
M6.UF3.A5.P1
PERSISTENCIA EN BD NATIVAS
XML

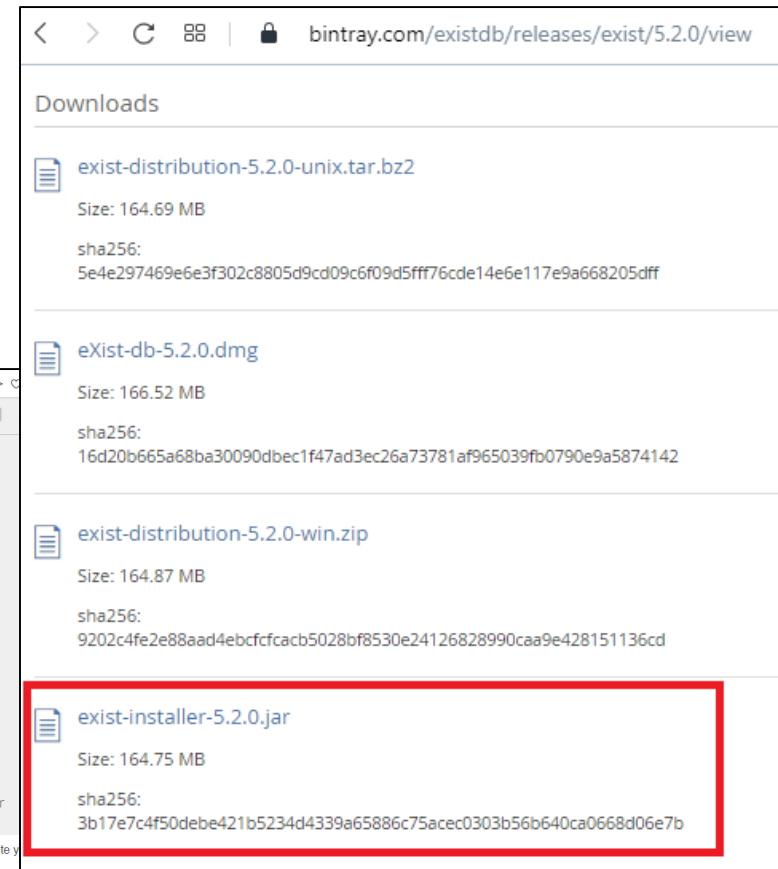
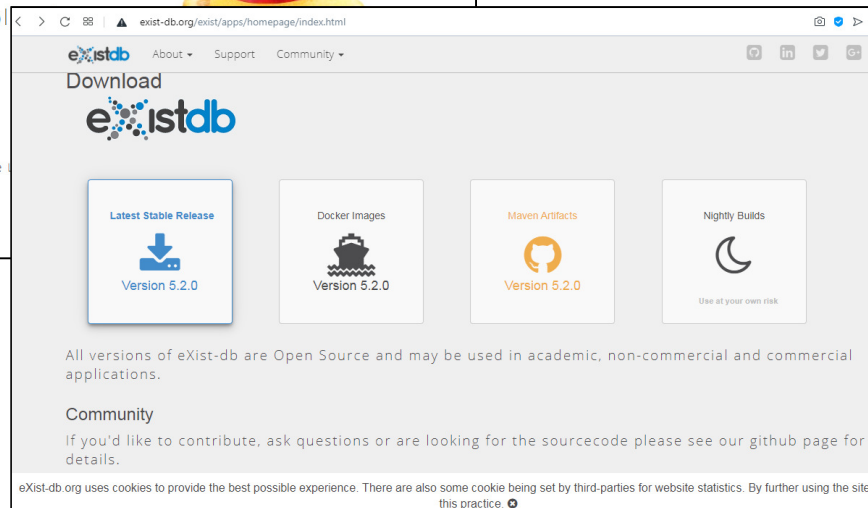
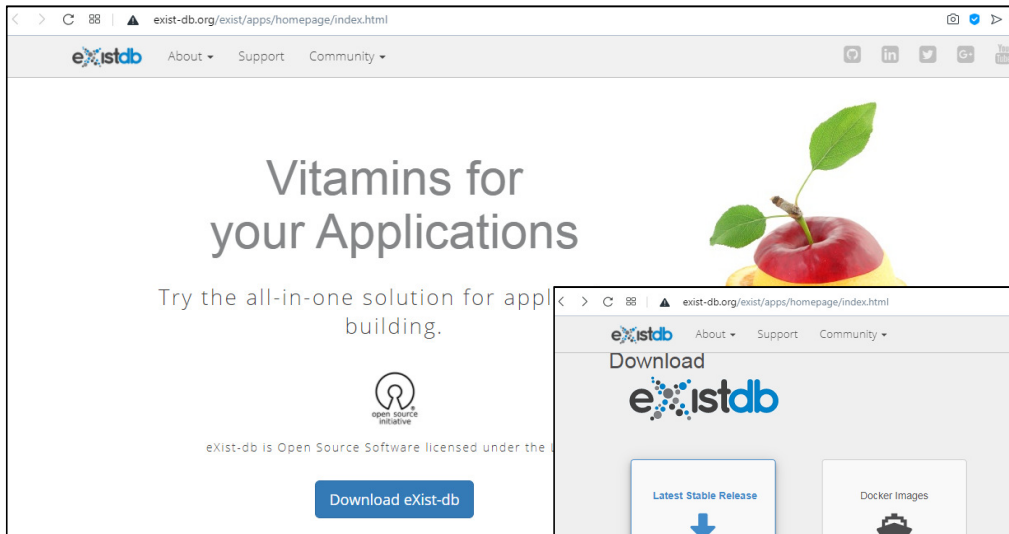
Eduard Lara

INDICE

1. INSTALACION DE EXIST-DB
2. CONSULTAS XPATH
3. CONSULTAS XQUERY
4. API XQJ
5. PRACTICA 1
6. PRACTICA 2

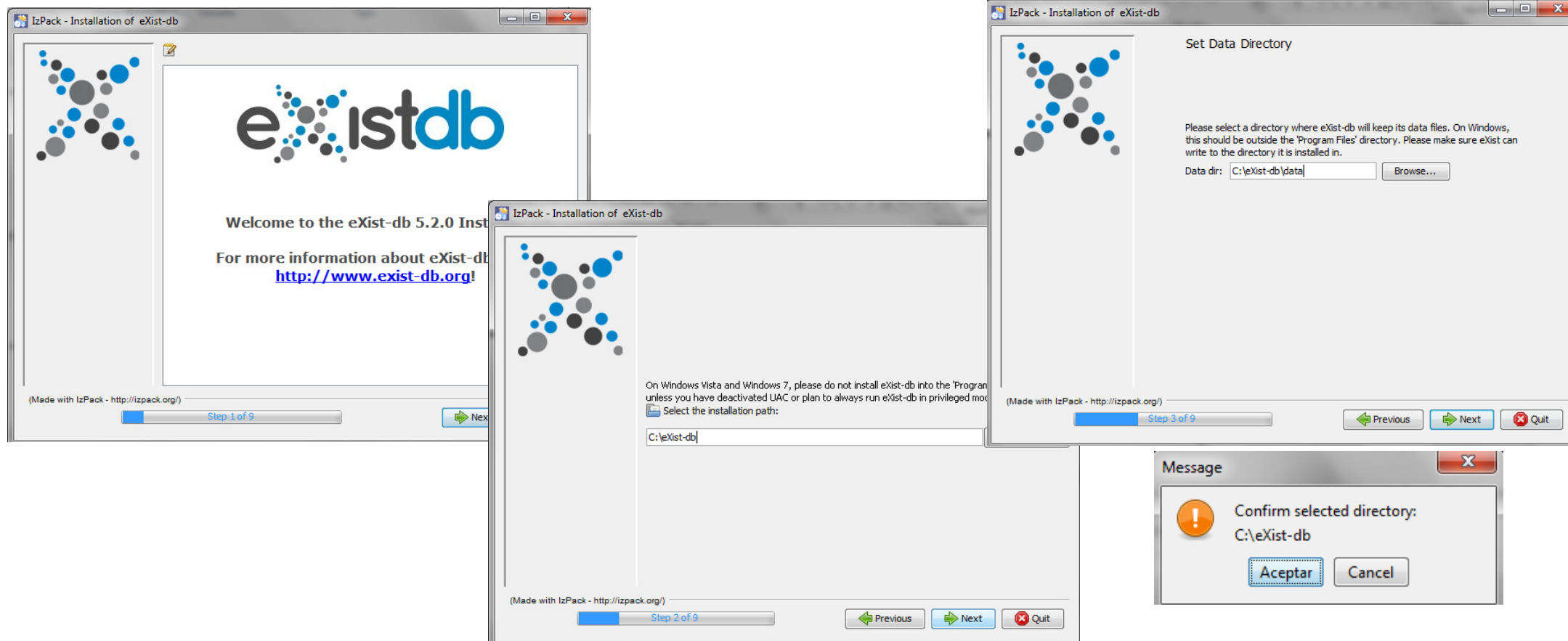
1. INSTALACION DE EXIST-DB

Paso 1. Teniendo instalado el jdk de java, debemos ir a la pagina web <http://exist-db.org/exist/apps/homepage/index.html> y descargar la ultima versión disponible de la base de datos exist-db.



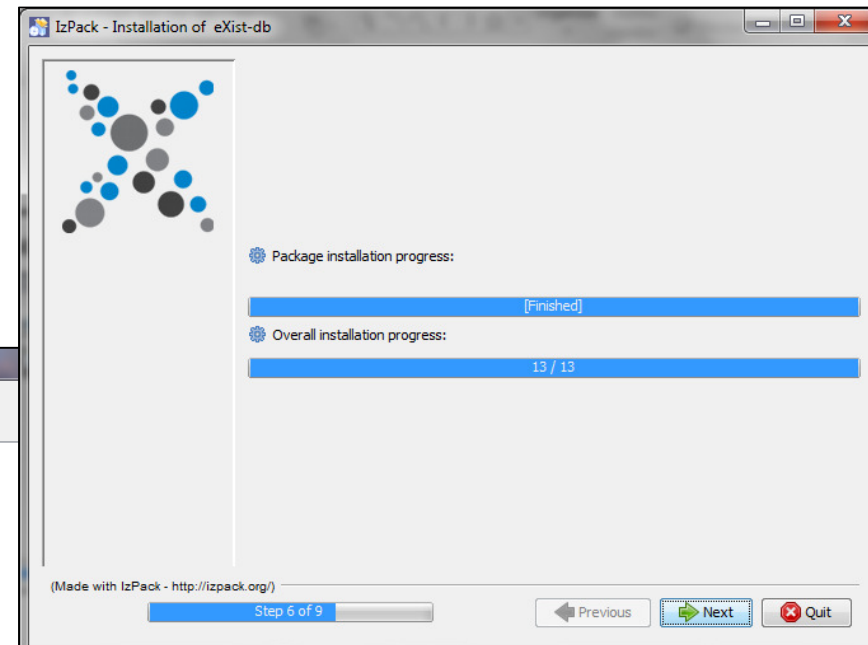
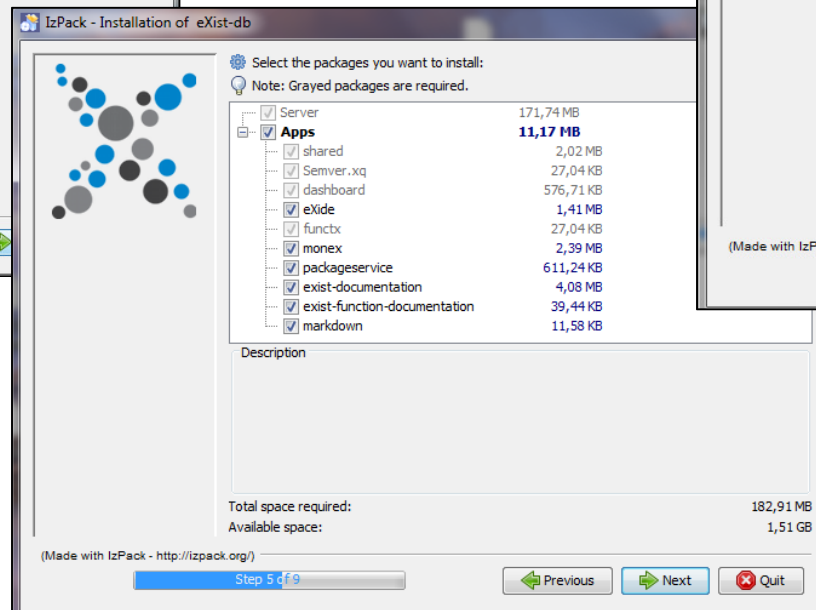
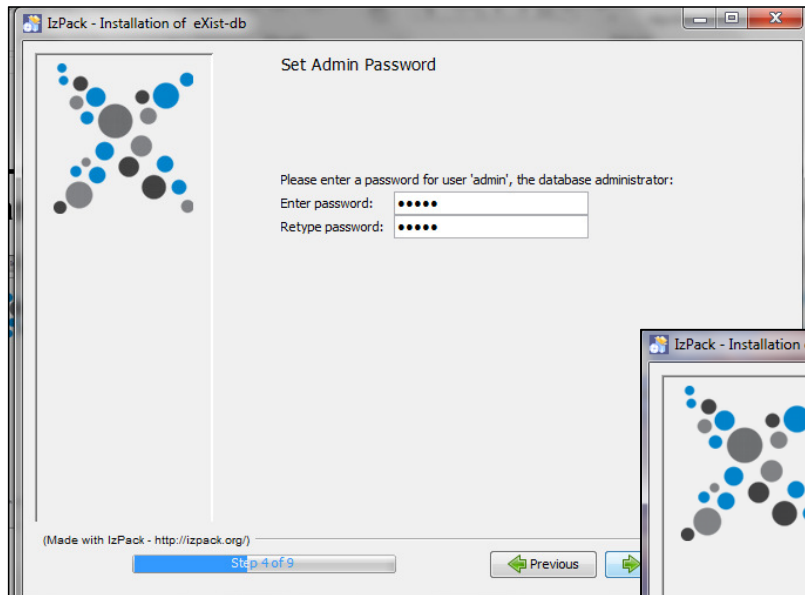
1. INSTALACION DE EXIST-DB

