

Bases de datos xml.

Caso práctico

A la empresa BK Programación no dejan de llegar nuevos proyectos, está claro que sus productos finales convencen a sus clientes y éstos están bastante contentos con ellos.

Ana está orgullosa de haber participado y aportado su granito de arena en el desarrollo y mantenimiento de diferentes aplicaciones, algunas bastante complejas.

Este fin de semana se ha tomado un respiro. Sabe que el lunes comienza con algo nuevo. Debe ayudar a **María y Juan** en la tarea de compatibilizar diferentes formatos de datos que intercambian varias academias de Inglés de una misma firma.



1.- Introducción.Comparativa con BD relacionales

Caso práctico

Las academias gestionan la información relativa a sus cursos, alumnos, profesores, distribuidores de material, libros de lectura, artículos, apuntes, exámenes, etc. con programas diferentes y basados también en sistemas de bases de datos diferentes.

Hasta ahora, las academias han hecho uso de XML, para intercambiar los datos entre sus sistemas, pero cada vez surge algún problema nuevo, sobre todo con el almacenamiento, consulta y recuperación de ciertos documentos de texto. Por tanto, han decidido acudir a BK Programación para buscar la mejor solución. No les importa empezar desde cero y, si es necesario, utilizar una base de datos nativa XML, tal y como les ha sugerido Ada en su primera entrevista.

Ana apura la tarde del domingo pues esa misma noche va a comenzar su nueva aventura, ¡repasar XML y las tecnologías relacionadas! Ana, es muy profesional y no deja para mañana lo que puede hacer hoy.

Atendiendo al nivel de estructuración, podemos decir que existen tres tipos de datos:

- ✓ **Datos estructurados.** Son los que tienen un formato estricto. Toda la información recogida se ajusta al mismo formato, como por ejemplo: los datos tabulados en filas y columnas de una tabla.
- ✓ **Datos desestructurados.** No tienen ninguna estructura, como un documento de texto o un archivo de vídeo.
- ✓ **Datos semi-estructurados.** Tienen cierta estructura, pero no toda la información recogida tiene la misma forma, y además puede ir variando de manera dinámica.



Las bases de datos tradicionales, como las Bases de Datos Relacionales (BDR) son apropiadas para almacenar datos estructurados, pero la cuestión es que, en la actualidad, son muchas las situaciones en las que interesa almacenar grandes volúmenes de datos no estructurados, e incluso integrarlos con datos estructurados. Pero ¿cómo hacerlo? Aquí es donde entra en juego XML (eXtensible Markup Language).

XML define un conjunto de reglas semánticas que permiten la organización de información de distintas maneras. Es un estándar definido por el W3C y ofrece muchas ventajas, entre ellas:

- ✓ Es un lenguaje bien formado. No puede haber etiquetas sin finalizar.
- ✓ Extensible. Permite ampliar el lenguaje mediante nuevas etiquetas y la definición de lenguajes nuevos
- ✓ Fácil de leer:
- ✓ Autodescriptivo. La estructura de la información de alguna manera está definida dentro del mismo documento
- ✓ Intercambiable. Portable entre distintas arquitecturas.
- ✓ Para su lectura e interpretación es necesario un parser, y hay productos y versiones libres.

Debido fundamentalmente a estas ventajas y a su sencillez, **el estándar XML ha sido ampliamente aceptado y adoptado para el almacenamiento e intercambio de información**, y como consecuencia de este uso se ha creado la necesidad de almacenar dicha información.

Y ¿cómo se almacena la información en formato XML de cara a su recuperación y consulta? Existen varias aproximaciones para organizar y almacenar información XML que fundamentalmente van a depender de los diferentes archivos o documentos XML que nos podemos encontrar (centrados en datos y centrados en texto).

Pero en definitiva, lo que se busca es una estrategia que permita almacenar y recuperar datos poco estructurados con la eficiencia de las Bases de Datos convencionales, y es por ello que surgen las Bases de Datos XML.

Citas para pensar

El experimentador que no sabe lo que está buscando no comprenderá lo que encuentra

Claude Bernard

Para saber más

En el siguiente enlace puedes consultar diferentes tecnologías basadas en XML.

