total>{count($autor/autor)}</total>
}
</li>
</ol>
```



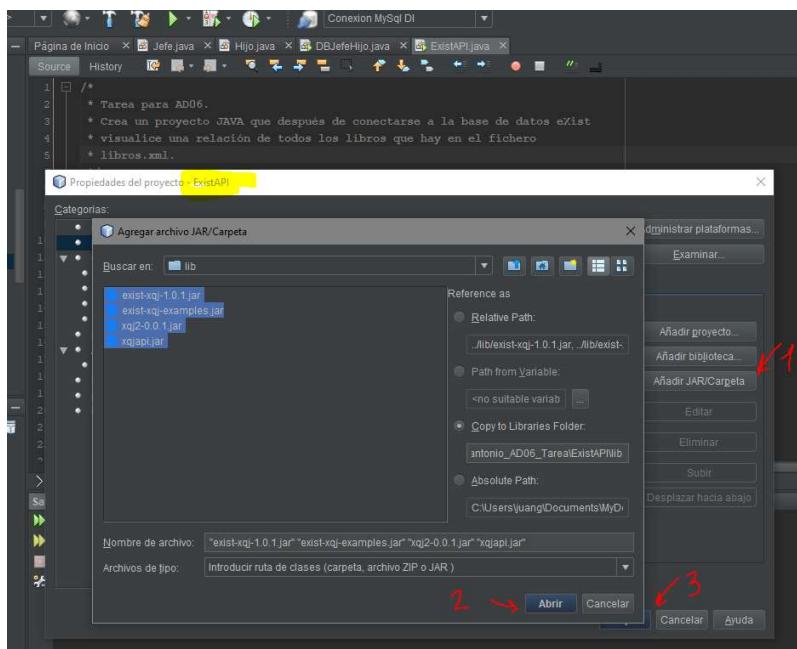
### EJERCICIO 3. (utiliza eXist en un programa JAVA).

Crea un proyecto JAVA que después de conectarse a la base de datos **eXist** visualice una relación de todos los libros que hay en el fichero **libros.xml**. Se enviará todo el proyecto comprimido.

El conector se adjunta al principio de la práctica.

Para la realización de este ejercicio he descargado el paquete de archivos necesarios desde el sitio de [XQJ API](#), pudiendo así tener acceso a BBDD XML nativas.

Lo siguiente ha sido crear un proyecto Java, que he llamado **ExistAPI**, e incluirlos en el mismo.



Sigo básicamente los criterios del ejemplo mostrado a partir del punto 3.7.2 del temario.

Como en tareas anteriores, intento dejar documentado el código que muestro a continuación para una clara interpretación del trabajo.

```
/*
 * Tarea para AD06.
 * Crea un proyecto JAVA que después de conectarse a la base de datos
 * eXist visualice una relación de todos los libros que hay en el
 * fichero
 * fichero libros.xml.
 */
package existapi;

import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQExpression;
import javax.xml.xquery.XQResultSequence;
import net.xqj.exist.ExistXQDataSource;

/**
 *
 * @author Juan A. García Muelas <juangmuelas@gmail.com>
 * @version 1
 * @since 15/03/22
 */
```

```

public class ExistAPI {

    /**
     * Creamos la lógica de la nueva conexión y sacamos así fuera las
     * posibles excepciones.
     */
    private XQConnection newConnection(){
        /**
         * Inicializamos un nuevo objeto de la interfaz XQConnection
         * y a continuación le aportamos los datos con la interfaz
         * XQDataSource.
         * En ella, añadimos los datos de conexión que tengamos en
         * nuestro servidor eXist-db
         */
        XQConnection conn = null;
        try {
            XQDataSource ds = new ExistXQDataSource();
            ds.setProperty("serverName", "localhost");
            ds.setProperty("port", "8082");
            ds.setProperty("user", "admin");
            ds.setProperty("password", "admin");

            //Con los datos recogidos, creamos conexión.
            conn = ds.getConnection();
        } catch (XQException e) {
            e.printStackTrace();
        }
        return conn;
    }

    public static void main(String[] args) {
        // Creamos un nuevo objeto y su conexión
        ExistAPI existTareaAD06 = new ExistAPI();
        XQConnection conn1 = existTareaAD06.newConnection();