Paso 2. Hacemos doble click en el archivo exist-installer-5.2.0.jar y se inicia la instalación. Indicamos los directorios de instalación:



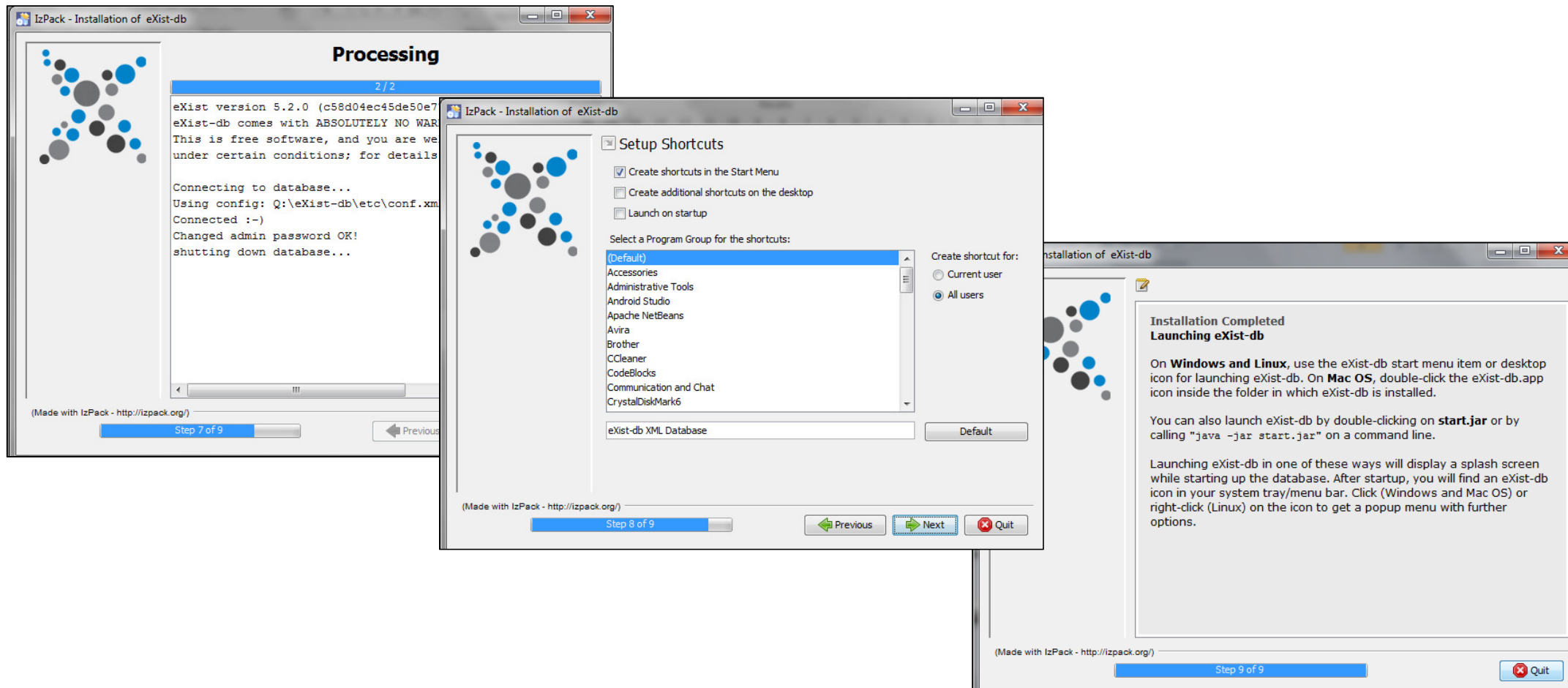
1. INSTALACION DE EXIST-DB

Paso 3. Poner el password “admin” para el usuario administrador “admin”:



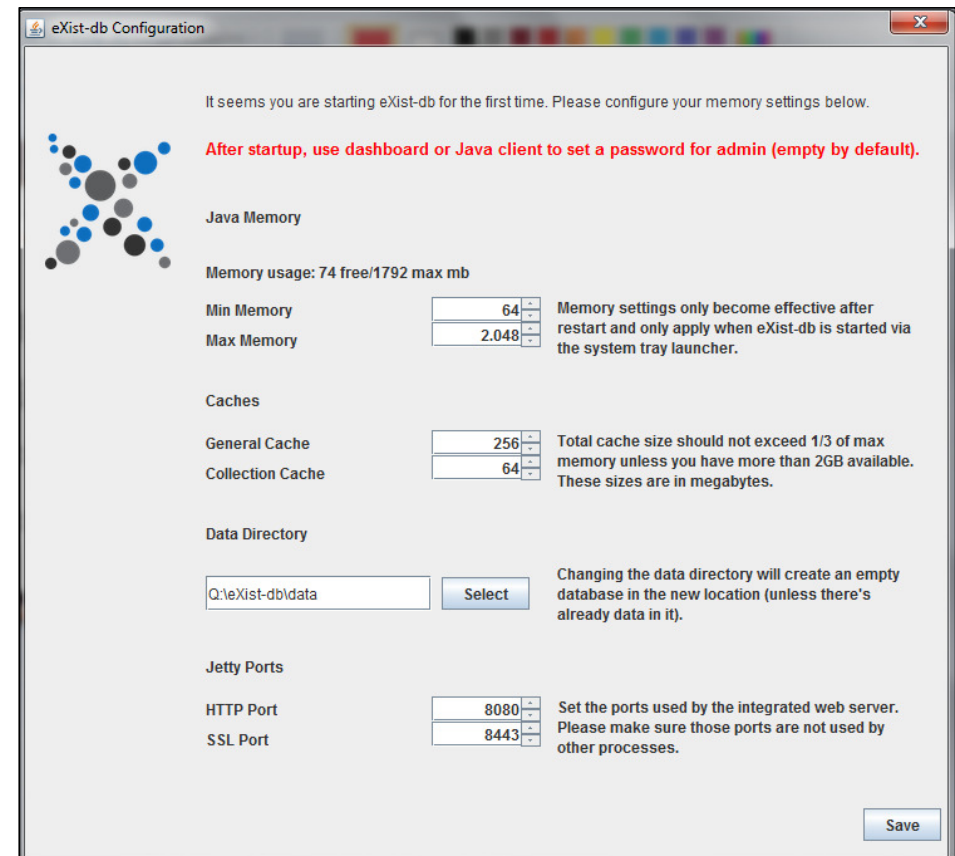
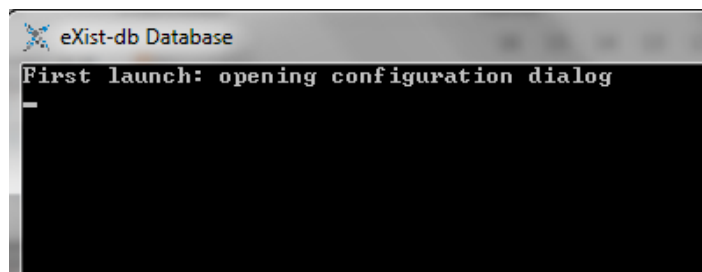
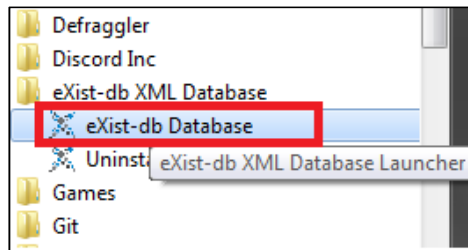
1. INSTALACION DE EXIST-DB

Paso 4. Finalizamos la instalación de exist-db:



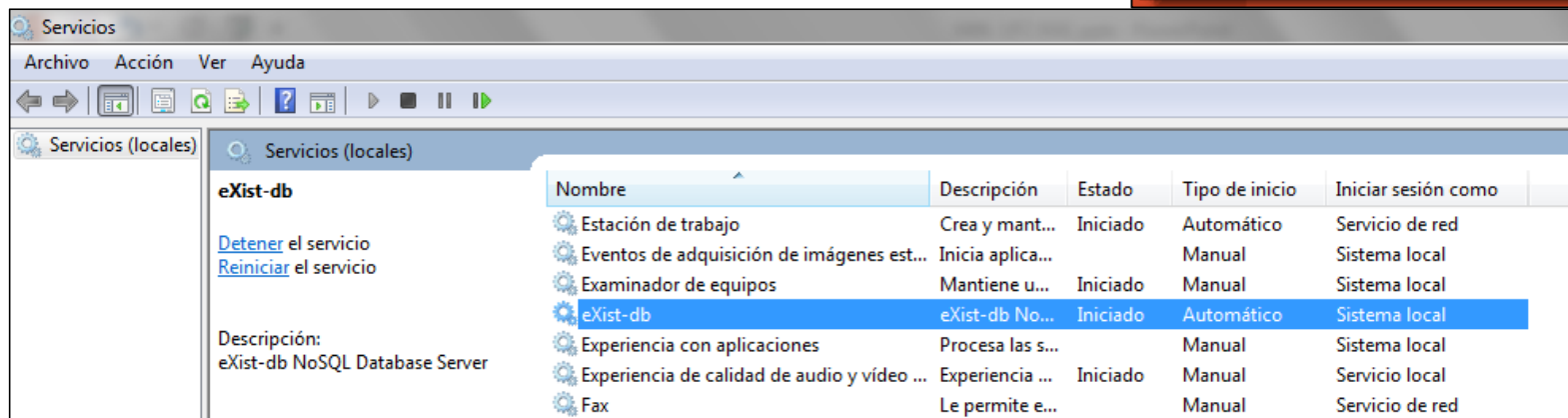
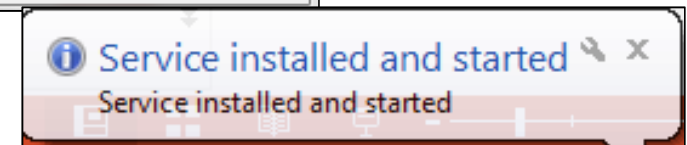
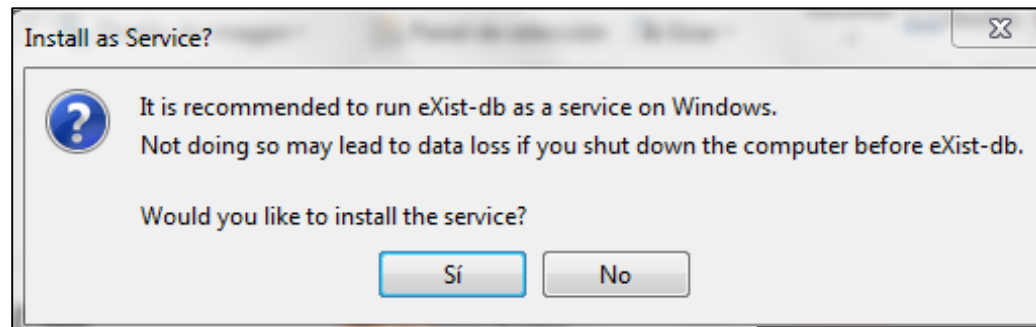
1. INSTALACION DE EXIST-DB

Paso 5. Buscamos el programa exist-db Database para iniciar el servicio por 1º vez. La base de datos exist-db por defecto utiliza el puerto 8080 igual que Tomcat. Hacemos click en Save:



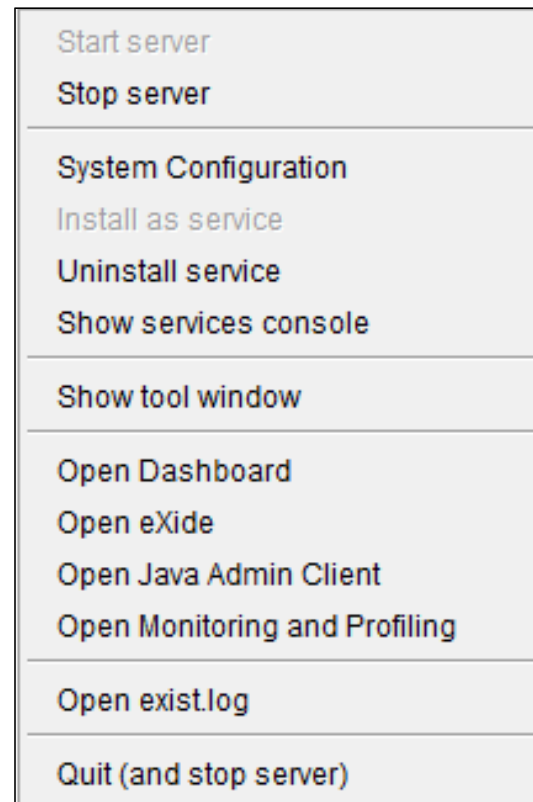
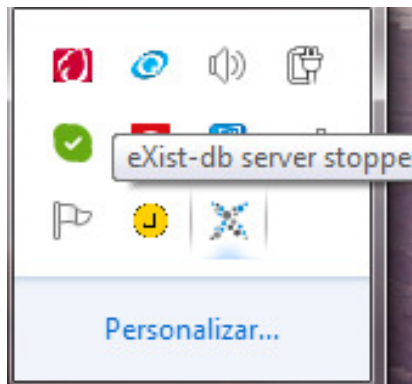
1. INSTALACION DE EXIST-DB

Paso 6. Finalmente instala los programas de gestión de exist-db y nos recomienda que ejecutemos existdb como un servicio de Windows. Podemos observar el servicio activado en services.msc:



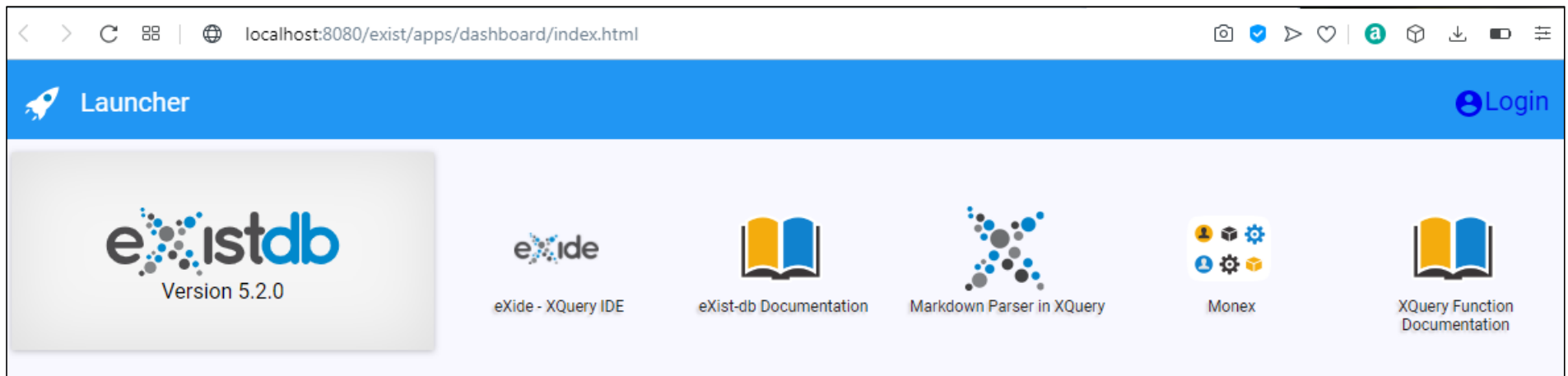
1. INSTALACION DE EXIST-DB

Paso 7. En el área de notificaciones de Windows podemos ver el icono de exist-db. Si hacemos click observamos los diferentes programas que contiene:



1. INSTALACION DE EXIST-DB

Paso 8. Una vez esta instalado y en marcha el servicio, podemos acceder al panel de control de existdb, escribiendo en un navegador localhost:8080 o a “Open Dashboard” desde el icono de notificaciones. Hacemos click en Login:



Ponemos usuario: admin
Password: admin

Login

User
admin

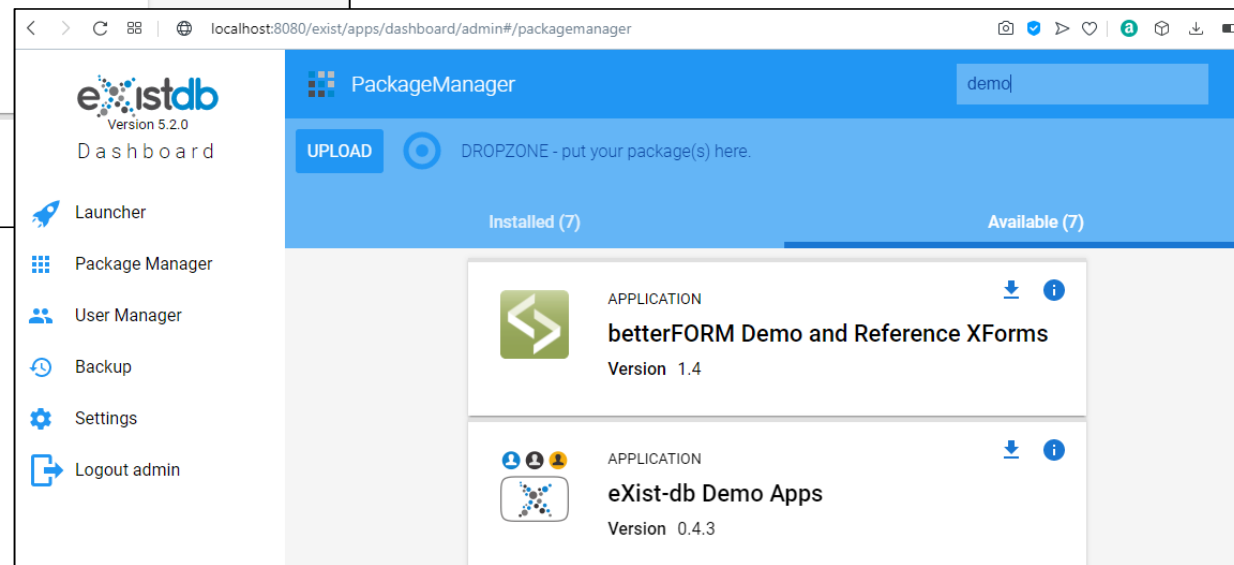
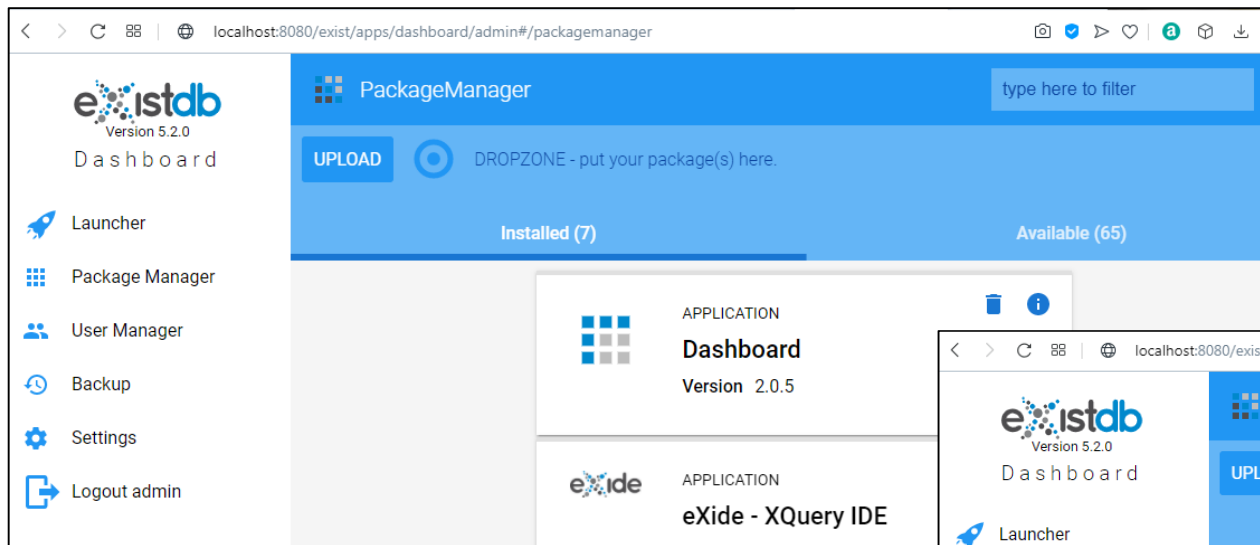
Password
.....

Wrong password or invalid user(must be member of group dba)

LOGIN

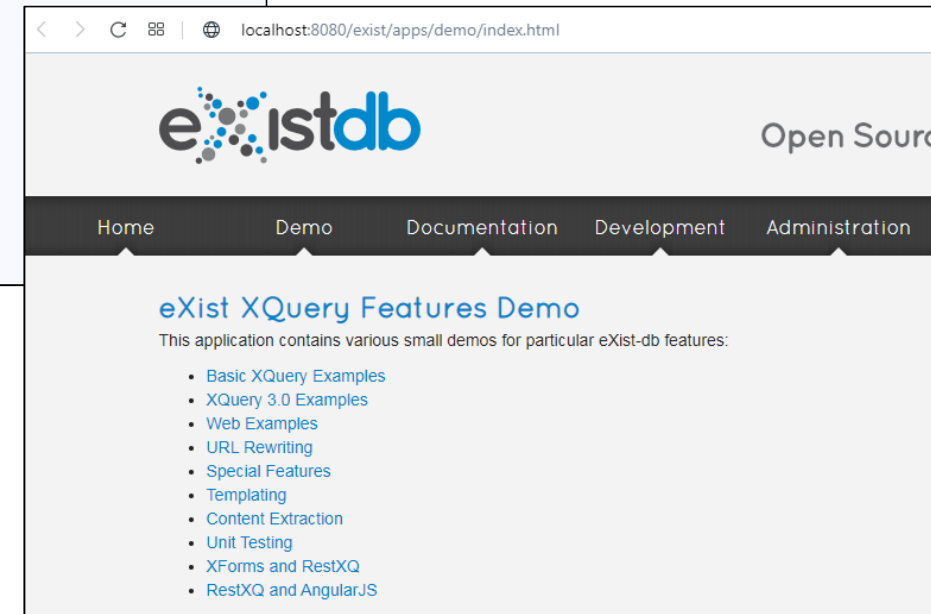
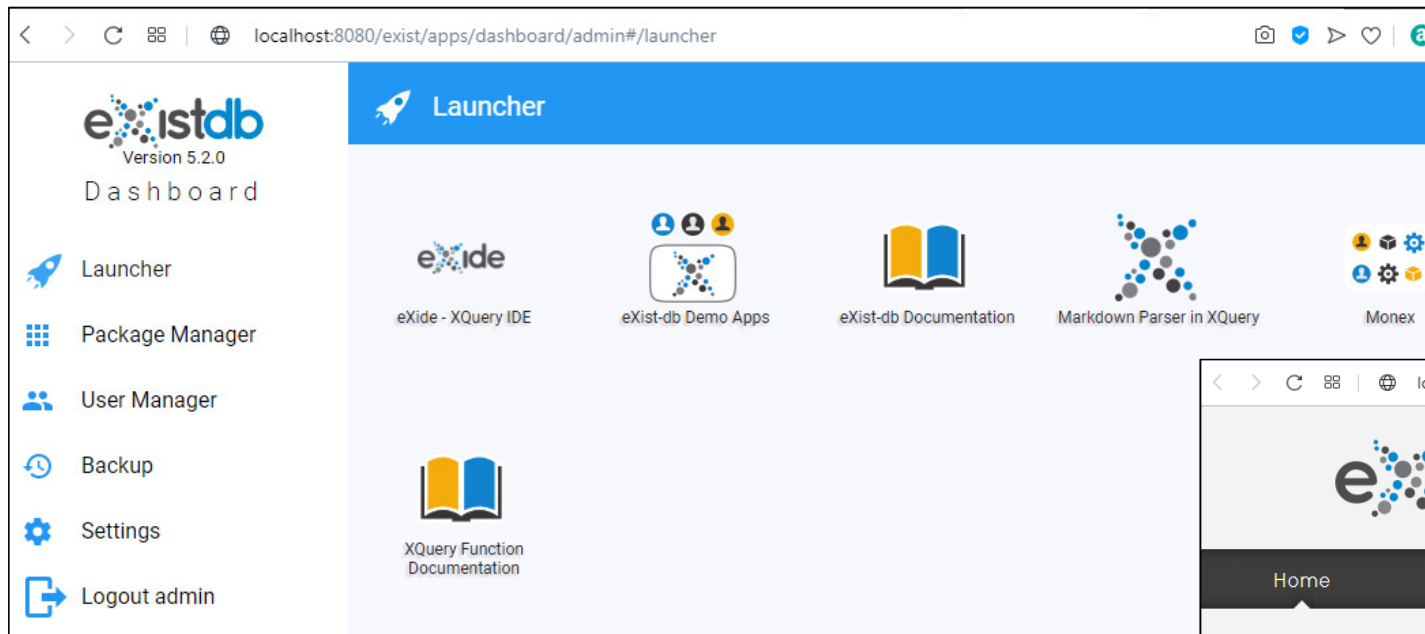
1. INSTALACION DE EXIST-DB