[Tecnologías XML](#) (pdf - 235.43 KB)

Las bases de datos relacionales centran su interés en la fiabilidad de las transacciones, bajo el conocido principio ACID (Atomicity, Consistency, Isolation and Durability o Atomicidad, Consistencia, Aislamiento y Durabilidad), al contrario de las bases de datos NoSQL siguen los principios de rendimiento, disponibilidad y escalabilidad.

PRINCIPIO ACID- BD RELACIONALES

Atomicidad	Asegura que la transacción se completa o no sin quedarse a medias.
Consistencia	Se asegura que los datos son correctos .
Aislamiento	Independencia de las transacciones
Durabilidad	Persistencia de la transacción ante cualquier fallo.

Actualmente los sistemas más modernos se centran el principio BASE.

PRINCIPIO BASE

Basic Availability	La prioridad es la disponibilidad de los datos.
Soft state	Se prioriza la propagación de datos, dejando el control de inconsistencias a agentes externos.
Eventually consistency	Se acepta que inconsistencias temporales progresen a un estado final estable.

NoSQL frente SQL

	NoSQL	SQL
Lenguaje de consultas	Cassandra-->CQL MongoDB-->JSON BigTable-->GQL	Lenguaje SQL
Modelos de almacenamiento	Sistemas clave-valor, objetos o grafos	estructuras fijas, tablas.
Operaciones JOIN	No son deseables por la sobrecarga al disponer de un volumen de datos grande .	Se realizan normalmente.
Arquitectura distribuida	Compartida en varias máquinas, mecanismos de tablas Hash distribuidas.	Centralizadas en una sola máquina o estructura máster-esclavo.

1.1.- Documentos XML centrados en datos y centrados en texto.

Como te hemos comentado antes, los tipos de documentos XML que nos podemos encontrar son:

- ✓ **Documentos centrados en datos**, con las siguientes características:
 - Muchos elementos de datos de pequeño tamaño.
 - Con estructura regular y bien definida.
 - Datos muy estructurados o semi-estructurados.
 - Dirigidos a utilización automática (por máquinas).
 - Ejemplos: Facturas, Pedidos Ficha de alumno.
- ✓ **Documentos centrados en texto, contenido o documentos** con las siguientes características:
 - Pocos elementos.
 - Con grandes cantidades de texto.
 - Con estructuras impredecibles en tamaño y contenido.
 - Datos poco estructurados.
 - Orientados a ser interpretados por humanos.
 - Enfocados a sistemas documentales y de gestión de contenidos.
 - Ejemplos: Libros, Informes, Memorias, Artículos bibliográficos.

En la siguiente imagen ampliable puedes apreciar las características indicadas para estos dos tipos de documentos.



Para saber más

Te proponemos el siguiente enlace para repasar el significado de documento XML bien formado.

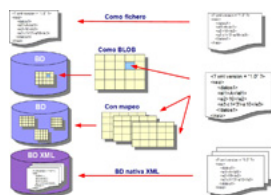
[Documento XML bien formado.](#)

1.2.- Opciones de almacenamiento.

Para almacenar documentos XML tenemos las siguientes opciones:

- ✓ **Almacenamiento directo del fichero.** Es una opción pobre ya que las operaciones que podemos hacer sobre ellos son limitadas, las que proporcione el sistema de archivos.
- ✓ **Almacenar el documento en una base de datos existente** (base de datos relacional, orientada a objetos u objeto-relacional). En este caso existen las siguientes posibilidades:
 - ➔ **Directamente como una columna tipo binario grande (BLOB) dentro de una tabla.** En este caso se almacena el documento intacto. Es una buena estrategia si el documento XML contiene contenido estático, que solo será modificado cuando se reemplaza el documento completo. Almacenar el documento completo en formato de texto es fácil de implementar dado que no se necesita mapeo o traducción, pero limita las consultas y búsquedas de contenido.
 - ➔ **Mediante mapeo basado en tablas, o basado en objetos.** En ambos casos hay que realizar una transformación para ajustarlo a la estructura de la BD antes de almacenarlo, y normalmente esto conlleva prescindir de cierta información, que supondrá que el documento y su formato no se almacena y recupera de manera intacta. Además, en el caso de documentos centrados en texto, no podemos controlar todas las posibles estructuras del documento, y por tanto realizar un mapeo sobre la base de datos será prácticamente imposible.
- ✓ **Almacenar el documento en una base de datos nativa XML.** El documento, tanto si es centrado en datos como en texto, se almacena y recupera de forma intacta. Como veremos, será la mejor opción, sobre todo si el documento está basado en texto.