        //Si tenemos un error en tiempo de ejecución lo recogemos.
        if (conn1 == null) {
            throw new IllegalArgumentException("eXist NO HA CONECTADO.
                                                Revise los parámetros");
        }

        /**
         * @param String consulta cadena que recoge la consulta
         * en XQuery.
         * Es la primer punto que hemos hecho en el ejercicio 2.
         */
        String consulta = "for $libro in /bib/libro return
                          $libro/titulo";

        /**
         * Preparamos la ejecución inmediata con un objeto
         * XQExpression.
         * Recogemos los datos obtenidos mediante un bucle while.
         * Envolvemos todo en un try-catch que recoja posibles
         * excepciones.
         */
        try {
            XQExpression query = conn1.createExpression();
            XQResultSequence result = query.executeQuery(consulta);

            System.out.println("\033[32m==== LISTADO DE LIBROS ====");
            System.out.println("\033[32m===== ===== ===== =====");
        }
    }
}

```

```

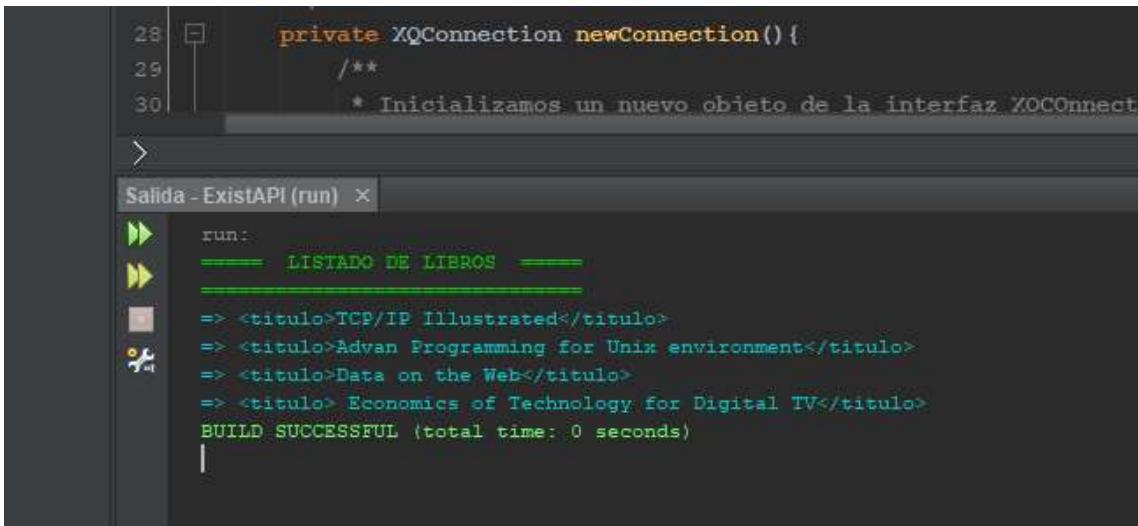
        while (result.next()) {
            System.out.println("\033[36m=> "
                + result.getItemAsString(null));
        }

        //cerramos las expresiones y liberamos recursos.
        query.close();
        result.close();

    } catch (XQException e) {
        e.printStackTrace();
    }
} //Fin main
}//Fin clase ExistAPI

```

Podemos comprobar su ejecución:



The screenshot shows an IDE interface with a code editor and a terminal window. The code editor displays Java code for a class named ExistAPI. The terminal window, titled 'Salida - ExistAPI (run)', shows the execution results. It starts with 'run:' followed by a list of book titles: 'TCP/IP Illustrated', 'Advanced Programming for Unix environment', 'Data on the Web', and 'Economics of Technology for Digital TV'. The output concludes with 'BUILD SUCCESSFUL (total time: 0 seconds)'.

```

28 |     private XQConnection newConnection(){
29 |         /**
30 |          * Inicializamos un nuevo objeto de la interfaz XQConnect
|      >
Salida - ExistAPI (run) x
run:
===== LISTADO DE LIBROS =====
=> <titulo>TCP/IP Illustrated</titulo>
=> <titulo>Advanced 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)
| 

```