Paso 9. Dentro de la zona del administrador nos dirigimos al Package Manager e instalamos la utilidad Demo Apps:



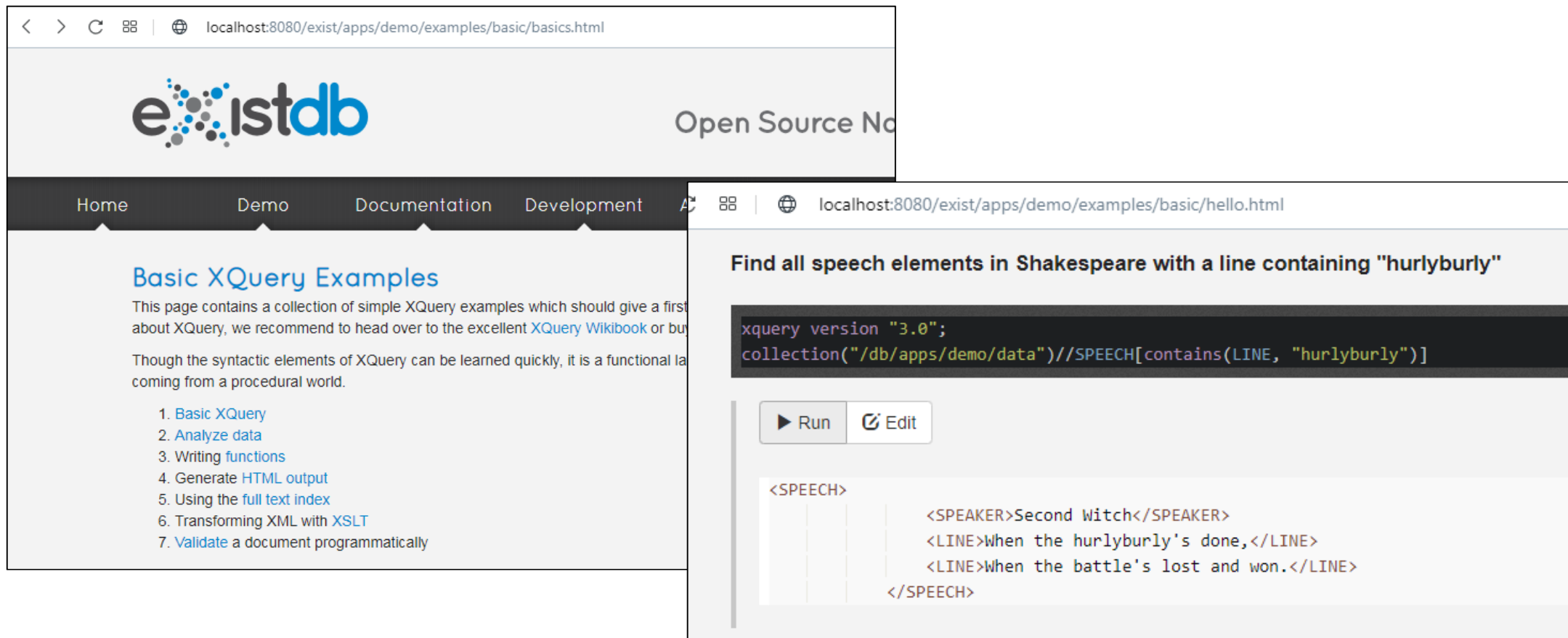
1. INSTALACION DE EXIST-DB

Paso 10. Vamos al Launcher (pagina principal) y hacemos click en Demo Apps:



1. INSTALACION DE EXIST-DB

Paso 11. Vamos a Basic XQuery Examples y a Basic XQuery. Escogemos un ejemplo y lo ejecutamos (hacemos click en el botón run):



The screenshot displays the Exist-DB web interface. The top navigation bar includes links for Home, Demo, Documentation, and Development. The main content area is titled "Basic XQuery Examples" and provides an introduction to XQuery, recommending the XQuery Wikibook and explaining that XQuery is a functional language derived from the procedural world. A list of seven examples is provided, with the first example, "Basic XQuery", selected.

Overlaid on the right side of the interface is a query execution window titled "Find all speech elements in Shakespeare with a line containing 'hurlyburly'". This window contains an XQuery query:

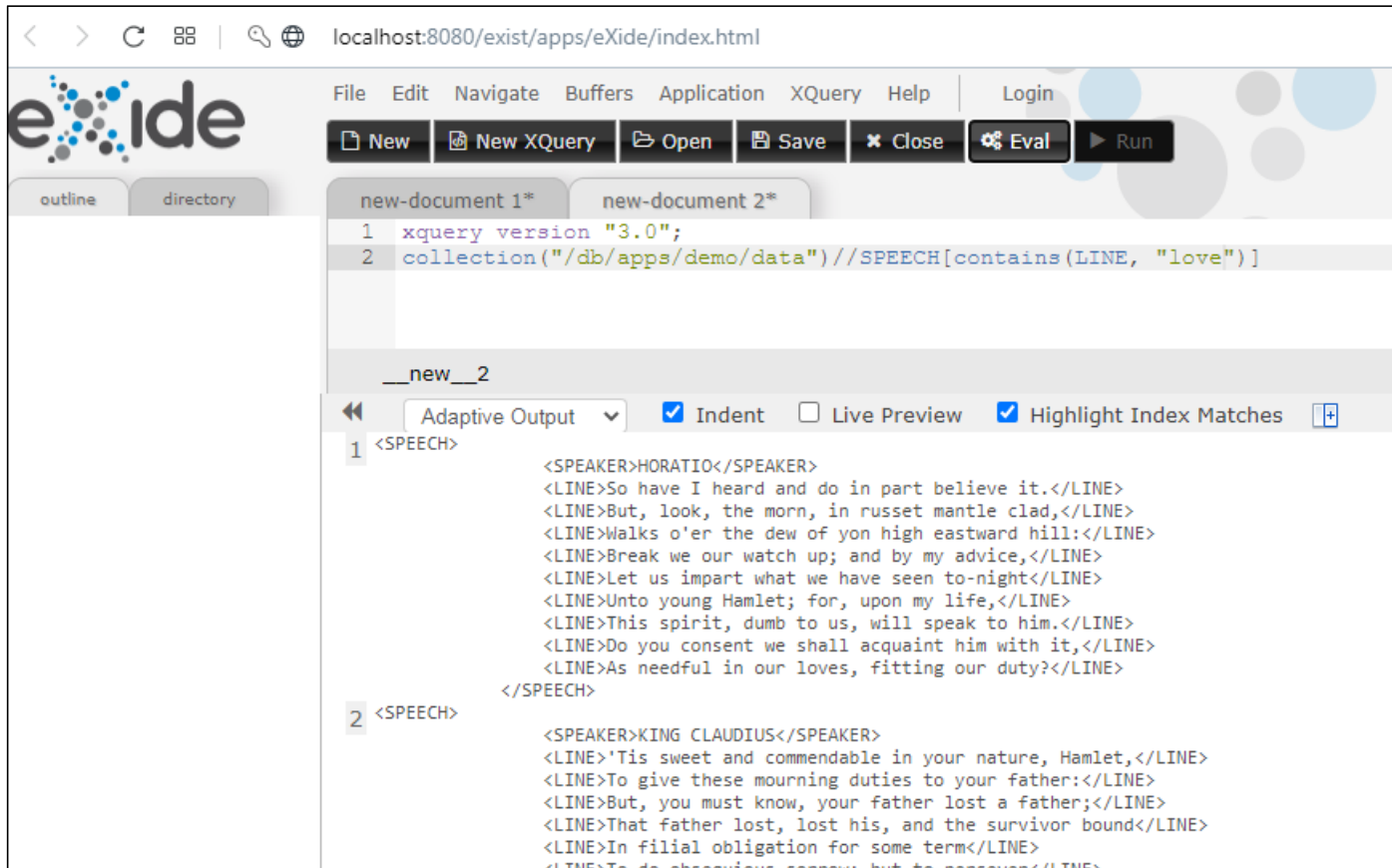
```
xquery version "3.0";
collection("/db/apps/demo/data")//SPEECH[contains(LINE, "hurlyburly")]
```

Below the query, there are "Run" and "Edit" buttons. The results of the query are displayed in a table with three columns. The first column is labeled "<SPEECH>" and the second column is labeled "<SPEAKER>". The results show two entries:

<SPEECH>	<SPEAKER>
<LINE>When the hurlyburly's done,</LINE>	Second Witch</SPEAKER>
<LINE>When the battle's lost and won.</LINE>	

1. INSTALACION DE EXIST-DB

Paso 12. Ahora le damos al boton edit y nos lleva esta consulta a la herramienta eXide. Podemos buscar la palabra “love”, haciendo click en el botón “Eval”:



The screenshot shows the eXide web interface in a browser window at localhost:8080/exist/apps/eXide/index.html. The interface includes a menu bar (File, Edit, Navigate, Buffers, Application, XQuery, Help, Login) and a toolbar with buttons for New, New XQuery, Open, Save, Close, Eval, and Run. Two document tabs are open: 'new-document 1*' and 'new-document 2*'. The active document 'new-document 2*' contains an XQuery query:

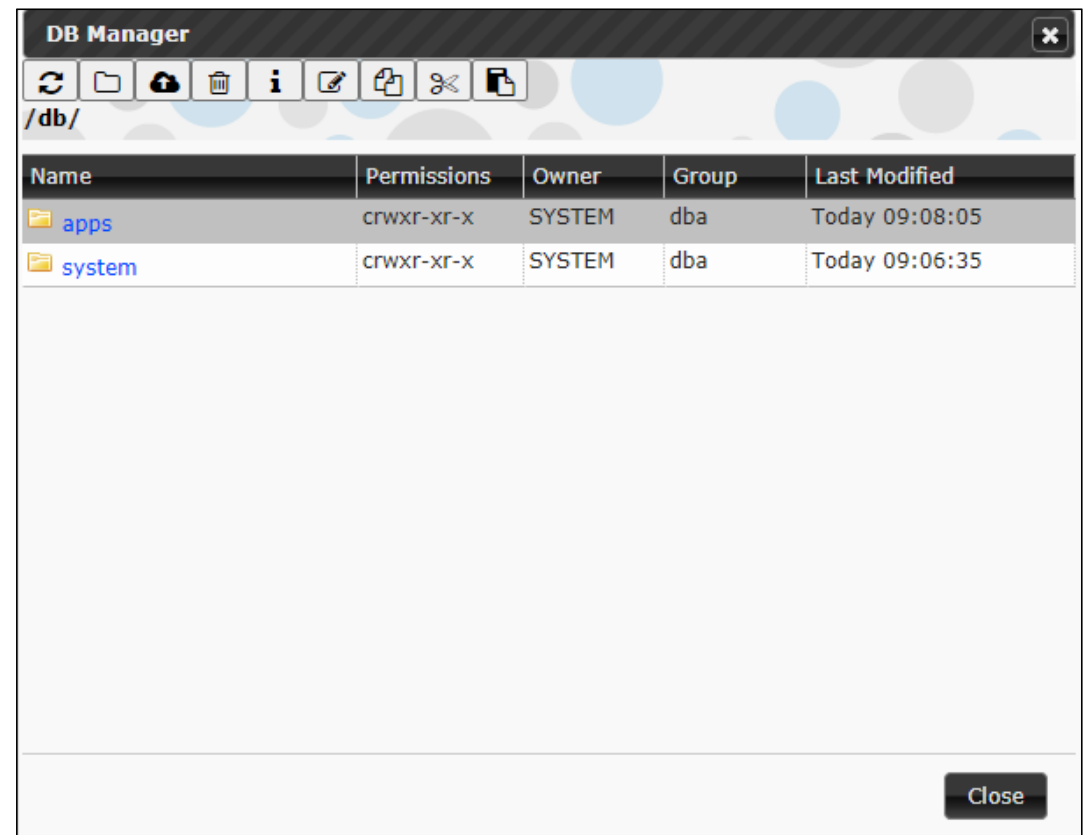
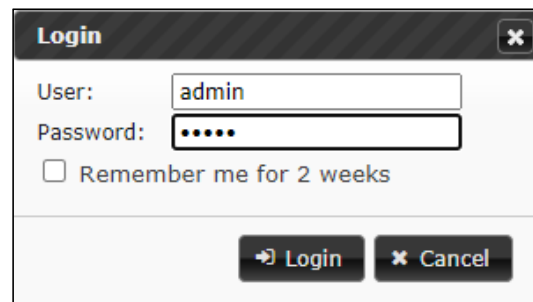
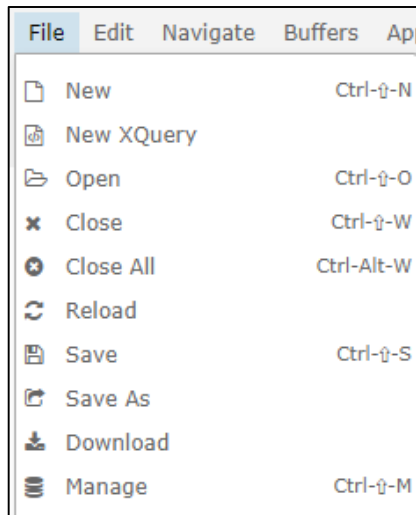
```
1 xquery version "3.0";
2 collection("/db/apps/demo/data")//SPEECH[contains(LINE, "love")]
```