La siguiente imagen ampliable ilustra las anteriores opciones de almacenamiento:



En la actualidad, cada vez más, las BD convencionales ofrecen posibilidades de almacenamiento XML, por lo que se habla de BD XML-compatible o BD XML-enabled. Por tanto, podemos hablar de dos **tipos de Sistemas de Bases de Datos que soportan documentos XML**:

- ✓ **BD XML-compatibles:** desglosan un documento XML en su correspondiente modelo relacional o de objetos.
- ✓ **BD XML Nativas:** respetan la estructura del documento, permiten hacer consultas sobre dicha estructura y recuperan el documento tal y como fue insertado originalmente

Las principales diferencias entre ambos tipos de BD XML son las siguientes:

- ✓ Una BD XML nativa proporciona un modelo de datos propio, mientras que un BD XML-compatible tiene ya un modelo de datos y añade una capa de software que permite de alguna manera almacenar documentos XML y recuperar los datos generando nuevos documentos XML.
- ✓ Una BD XML nativa debe manejar todos los tipos de documentos posibles, mientras que una BD XML-compatible solo puede manejar y almacenar los documentos que encajan dentro del modelo definido para ellos.

Para saber más

No dejes de visitar la página de Rounald Bourret, uno de los más importantes investigadores y consultores sobre BD XML. Aunque en inglés, en esta página encontrarás mucha información sobre el desarrollo de las bases de datos XML y sus especificaciones.

[Página de Rounald Bourret sobre BD XML.](#)

Autoevaluación

Las BD XML-compatibles recuperan el documento tal y como fue insertado originalmente.

☐ Verdadero ☐ Falso

Falso

No, esas son las bases de datos XML Nativas!

2.- Bases de Datos Nativas XML.

Caso práctico

Esta mañana, **María** ha estado hablando con una amiga del gremio. Su amiga lleva varios años trabajando en proyectos con tecnologías XML y además ha asistido personalmente a algunas conferencias del mismísimo Ronald Bourret, por lo que está más que al tanto de lo último en estas tecnologías.

María le ha pedido consejo sobre algunas de las bases de datos nativas XML, actualmente en el mercado.



Las **Bases de Datos nativas XML NXD** (Native XML Database) aunque existen desde hace años, en la actualidad siguen evolucionando, por lo que existen ciertas diferencias entre los productos de este tipo que podemos encontrar en el mercado. Lo que sí está bastante claro, es que **una NXD o BD XML debe cumplir las siguientes propiedades:**



- ✓ **Define un modelo lógico de datos XML**, estableciendo los elementos que son significativos, de forma que el documento XML se pueda almacenar y recuperar de manera intacta, esto es, con todos sus componentes. Por tanto, el modelo, como mínimo, debe tener en cuenta: los elementos, atributos, texto, secciones CDATA, y preservar el orden en el documento.
- ✓ **El documento XML es la unidad lógica de almacenamiento**, esto es, la unidad mínima de almacenamiento.
- ✓ **No tiene ningún modelo de almacenamiento físico subyacente concreto**. Pueden ser construidas sobre bases de datos relacionales, jerárquicas, orientadas a objetos o bien mediante formatos de almacenamiento propietarios.
- ✓ **Permite las tecnologías de consulta y transformación** propias de XML, XQuery, XPath, XSLT, etc., como vehículo principal de acceso y tratamiento.

Ventajas de las bases de datos XML:

Acceso y almacenamiento de información directamente en formato XML, sin necesidad de código adicional
 Motor de búsqueda de alto rendimiento, por lo general.
 Facilidad para incorporar nuevos documentos XML al repositorio.
 Almacenamiento de datos heterogéneos.

Inconvenientes de las bases de datos XML:

Dificultad para indexar documentos a la hora de realizar búsquedas.
 Se almacena la información en XML como un documento o como un conjunto de nodos, por lo que su síntesis para formar nuevas estructuras puede resultar compleja y lenta.

Y ¿en qué situaciones puede resultar imprescindible su uso? Fundamentalmente en las siguientes situaciones:

- ✓ Existencia de documentos con anidamientos profundos.
- ✓ Importancia de preservar la integridad de los documentos.
- ✓ Frecuentes búsquedas de contenido.

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

Una BD XML nativa solo puede almacenar documentos XML válidos.

☐ Verdadero ☐ Falso

Falso

Es falso, en la mayoría de las BDs XML nativas basta con que los documentos sean XML bien formados.

Para saber más

Consulta el siguiente enlace para recordar o bien familiarizarte con lo que son los DTDs y los esquemas XML.

[Los DTDs y esquemas XML.](#)

2.1.- Estrategias de almacenamiento.

Podemos diferenciar o clasificar las BD XML nativas en función del tipo de almacenamiento que utilicen, que puede ser:

- ✓ Almacenamiento basado en texto
- ✓ Almacenamiento basado en el modelo
- ✓ Soluciones desarrolladas específicamente para la gestión de documentos XML



¿En qué consiste el almacenamiento basado en texto y basado en modelo?

- ✓ El **almacenamiento basado en texto** consiste en almacenar el documento XML entero en forma de texto (fichero de texto), y proporcionar alguna funcionalidad de base de datos para acceder a él. Se suelen aplicar técnicas de compresión para reducir el espacio de almacenamiento, utilizar índices adicionales para mejorar el acceso a la información, y se pueden definir sobre BD tradicionales o sistemas de ficheros. Básicamente existen dos posibilidades:
 - ✦ Almacenar el documento como un binario largo (BLOB) en una base de datos relacional, o mediante un fichero, y proporcionar algunos índices sobre el documento que aceleren el acceso a la información.
 - ✦ Almacenar el documento en un almacén adecuado con índices, soporte para transacciones, etc.
- ✓ El **almacenamiento basado en modelo** consiste en definir un modelo de datos lógico, como DOM, para la estructura jerárquica de los documentos XML y almacenar el modelo binario del documento en un almacén existente o bien específico. En esta caso las posibilidades que existen son:
 - ✦ Traducir el DOM a tablas relacionales como elementos, atributos, entidades, etc.
 - ✦ Traducir el DOM a objetos en una BDDO.
 - ✦ Utilizar un almacén creado especialmente para esta finalidad

A continuación, te indicamos algunos ejemplos de BD XML nativas clasificadas según su sistema de almacenamiento:

- ✓ Sistema propietario: XIndex, Virtuoso, Tamino XML Server.
- ✓ Sistema relacional: eXist, DBCOM, XDB
- ✓ Sistema orientado a objetos: Ozone, MindSuite XDB.

¿Y qué ventajas proporcionan las BD XML nativas sobre otros sistemas de almacenamiento? Las **principales ventajas** son las siguientes:

- ✓ No necesitan mapeos adicionales.
- ✓ Conservan la integridad de los documentos.
- ✓ Permiten almacenar documentos heterogéneos.

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

El almacenamiento basado en modelo supone almacenar el documento como texto en un almacén adecuado con índices y que soporte transacciones.

☐ Verdadero ☐ Falso

Falso

Es falso, esta posibilidad de almacenamiento la contempla la estrategia de almacenamiento basado en texto.

Para saber más

Para refrescar tus conocimientos sobre el modelo de objetos de documentos DOM te recomendamos que visites el siguiente enlace. También obtendrás información sobre el parser SAX (Simple API for XML):

[Los modelos DOM.](#)

2.1.1 Ejemplo estructura XML

La estructura de un documento XML no tiene por qué conocerse a priori, ni siquiera todos los datos almacenados en el documento XML tiene por qué seguir la misma estructura.

Veamos un ejemplo:

```
<a href="https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/18258/mo
```

Elaboración propia (CC)

En el documento anterior XML, para una librería Libros dentro hay diferentes elementos libro.

Para saber más

En el siguiente enlace puedes ver ejemplos de documentos XML

<http://flanagan.ugr.es/xml/documento.htm>

2.2.- Colecciones y documentos.