Below the query editor, the 'Eval' button has been clicked, resulting in an XML output displayed in the lower pane. The output shows two speech elements, each containing a series of lines of text. The first speech element is from HORATIO, and the second is from KING CLAUDIUS.

```
__new__2
Adaptive Output  Indent  Live Preview  Highlight Index Matches
1 <SPEECH>
  <SPEAKER>HORATIO</SPEAKER>
  <LINE>So have I heard and do in part believe it.</LINE>
  <LINE>But, look, the morn, in russet mantle clad,</LINE>
  <LINE>Walks o'er the dew of yon high eastward hill:</LINE>
  <LINE>Break we our watch up; and by my advice,</LINE>
  <LINE>Let us impart what we have seen to-night</LINE>
  <LINE>Unto young Hamlet; for, upon my life,</LINE>
  <LINE>This spirit, dumb to us, will speak to him.</LINE>
  <LINE>Do you consent we shall acquaint him with it,</LINE>
  <LINE>As needful in our loves, fitting our duty?</LINE>
</SPEECH>
2 <SPEECH>
  <SPEAKER>KING CLAUDIUS</SPEAKER>
  <LINE>'Tis sweet and commendable in your nature, Hamlet,</LINE>
  <LINE>To give these mourning duties to your father:</LINE>
  <LINE>But, you must know, your father lost a father;</LINE>
  <LINE>That father lost, lost his, and the survivor bound</LINE>
  <LINE>In filial obligation for some term</LINE>
  <LINE>To do obscure mourning, but to recover</LINE>
```

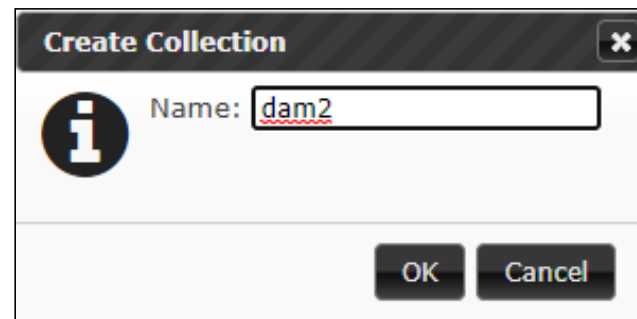
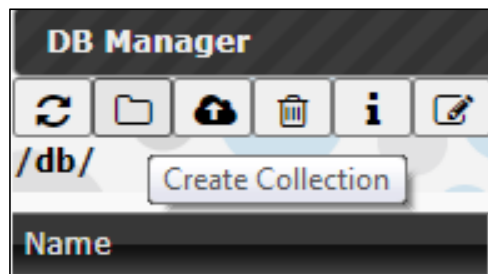
1. INSTALACION DE EXIST-DB

Paso 13. Vamos a crear una base de datos propia. Vamos a File/Manage. Nos logueamos con admin y llegamos al DB Manager:



1. INSTALACION DE EXIST-DB

Paso 14. Vamos a crear una colección de nombre dam2:

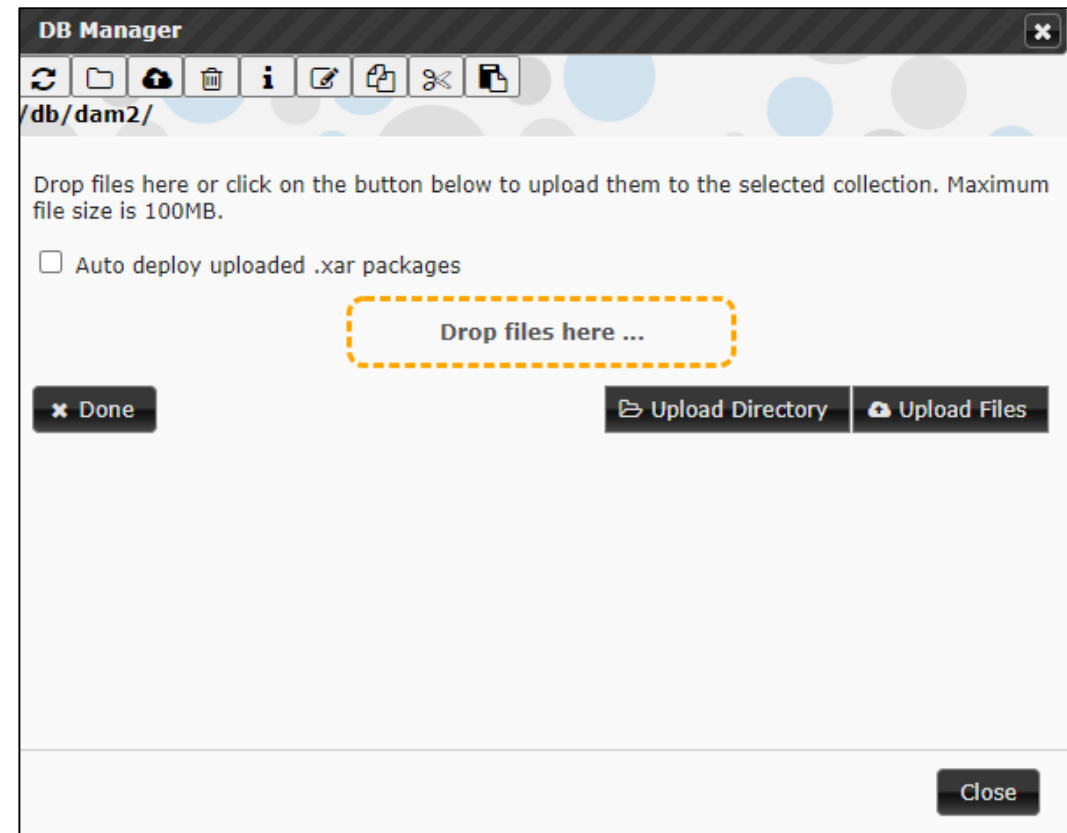
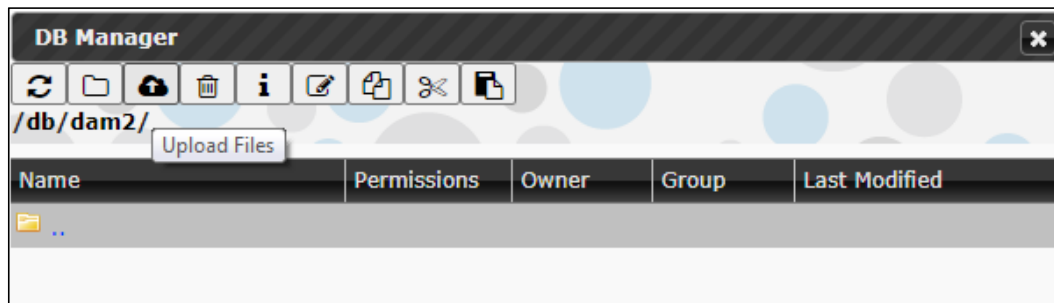


The screenshot shows the 'DB Manager' window with a list of collections. The table has columns for Name, Permissions, Owner, Group, and Last Modified. The collections listed are 'apps', 'dam2', and 'system'. The 'dam2' collection is highlighted.

Name	Permissions	Owner	Group	Last Modified
apps	crwxr-xr-x	SYSTEM	dba	Today 09:08:05
dam2	crwxr-xr-x	admin	dba	Today 15:51:02
system	crwxr-xr-x	SYSTEM	dba	Today 09:06:35

1. INSTALACION DE EXIST-DB

Paso 15. Entramos en la colección dam2 (haciendo doble click) y una dentro de /db/dam2/ hacemos click en el botón de subida de ficheros:



1. INSTALACION DE EXIST-DB

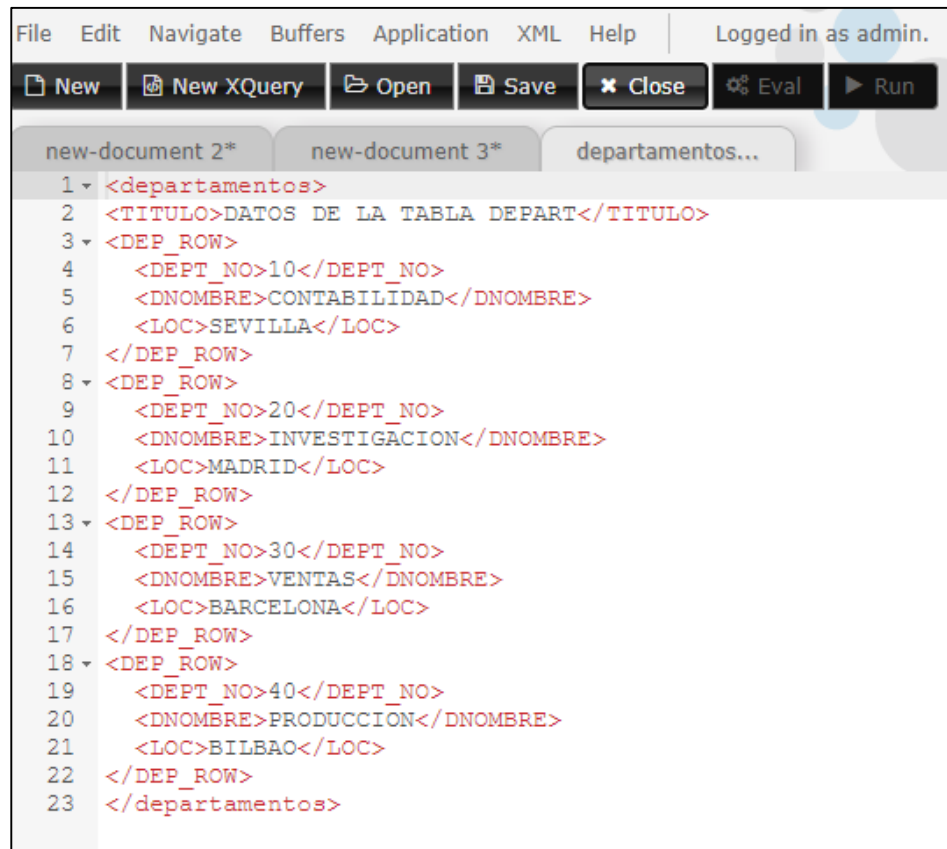
Paso 16. Subimos a la colección dam2 todos los documentos xml contenidos en el archivo “ColecciónPruebas.zip”

Nombre	Fecha de modifica...
departamentos.xml	31/01/2021 16:13
departamentosnuevo.xml	31/01/2021 16:13
empleados.xml	31/01/2021 16:13
productos.xml	31/01/2021 16:13
sucursales.xml	31/01/2021 16:13
universidad.xml	31/01/2021 16:13
zonas.xml	31/01/2021 16:13

DB Manager				
/db/dam2/				
Name	Permissions	Owner	Group	Last Modified
..				
departamentos.xml	-rw-r--r--	admin	dba	Today 16:14:07
departamentosnuevo.xml	-rw-r--r--	admin	dba	Today 16:14:07
empleados.xml	-rw-r--r--	admin	dba	Today 16:14:07
productos.xml	-rw-r--r--	admin	dba	Today 16:14:07
sucursales.xml	-rw-r--r--	admin	dba	Today 16:14:07
universidad.xml	-rw-r--r--	admin	dba	Today 16:14:07
zonas.xml	-rw-r--r--	admin	dba	Today 16:14:07
Close				

1. INSTALACION DE EXIST-DB

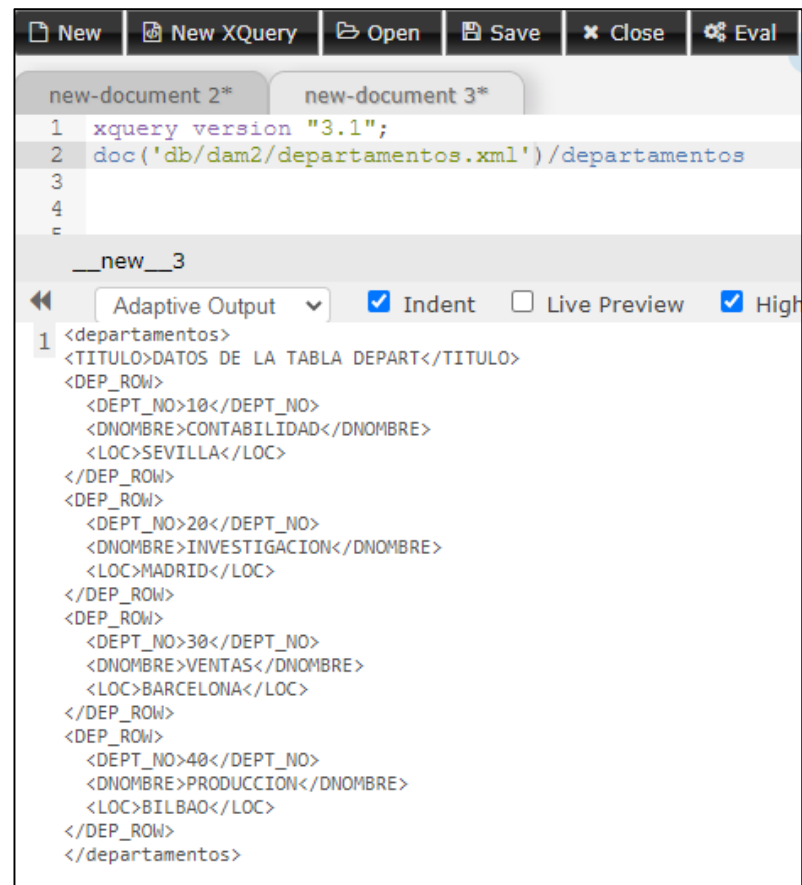
Paso 17. Se puede visualizar cualquier de los documentos subidos haciendo doble click encima de ellos.