En general, una BD XML tiene una estructura jerárquica organizada en colecciones y documentos XML. La estructura jerárquica comienza con un **nodo raíz ('/')**, del que parten colecciones y documentos.

Una colección:

- ✓ Es un conjunto de documentos agrupados, normalmente, en función de la información que contienen.
- ✓ Puede contener otras colecciones.

Un documento:

- ✓ Información XML
- ✓ Información de otro tipo y entonces se le denomina non-XML data (Datos no-XML)



Comparando con una base de datos relacional o un sistema de archivos:

- ✓ Las colecciones juegan en las bases de datos nativas el papel de las tablas en las DB relacionales, o de un directorio en un sistemas de archivos.
- ✓ Los documentos juegan el papel de las filas de una tabla de una BD relacional o un fichero en un sistema de archivos.

Aunque depende de cada implementación, en muchos casos:

- ✓ Cada colección puede tener más de un DOCTYPE asociado.
- ✓ El elemento raíz del documento XML define a que DOCTYPE estará asociado el documento, en caso de no poseer ninguno, éste se crea dinámicamente. Esto posibilita el almacenamiento de documentos sin formato definido.
- ✓ La colección también puede tener asociado un Schema con información tanto física como lógica de la colección. La parte lógica define las relaciones y propiedades de los documentos XML y la física contiene información sobre el almacenamiento e indexación de los mismos.
- ✓ También se pueden almacenar documentos no-XML, para estos existe un DOCTYPE especial llamado non-XML.

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

Una colección solo puede almacenar documentos XML.

☐ Verdadero ☐ Falso

Falso

Es falso, también puede almacenar colecciones y documentos non-XML.

Citas para pensar

A veces sentimos que lo que hacemos es tan solo una gota en el mar, pero el mar sería menos si le faltara una gota.
Madre Teresa de Calcuta

2.3.- Gestores nativos XML comerciales y libres

Te indicamos a continuación algunos ejemplos de gestores nativos XML clasificados según sean de código libre o código propietario. Los comerciales, generalmente ofrecen versiones de prueba con limitaciones de tiempo de uso y de funcionalidades. Los libres, a veces también ofrecen dualidad de licencia, comercial y libre.

Algunos de estos gestores nativos XML son los siguientes:



- ✓ **Gestores nativos XML comerciales** o de código propietario son:
 - ✓ **TaminoXML Server.** Es un gestor nativo XML de la empresa SoftwareAG. Es un producto comercial de alto rendimiento y disponibilidad, además de ser uno de los primeros SGBD XML nativos disponibles. Algunas de sus características son las siguientes:
 - ✦ Los documentos se almacenan en una base de datos propia y no se transforman en otro modelo.
 - ✦ Existe un espacio separado para documentos y para índices.
 - ✦ Soporta el lenguaje de consultas XQuery y APIs para Java, C, y .NET, entre otras.
 - ✓ **TEXTML** de Isiasoft. Los documentos se almacenan en su formato nativo, sin ser mapeados.
 - ✦ Permite almacenar documentos sin DTD o esquema.
 - ✦ Proporciona índices de acuerdo a la propia estructura del documentos.
 - ✦ Permite la utilización de varios índices al mismo tiempo.
 - ✦ Incluye API para Java, WebDAV, OLE DB y .NET.
 - ✓ **Qizx.** Es una base de datos nativa XML escrita en Java. Almacena los datos utilizando una representación que se basa en el modelo de XPath 2.0.
 - ✦ La representación utiliza compresión de manera que un documento y sus índices suelen utilizar menos espacio que el original XML.
 - ✦ Los documentos se almacenan en una jerarquía de colecciones.
 - ✦ Permite almacenar documentos sin DTD o esquema.
 - ✦ Soporta el lenguaje de consulta XQuery y sus extensiones, como XUpdate, así como API para Java.

Gestor nativos XML libre o de código abierto:

- ✓ **eXist.** Utiliza un sistema de almacenamiento propio (árboles B+ y archivos paginados). Se puede ejecutar como un servidor de base de datos independiente, como una biblioteca de Java embebida, o en el motor servlet de una aplicación Web.
 - ✦ Los documentos se almacenan en una jerarquía de colecciones.
 - ✦ Permite almacenar documentos sin DTD o esquema.
 - ✦ Soporta el lenguaje de consulta XQuery y sus extensiones, como XUpdate, así como API para Java.

Para saber más

Puedes consultar una relación extensa de bases de datos nativas XML, comerciales y libres, en el siguiente enlace:

[Relación de BD XML nativas, comerciales y libres.](#)

Si no sabes lo que es un árbol B+, consulta este enlace para conocer estas estructuras de datos:

[Información sobre árboles B+](#)

Citas para pensar

El éxito no se logra sólo con cualidades especiales. Es sobre todo un trabajo de constancia, de método y de organización.

J.P. Sergent

3.- Base de datos eXist

Caso práctico

Juan y María están muy satisfechos con **Ana**, realmente les está ayudando bastante con sus conocimientos de XML. Además, justo el gestor XML nativo que propuso **Ana** desde el principio, es uno de los que a **María** le sugirió su amiga. **Juan** le pregunta a **Ana** —¿Has trabajado antes con el gestor eXist? **Ana** le contesta —Pues....., sí. En clase, hicimos algunas prácticas con esta BD XML y realmente me gustó, es sencilla y está optimizada para consultas. Además es software libre. A lo que **Juan** le dice —Pues ahora es tu momento, serás nuestra profesora estos primeros días, para familiarizarnos con eXist.



En los siguientes apartados trabajaremos con el sistema gestor de bases de datos eXist que es un sistema libre y de código abierto.

Características principales del SGBD eXist:

- libre de código abierto.

- Motor de la BD escrito en JAVA

- soporta estándares de consulta XPath, XQuery, XSLT, indexación de documentos y soporte para actualización

eXist-db se ejecuta en cualquier sistema en el que se ejecuta Java. Así que todas las versiones recientes de Linux, macOS y Windows están bien. Se deben cumplir los siguientes requisitos:

- Al menos Java versión 8 (desde eXist-db 3.0).

- Aproximadamente 200Mb de espacio en disco para la instalación.

- Al menos 512Mb de memoria para ejecutar.

Debes conocer

eXist-db se basa en Java, un entorno de desarrollo de software multiplataforma y gratuito, es importante que use la versión que requiere eXist-db: Java 8 (desde eXist-db 3.0). Para obtener instrucciones sobre cómo ejecutar Java en su sistema operativo, consulte la página de [instalación de Java de Oracle](#) . Se recomienda utilizar la máquina virtual Java de Oracle, [OpenJDK](#) es una buena alternativa de código abierto para la JVM de Oracle.

3.1.- Instalación eXist

En este punto vamos a proceder a la instalación del gestor eXist . Puedes proceder a la descarga de la última versión en el siguiente enlace.

[Descarga eXist.](#)

Además en el siguiente vídeo puedes ver como realizamos la descarga desde la página anteriormente mencionada.

0:00

[Resumen contextual](#)

Una vez descargado el archivo .jar vamos a proceder a la instalación, esta puede realizarse invocando el archivo desde la línea de comandos o bien haciendo doble clic sobre el icono correspondiente una vez descargado, mira el siguiente vídeo en el que se ven los pasos de instalación. (Nota: el vídeo no tiene audio, no es necesario).

0:00

[Resumen contextual](#)

Autoevaluación

eXistdb no soporta el estándar de consulta XQuery.

☐ Verdadero ☐ Falso

Falso

Correcto!!, como veíamos en el apartado anterior eXistdb soporta los estándares de consulta XPath, XQuery, XSLT, e indexación de documentos

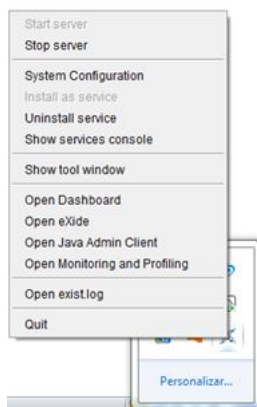
Debes conocer

La documentación sobre la instalación de eXistdb la podéis encontrar en el siguiente enlace:

[Documentación instalación](#)

3.1.1.-Arranque de eXist

Una vez instalada la aplicación vamos a proceder a arrancarla. Esto puede realizarse desde el acceso directo del escritorio (si en la instalación dijimos que nos lo pusiera) o bien lanzando el fichero start.jar dentro de la carpeta de la instalación.