The screenshot shows the EXIST-DB web interface. At the top, there is a menu bar with 'File', 'Edit', 'Navigate', 'Buffers', 'Application', 'XML', and 'Help'. To the right of the menu, it says 'Logged in as admin.'. Below the menu is a toolbar with buttons: 'New', 'New XQuery', 'Open', 'Save', 'Close', 'Eval', and 'Run'. There are three tabs: 'new-document 2*', 'new-document 3*', and 'departamentos...'. The 'departamentos...' tab is active, displaying an XML document. The XML content is as follows:

```
1 <departamentos>
2 <TITULO>DATOS DE LA TABLA DEPART</TITULO>
3 <DEP_ROW>
4   <DEPT_NO>10</DEPT_NO>
5   <DNOMBRE>CONTABILIDAD</DNOMBRE>
6   <LOC>SEVILLA</LOC>
7 </DEP_ROW>
8 <DEP_ROW>
9   <DEPT_NO>20</DEPT_NO>
10  <DNOMBRE>INVESTIGACION</DNOMBRE>
11  <LOC>MADRID</LOC>
12 </DEP_ROW>
13 <DEP_ROW>
14   <DEPT_NO>30</DEPT_NO>
15   <DNOMBRE>VENTAS</DNOMBRE>
16   <LOC>BARCELONA</LOC>
17 </DEP_ROW>
18 <DEP_ROW>
19   <DEPT_NO>40</DEPT_NO>
20   <DNOMBRE>PRODUCCION</DNOMBRE>
21   <LOC>BILBAO</LOC>
22 </DEP_ROW>
23 </departamentos>
```

1. INSTALACION DE EXIST-DB

Paso 18. Crea en la pestaña new XQuery una primera consulta:
`doc('db/dam2/departamentos.xml')/departamentos`



The screenshot shows the EXIST-DB XQuery editor interface. The top toolbar includes buttons for 'New', 'New XQuery', 'Open', 'Save', 'Close', and 'Eval'. Below the toolbar, there are two tabs: 'new-document 2*' and 'new-document 3*'. The 'new-document 3*' tab is active and contains the following XQuery code:

```
1 xquery version "3.1";
2 doc('db/dam2/departamentos.xml')/departamentos
3
4
5
```

Below the code editor, there is a section labeled '___new___3' which displays the XML output of the query. The output is an XML document with a root element 'departamentos' and a title 'DATOS DE LA TABLA DEPART'. It contains four rows of department data:

```
1 <departamentos>
  <TITULO>DATOS DE LA TABLA DEPART</TITULO>
  <DEP_ROW>
    <DEPT_NO>10</DEPT_NO>
    <DNOMBRE>CONTABILIDAD</DNOMBRE>
    <LOC>SEVILLA</LOC>
  </DEP_ROW>
  <DEP_ROW>
    <DEPT_NO>20</DEPT_NO>
    <DNOMBRE>INVESTIGACION</DNOMBRE>
    <LOC>MADRID</LOC>
  </DEP_ROW>
  <DEP_ROW>
    <DEPT_NO>30</DEPT_NO>
    <DNOMBRE>VENTAS</DNOMBRE>
    <LOC>BARCELONA</LOC>
  </DEP_ROW>
  <DEP_ROW>
    <DEPT_NO>40</DEPT_NO>
    <DNOMBRE>PRODUCCION</DNOMBRE>
    <LOC>BILBAO</LOC>
  </DEP_ROW>
</departamentos>
```

2. XPATH Y XQUERY

- Tanto XPath com XQuery son estándares para acceder y obtener datos desde documentos XML.
- Estos lenguajes tienen en cuenta que la información en los documentos está semiestructurada o jerarquizada como árbol.
- **XPath** → Lenguaje de rutas XML, se utiliza para navegar dentro de la estructura jerárquica de un XML
- **XQuery** → es a XML lo mismo que SQL es a las bbdd relacionales, es decir, un lenguaje de consulta diseñado para consultar documentos XML. XQuery contiene a XPath, toda expresión de consulta en XPath es válida en XQuery, pero XQuery permite mucho más

2. CONSULTAS XPATH

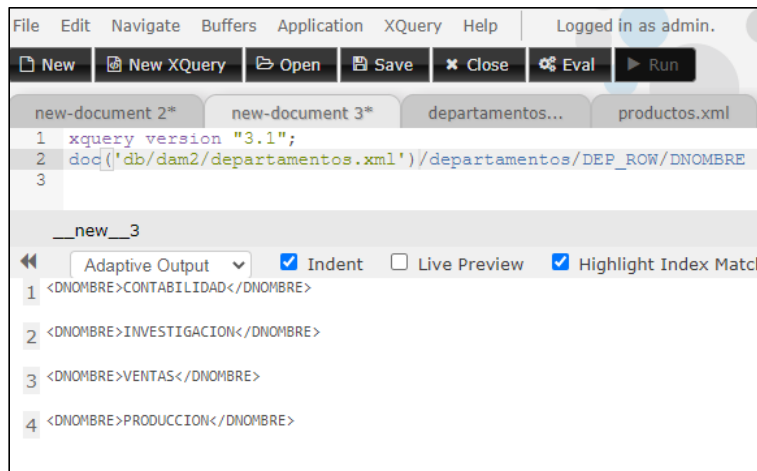
Expresiones Xpath

- XPath es un lenguaje que permite seleccionar nodos de un documento XML y calcular valores a partir de su contenido. Existen varias versiones de XPath aprobadas por W3C aunque la versión más utilizada sigue siendo la 1.
- La forma en que XPath selecciona partes del documento XML se basa en la representación arbórea que se genera del documento.
- A la hora de recorrer un árbol XML podemos encontrarnos con los siguientes tipos de nodos:
 - **nodo raíz** → raíz del árbol, se representa por /
 - **nodos elemento** → cualquier elemento del árbol, son las etiquetas del árbol
 - **nodos texto** → los caracteres entre etiquetas
 - **nodos atributo** → propiedades añadidas a los nodos elementos, se representan con @
 - **nodos comentario** → etiquetas de comentario
 - **nodos espacio de nombres** → contienen espacio de nombres
 - **nodos instrucción de proceso** → instrucciones de proceso, van entre <?.....?>

2. CONSULTAS XPATH

Ejercicio 1. Comprueba el resultado de las siguientes consultas:

- /departamentos → devuelve todos los datos de departamentos (Esta sería la misma consulta del ejercicio anterior pero sin indicar toda la ruta)
- /departamentos/DEP_ROW → devuelve las etiquetas de cada DEP_ROW
- /departamentos/DEP_ROW/DNOMBRE → devuelve nombres de departamentos entre etiquetas

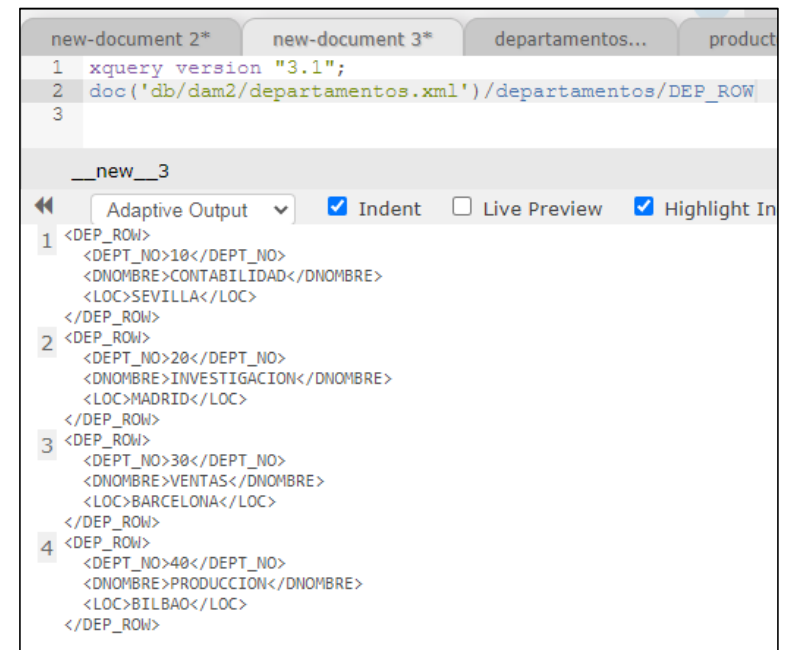


The screenshot shows the XQuery IDE interface. The top menu bar includes File, Edit, Navigate, Buffers, Application, XQuery, and Help. The status bar indicates 'Logged in as admin.'. The main editor area displays the following XQuery code:

```
1 xquery version "3.1";
2 doc('db/dam2/departamentos.xml')/departamentos/DEP_ROW/DNOMBRE
3
```

The result pane, titled '__new__3', shows the output of the query. It lists four department names, each enclosed in its XML element tags:

```
1 <DNOMBRE>CONTABILIDAD</DNOMBRE>
2 <DNOMBRE>INVESTIGACION</DNOMBRE>
3 <DNOMBRE>VENTAS</DNOMBRE>
4 <DNOMBRE>PRODUCCION</DNOMBRE>
```



The screenshot shows the XQuery IDE interface. The top menu bar includes File, Edit, Navigate, Buffers, Application, XQuery, and Help. The status bar indicates 'Logged in as admin.'. The main editor area displays the following XQuery code:

```
1 xquery version "3.1";
2 doc('db/dam2/departamentos.xml')/departamentos/DEP_ROW
3
```

The result pane, titled '__new__3', shows the output of the query. It lists four department elements, each enclosed in its XML element tags:

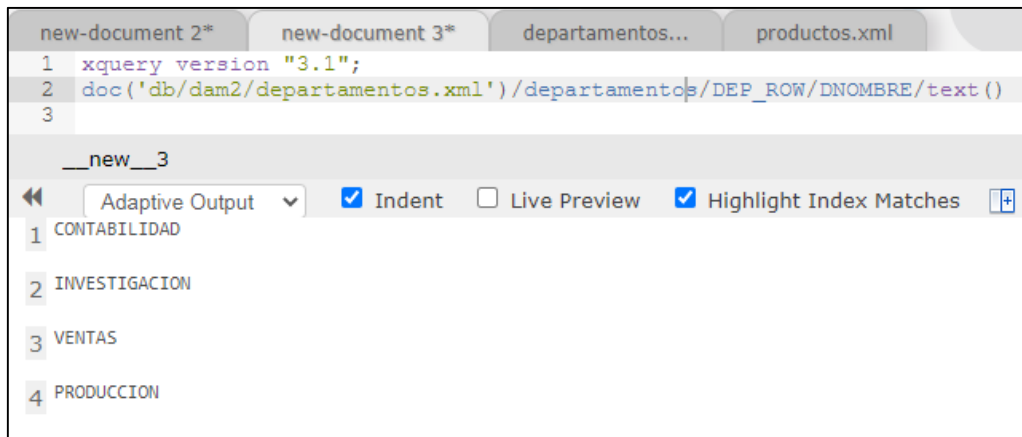
```
1 <DEP_ROW>
  <DEPT_NO>10</DEPT_NO>
  <DNOMBRE>CONTABILIDAD</DNOMBRE>
  <LOC>SEVILLA</LOC>
</DEP_ROW>
2 <DEP_ROW>
  <DEPT_NO>20</DEPT_NO>
  <DNOMBRE>INVESTIGACION</DNOMBRE>
  <LOC>MADRID</LOC>
</DEP_ROW>
3 <DEP_ROW>
  <DEPT_NO>30</DEPT_NO>
  <DNOMBRE>VENTAS</DNOMBRE>
  <LOC>BARCELONA</LOC>
</DEP_ROW>
4 <DEP_ROW>
  <DEPT_NO>40</DEPT_NO>
  <DNOMBRE>PRODUCCION</DNOMBRE>
  <LOC>BILBAO</LOC>
</DEP_ROW>
```

2. CONSULTAS XPATH

- d. `/departamentos/DEP_ROW/DNOMBRE/text()` → Lo mismo que antes pero sin etiquetas
- e. `//LOC/text()` → localidades

NOTA: `/` se usa para dar rutas absolutas.

Si el descriptor comienza con `//` se supone que la ruta descrita puede comenzar en cualquier parte

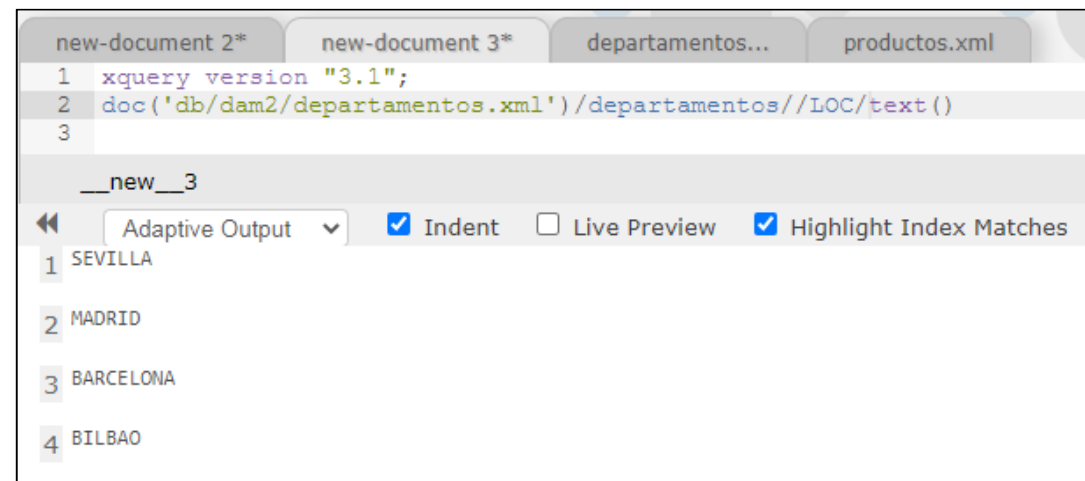


The screenshot shows an XML editor with tabs for 'new-document 2*', 'new-document 3*', 'departamentos...', and 'productos.xml'. The active tab is 'departamentos...'. The XQuery editor contains the following code:

```
1 xquery version "3.1";
2 doc('db/dam2/departamentos.xml')/departamentos/DEP_ROW/DNOMBRE/text()
3
```

The result pane, titled '__new__3', displays a list of department names:

- 1 CONTABILIDAD
- 2 INVESTIGACION
- 3 VENTAS
- 4 PRODUCCION



The screenshot shows the same XML editor with the same tabs. The XQuery editor contains the following code:

```
1 xquery version "3.1";
2 doc('db/dam2/departamentos.xml')/departamentos//LOC/text()
3
```

The result pane, titled '__new__3', displays a list of location names:

- 1 SEVILLA
- 2 MADRID
- 3 BARCELONA
- 4 BILBAO

2. CONSULTAS XPATH

Ejercicio 2. Averigua el resultado de las siguientes consultas (utilizaremos el documento 'db/dam2/empleados.xml')

- a. /EMPLEADOS/EMP_ROW[DEPT_NO=10]
- b. /EMPLEADOS/EMP_ROW/APELLIDO|/EMPLEADOS/EMP_ROW/DEPT_NO
- c. /EMPLEADOS/EMP_ROW [DEPT_NO=10]/APELLIDO/text()
- d. /EMPLEADOS/EMP_ROW [not(OFICIO='ANALISTA')]
- e. /EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=20]/APELLIDO
- f. /EMPLEADOS/EMP_ROW[1]

2. CONSULTAS XPATH

Ejercicio 3. Investiga en la web las siguientes funciones de XPath y pon algún ejemplo utilizando los documentos departamentos.xml y empleados.xml

- a. last()
- b. position()
- c. count()
- d. sum(),div(),mod()
- e. max(), min(),avg()
- f. concat(cadena1, cadena2,...)
- g. starts-with (cadena1, cadena2)
- h. contains(cad1,cad2)
- i. string-length(argumento)

2. CONSULTAS XPATH

Ejercicio 4. Resuelve las siguientes consultas:

- a. Devuelve el apellido del penúltimo empleado (NOTA: utilizar last())
- b. Obtén los elementos del empleado que ocupa la posición 3 (position())
- c. Cuenta el número de empleados del departamento 10
- d. Obtén la suma de SALARIO de los empleados del DEPT_NO =20
- e. Obtén el salario máximo, el mínimo de los empleados con OFICIO=ANALISTA
- f. Obtén la media de salario en el DEPT_NO=10
- g. Devuelve la concatenación de apellido, oficio y salario
- h. Obtén los elementos de los empleados cuyo apellido empieza por 'A'
- i. Devuelve los oficios que contienen la sílaba 'OR'
- j. Obtén los datos de los empleados cuyo apellido tiene menos de 4 caracteres

2. CONSULTAS XPATH

Ejercicio 5. Resuelve las siguientes consultas referentes al documento productos.xml. Este documento contiene los datos de los productos de una distribuidora de componentes informáticos.

- a) Obtén la denominación y precio de todos los productos
- b) Obtén los productos que sean “Placa base”
- c) Obtén los productos cuyo precio sea mayor que 60€ y de la zona 20
- d) Obtén el número de los productos que sean memorias y de la zona 10
- e) Obtén la media de los precios de los micros
- f) Obtén los datos de los productos cuyo stock mínimo sea mayor que el stock actual (usa función number())
- g) Obtén el producto más caro
- h) Obtén el producto más barato de la zona 20

3. CONSULTAS XQUERY

- Una consulta XQuery es una expresión que lee datos de uno o más documentos en XML y devuelve como resultado otra secuencia de datos en XML.
- XQuery contiene a XPath. Es decir, toda expresión de consulta en XPath es válida y devuelve el mismo resultado en XQuery.
- XQuery nos va a permitir:
 - Seleccionar información basada en un criterio específico
 - Buscar información en un documento o conjunto de documentos
 - Unir datos desde múltiples documentos
 - Transformar y reestructurar datos XML en otro vocabulario o estructura
 - ...

3. CONSULTAS XQUERY

- En las consultas XQuery podemos utilizar las siguientes funciones para referirnos a colecciones y documentos dentro de la bbdd:
 - **collection("/ruta")** → indicamos el camino para referirnos a una colección
 - **doc("/ruta/documento.xml")** → indicamos el camino de un documento de una colección
- En XQuery las consultas se pueden construir utilizando expresiones FLWOR que corresponde a las siglas de **For, Let, Where, Order y Return**.
- La sintaxis general es:

```
for <variable> in <expresión XPath>  
let <variables vinculadas>  
where <condición XPath>  
order by <expresión>  
return <expresión de salida>
```

3. CONSULTAS XQUERY

- **For** → se usa para seleccionar nodos y almacenarlos en una variable, similar a la cláusula FROM de SQL. Dentro del for escribimos una expresión XPath que seleccionará a los nodos. Si se especifica más de una variable en el for se actúa como producto cartesiano. Las variables comienzan con \$
- Las consultas XQuery deben llevar obligatoriamente una orden **Return**, donde indicaremos lo que queremos que nos devuelva la consulta.
- Ejemplo comparativo XQuery - XPath

XQuery	XPath
for \$emp in /EMPLEADOS/EMP_ROW return \$emp	/EMPLEADOS/EMP_ROW
for \$emp in /EMPLEADOS/EMP_ROW return \$emp/APELLIDO	/EMPLEADOS/EMP_ROW/APELLIDO

3. CONSULTAS XQUERY

- **Let** → permite que se asignen valores resultantes de expresiones XPath a variables para simplificar la representación. Se pueden poner varias líneas let una por cada variable, o separar las variables por comas.
- En el siguiente ejemplo, se crean dos variables \$nom y \$ofi. La salida sale ordenada por OFICIO, y se crea una etiqueta <APE_OFI> que incluye el nombre y oficio concatenado.

```
for $emp in /EMPLEADOS/EMP_ROW
let $nom:=$emp/APELLIDO, $ofi:=$emp/OFICIO
order by $emp/OFICIO
return <APE_OFI> {concat($nom, ' ', $ofi)} </APE_OFI>
```


3. CONSULTAS XQUERY

- **Where** → para filtrar elementos
- **Order by** → ordena los datos según un criterio dado
- **Return** → construye el resultado de la consulta en XML. Se pueden añadir etiquetas XML a la salida. Si lo hacemos, los datos a visualizar los encerramos entre llaves {}. Además, en el return se puede añadir condicionales usando **if-then-else**
- El siguiente ejemplo devuelve los departamentos de tipo A encerrados en una etiqueta

```
for $dep in /universidad/departamento
return if ( $dep/@tipo='A')
      then <tipoA> {data ( $dep/nombre)} </tipoA>
```
- Utilizaremos la función **data()** para extraer el contenido en texto de los elementos.

3. CONSULTAS XQUERY

Operadores en Xquery

- **Matemáticos:** +, - , * , div, idiv (división entera), mod
- **Comparación:** <, > , =, !=, <=, >=, not()
- **Redondeo:** floor(), ceiling(), round()
- **Funciones de agrupación:** count(), min(), max(), avg(), sum()
- **Funciones de cadena:** concat(), string-length(), starts-with(), ends-with(), substring(), upper-case(), lower-case(), string()
- **Uso general:**
 - distinct-values() → extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados
 - empty() → devuelve cierto cuando la expresión entre paréntesis está vacía.
 - exists() → devuelve cierto cuando una secuencia contiene, al menos, un elemento
- **Los comentarios en XQuery van encerrados entre caras sonrientes (: blabla :)**

3. CONSULTAS XQUERY

Ejercicio 1. Resuelve las siguientes consultas utilizando el documento EMPLEADOS.xml

- a. Obtén los nombres de oficio que empiezan por P
- b. Obtén los nombres de oficio y el número de los empleados de cada oficio.
Utiliza `distinct-values`
- c. Obtén el número de empleados que tiene cada departamento y la media de salario redondeada

3. CONSULTAS XQUERY

Ejercicio 2. Utilizando el documento productos.xml, resuelve con XQuery:

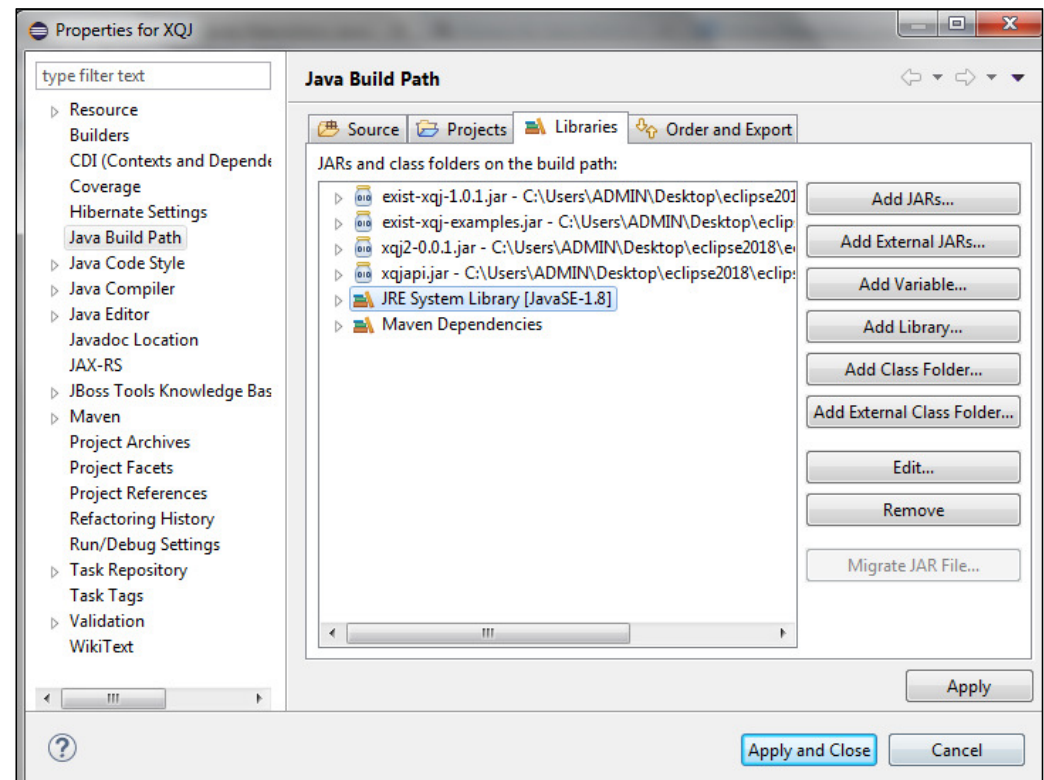
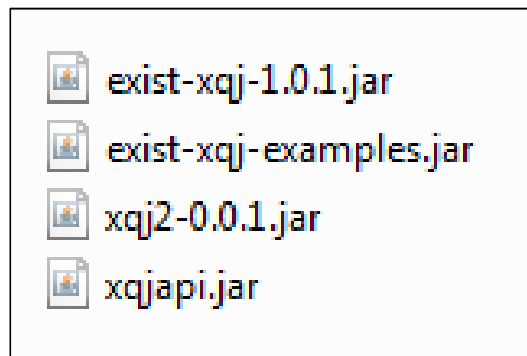
- a. Obtén por cada zona el número de productos que tiene
- b. Obtén la denominación de los productos entre las etiquetas `<zona10></zona10>` si son del código de zona 10, `<zona20></zona20>` si son del código de zona 20, etc.
- c. Obtén por cada zona la denominación del o de los productos más caros.
- d. Obtén la denominación de los productos contenida entre las etiquetas `<placa></placa>` para los productos en cuya denominación aparece la palabra Placa Base, `<memoria></memoria>`, para los que contienen la palabra Memoria `<micro></micro>`, para los que contienen la palabra Micro y `<otros></otros>` para el resto de productos

4. ACCESO A EXIST DESDE JAVA

- Ya hemos visto acceso a ficheros XML (DOM y SAX) , ahora veremos una API para acceder a la bbdd eXist
- Las APIs más conocidas son:
 - API XML:DB → cuyo objetivo es la definición de un método común de acceso a SGBD XML. Permite consulta, creación y modificación de contenido. La última actualización fue en el 2001 y aunque se sigue utilizando bastante se puede considerar obsoleta
 - API XQJ → propuesta de estandarización de interfaz Java para el acceso a bbdd XML nativas basado en modelo de datos XQuery. Es similar a JDBC. Es independiente del fabricante, fácil de usar pero solo permite realizar consultas.