Elaboración propia (CC)

Lo primero que debes hacer es arrancar el servidor en start server.

Una vez arrancado el servidor para trabajar con la base de datos (hacer consultas y operaciones con la misma) se nos presentan las siguientes opciones:

Seleccionar la opción eXide (Open eXide): al seleccionar esta opción se abrirá el navegador directamente con la herramienta que nos va a permitir crear y borrar colecciones, abrir nuevos documentos y realizar consultas.

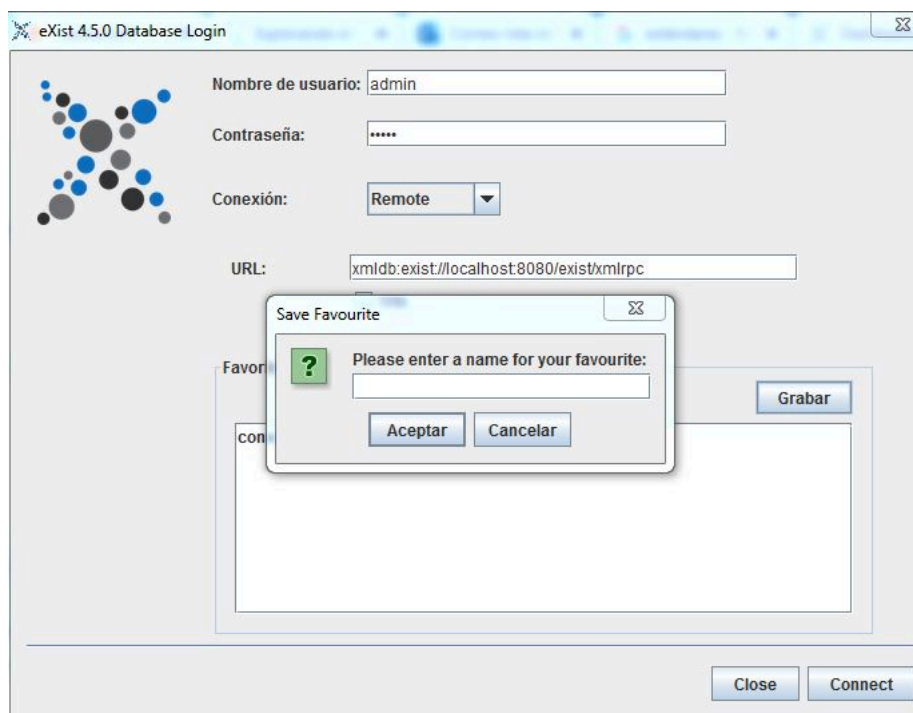


Elaboración propia (CC)

Seleccionar la opción **dashboard**: en esta opción se nos presenta el administrador de las aplicaciones de la BD. En ella encontramos aplicaciones como **eXist-bd documentation**, que contiene la documentación, **eXide-Xquery** el entorno para realizar consultas y **eXist-bd Demo Apps** la aplicación de demostración. En cuanto a los plugins encontramos el gestor de colecciones **Collections**, el gestor de Backups **Backup**, el gestor de usuarios **User Manager** y el gestor de paquetes **Package Manager**. En conclusión la parte de aplicaciones es la que va a gestionar la base de datos a nivel de consultas y la parte de plugin es la que gestiona la parte administrativa.



Seleccionar el **cliente Java** (Open Java Admin Client) : Esta será la opción en la que más trabajaremos durante la unidad. Cuando seleccionamos esta opción se nos abrirá la siguiente ventana:



Nombre de usuario y contraseña: los establecidos durante la instalación, normalmente admin y admin.

Conexión: Remote.

URL: verificar que el puerto es correcto aparecerá el 8080, pero a veces este puerto puede estar ocupado con algún otro servidor como por ejemplo Apache. En este caso habría que cambiarlo al 8083.

Al introducir los datos y darle a grabar nos pedirá un nombre para la conexión,.

Para iniciar la conexión seleccionamos el nombre de la conexión y le damos a conectar.