4. API XQJ

- Para descargar la API accederemos a <http://xqj.net/exist/>
- Necesitamos agregar a nuestro proyecto las siguientes librerías:



4. API XQJ

Configurar una conexión

- **XQDataSource:** identifica la fuente física de datos a partir de la cual crear conexiones. Cada implementación definirá las propiedades necesarias para efectuar la conexión.

```
XQDataSource server = new ExistXQDataSource();  
server.setProperty ("serverName","localhost");  
server.setProperty ("port","8080");  
server.setProperty ("user","admin");  
server.setProperty ("password", "admin");
```

- **XQConnection:** representa una sesión con la bbdd, manteniendo información de estado, transacciones, expresiones ejecutadas y resultados.

```
XQConnection conn = server.getConnection();
```

4. API XQJ

Clases y métodos para procesar consultas

- **XQExpression:** objeto creado a partir de una conexión para la ejecución de una expresión. Devuelve un **XQResultSetSequence**. La ejecución se produce llamando al método **executeQuery**.
- **XQPreparedExpression:** objeto creado a partir de una conexión para la ejecución de una expresión múltiples veces.
- **XQDynamicContext:** representa el contexto dinámico de una expresión (zona horaria, variables a usar en la expresión)
- **XQStaticContext:** representa el contexto estático para la ejecución de expresiones
- **XQResultItem:** representación de un elemento de un resultado. Con XQJ no se necesita seleccionar la colección de los documentos XML, la búsqueda la realiza en todas las colecciones. Por tanto, a la hora de hacer consultas indicaremos la colección o el documento de la colección

5. PRACTICA 1

Realiza un programa que:

1. Devuelva los productos del catalogo productos.xml
2. Devuelva el número de productos con precio mayor a 50.
3. Devuelva todos los empleados del departamento 10.

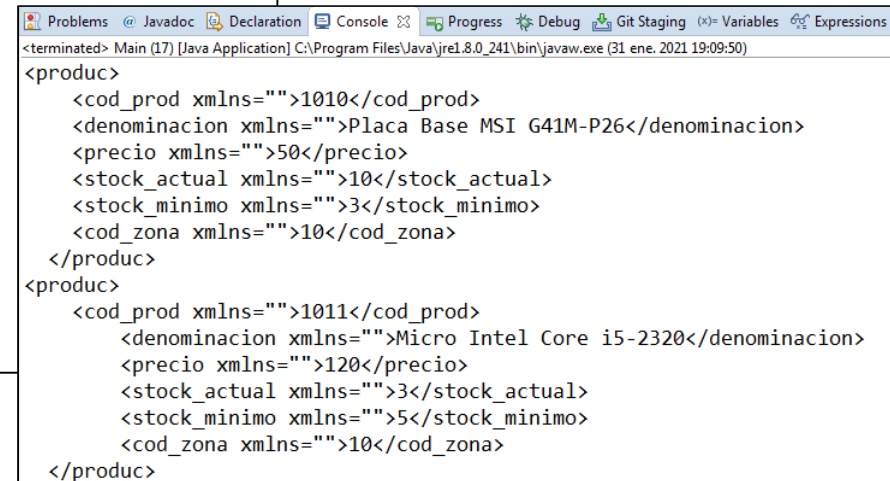
5. PRACTICA 1

Ejemplo 1 - Consulta Xquery

```
public static void main(String[] args) {
    try{
        XQDataSource server = new ExistXQDataSource();
        server.setProperty ("serverName","localhost");
        server.setProperty ("port","8080");
        server.setProperty ("user","admin");
        server.setProperty ("password", "admin");

        XQConnection conn = server.getConnection();

        String s = "for $pr in doc('dam2/productos.xml')/productos/produc return $pr";
        XQPreparedExpression consulta = conn.prepareExpression (s);
        XQResultSequence resultado = consulta.executeQuery();
        while (resultado.next()) {
            System.out.println(resultado.getItemAsString(null));
        }
        conn.close();
    } catch (XQException ex) {
        System.out.println("Error al operar"+ex.getMessage());
    }
}
```



The screenshot shows an IDE window with a console output. The console title bar includes tabs for Problems, Javadoc, Declaration, Console, Progress, Debug, Git Staging, Variables, and Expressions. The console text shows the execution of an XQuery, returning two XML product entries. The first entry is for a 'Placa Base MSI G41M-P26' with a price of 50, stock of 10, and a minimum stock of 3. The second entry is for a 'Micro Intel Core i5-2320' with a price of 120, stock of 3, and a minimum stock of 5. Both products are in zone 10.

```
<terminated> Main (17) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (31 ene. 2021 19:09:50)
<produc>
  <cod_prod xmlns="">1010</cod_prod>
  <denominacion xmlns="">Placa Base MSI G41M-P26</denominacion>
  <precio xmlns="">50</precio>
  <stock_actual xmlns="">10</stock_actual>
  <stock_minimo xmlns="">3</stock_minimo>
  <cod_zona xmlns="">10</cod_zona>
</produc>
<produc>
  <cod_prod xmlns="">1011</cod_prod>
  <denominacion xmlns="">Micro Intel Core i5-2320</denominacion>
  <precio xmlns="">120</precio>
  <stock_actual xmlns="">3</stock_actual>
  <stock_minimo xmlns="">5</stock_minimo>
  <cod_zona xmlns="">10</cod_zona>
</produc>
```

5. PRACTICA 1

Ejemplo 2 - Introducción de parámetros en consultas Xquery

```
public static void main(String[] args) {
    try{
        XQDataSource server = new ExistXQDataSource();
        server.setProperty ("serverName","localhost");
        server.setProperty ("port","8080");
        server.setProperty ("user","admin");
        server.setProperty ("password", "admin");
        XQConnection conn = server.getConnection();

        //Definimos la variable $x que representa el ID de un departamento
        String s= "declare variable $x as xs:int external;" + " /EMPLEADOS/EMP_ROW[DEPT_NO=$x]";
        XQPreparedExpression consulta = conn.prepareExpression(s);
        // Introducimos el valor del parámetro de la consulta
        int valor=10;
        consulta.bindInt(new QName("x"), valor, null);
        /* Más opciones en aquí */
        XQResultSequence resultado = consulta.executeQuery();
        while (resultado.next()) {
            System.out.println("Element E2: " + resultado.getItemAsString(null));
        }
        conn.close();
    } catch (XQException ex) {
        System.out.println("Error al operar"+ex.getMessage());
    }
}
```

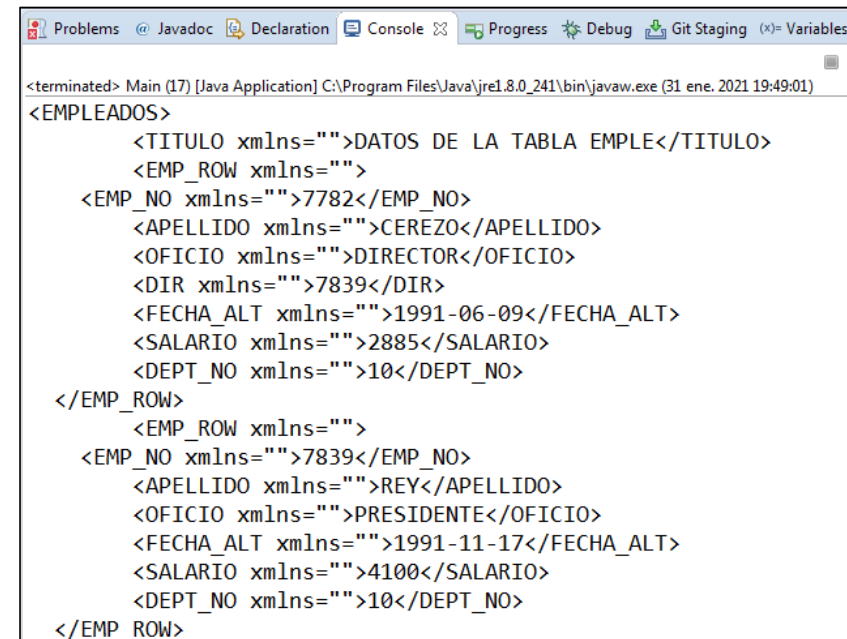
```
Problems @ Javadoc Declaration Console Progress Debug Git Staging
<terminated> Main (17) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (31 ene. 2021)
Element E2: <EMP_ROW>
    <EMP_NO xmlns="">7782</EMP_NO>
    <APELLIDO xmlns="">CEREZO</APELLIDO>
    <OFICIO xmlns="">DIRECTOR</OFICIO>
    <DIR xmlns="">7839</DIR>
    <FECHA_ALT xmlns="">1991-06-09</FECHA_ALT>
    <SALARIO xmlns="">2885</SALARIO>
    <DEPT_NO xmlns="">10</DEPT_NO>
</EMP_ROW>
Element E2: <EMP_ROW>
    <EMP_NO xmlns="">7839</EMP_NO>
    <APELLIDO xmlns="">REY</APELLIDO>
    <OFICIO xmlns="">PRESIDENTE</OFICIO>
    <FECHA_ALT xmlns="">1991-11-17</FECHA_ALT>
    <SALARIO xmlns="">4100</SALARIO>
    <DEPT_NO xmlns="">10</DEPT_NO>
</EMP_ROW>
```

5. PRACTICA 1

Ejemplo 3 - Crear un nuevo archivo XML con datos del eXist

```
//Devuelve todos los datos de los empleados del departamento 10 pero organizados
//según la estructura de los datos en el documento empleados.xml
String s= "let $titulo:= /EMPLEADOS/TITULO return " + "<EMPLEADOS>{$titulo} "+
"{for $em in /EMPLEADOS/EMP_ROW[DEPT_NO=10] return $em}</EMPLEADOS>";

XQPreparedExpression consulta = conn.prepareExpression(s);
XQResultSequence resultat = consulta.executeQuery();
// Escribimos los datos de la consulta en un fichero xml
BufferedWriter writer = new BufferedWriter(new FileWriter("empleados.xml"));
// Cabecera XML
writer.write("<?xml version='1.0' encoding='UTF-8'?>");
writer.newLine();
// Nodos XML
while (resultat.next()) {
    String cad = resultat.getItemAsString(null);
    System.out.println(cad);
    writer.write(cad);
    writer.newLine();
}
writer.close();
conn.close();
} catch (XQException ex) {
    System.out.println("Error al operar"+ex.getMessage());
}
```



The screenshot shows an IDE window with the following tabs: Problems, Javadoc, Declaration, Console, Progress, Debug, Git Staging, and Variables. The Console tab is active, displaying the output of the Java application. The output is an XML document with the following structure:

```
<terminated> Main (17) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (31 ene. 2021 19:49:01)
<EMPLEADOS>
  <TITULO xmlns="">DATOS DE LA TABLA EMPL</TITULO>
  <EMP_ROW xmlns="">
    <EMP_NO xmlns="">7782</EMP_NO>
    <APELLIDO xmlns="">CEREZO</APELLIDO>
    <OFICIO xmlns="">DIRECTOR</OFICIO>
    <DIR xmlns="">7839</DIR>
    <FECHA_ALT xmlns="">1991-06-09</FECHA_ALT>
    <SALARIO xmlns="">2885</SALARIO>
    <DEPT_NO xmlns="">10</DEPT_NO>
  </EMP_ROW>
  <EMP_ROW xmlns="">
    <EMP_NO xmlns="">7839</EMP_NO>
    <APELLIDO xmlns="">REY</APELLIDO>
    <OFICIO xmlns="">PRESIDENTE</OFICIO>
    <FECHA_ALT xmlns="">1991-11-17</FECHA_ALT>
    <SALARIO xmlns="">4100</SALARIO>
    <DEPT_NO xmlns="">10</DEPT_NO>
  </EMP_ROW>
```

6. PRACTICA 2

1. A partir del documento universidad.xml, haz un programa que muestre los empleados del departamento cuyo tipo es elegido por el usuario. Si no hay empleados o el tipo de departamento aportado por el usuario no existe, se debe de informar al usuario.
2. A partir de los documentos productos.xml y zonas.xml, haz un programa que reciba un número de zona por parámetro y genere un documento con nombre zonaXX.xml donde XX es la zona solicitada. El documento debe contener los productos de esta zona y las siguientes etiquetas: <cod_prod>, <denominación>, <precio>, <nombre_zona>, <director> y <stock>. Donde el stock se calcula restando el stock actual y el stock mínimo.