Para saber más

Si quieres consultar más en profundidad el dashboard de eXistdb, puedes hacerlo en este enlace:

[Explorando el dashboard](#)

Autoevaluación

El cliente java de eXist no nos permite realizar consultas, únicamente nos sirve para gestionar los usuarios de la base de datos

☐ Verdadero ☒ Falso

Falso

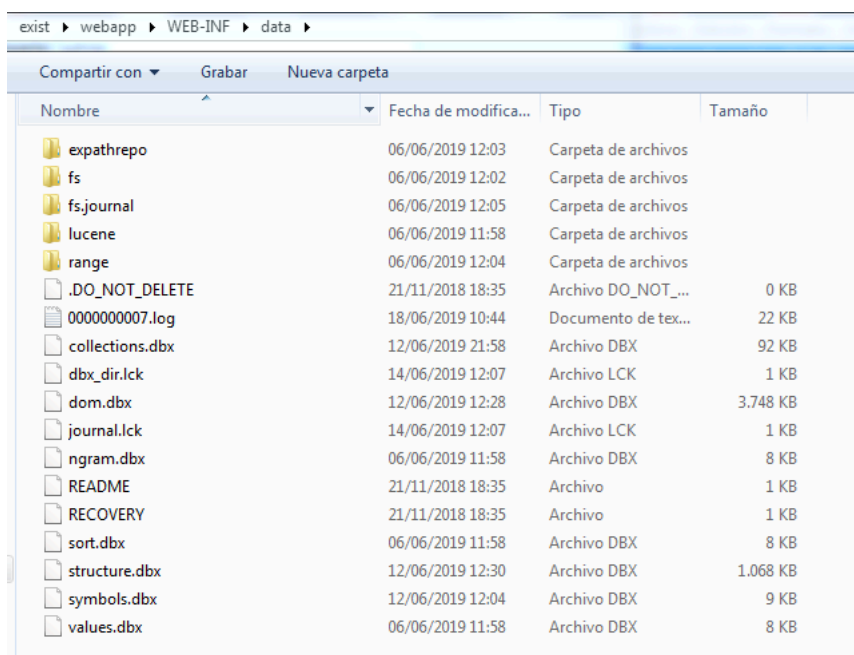
el cliente eXist no solo nos va a permitir realizar consultas si no también cualquier operación relacionada con la misma como crear y borrar colecciones.

3.2.- Gestión BD XML mediante eXist

Los documentos XML se almacenan en **colecciones**, estas colecciones pueden estar anidadas. Esto es algo similar a lo que ocurre en un sistema de ficheros con las carpetas y los ficheros. Cada colección sería una carpeta y cada fichero un documento. Los documentos se encuentran en colecciones.

Te detallamos a continuación algunos aspectos que debes tener en cuenta con **eXist**:

- ✓ en la carpeta eXist\weapp\WEB-INF\data están los archivos más importante de la BD, dentro de esta podemos encontrar:
 - **collections**: jerarquía de colecciones y su relación con los documentos. Cada documento tiene un identificador único que se almacena junto con su índice.
 - **dom.dbx**: es el almacén de datos nativo. En este fichero se almacenan los nodos del documento de acuerdo al modelo **DOM de W3C**.



Nombre	Fecha de modifica...	Tipo	Tamaño
expathrepo	06/06/2019 12:03	Carpeta de archivos	
fs	06/06/2019 12:02	Carpeta de archivos	
fsjournal	06/06/2019 12:05	Carpeta de archivos	
lucene	06/06/2019 11:58	Carpeta de archivos	
range	06/06/2019 12:04	Carpeta de archivos	
.DO_NOT_DELETE	21/11/2018 18:35	Archivo DO_NOT_...	0 KB
0000000007.log	18/06/2019 10:44	Documento de tex...	22 KB
collections.dbx	12/06/2019 21:58	Archivo DBX	92 KB
dbx_dir.lck	14/06/2019 12:07	Archivo LCK	1 KB
dom.dbx	12/06/2019 12:28	Archivo DBX	3.748 KB
journal.lck	14/06/2019 12:07	Archivo LCK	1 KB
ngram.dbx	06/06/2019 11:58	Archivo DBX	8 KB
README	21/11/2018 18:35	Archivo	1 KB
RECOVERY	21/11/2018 18:35	Archivo	1 KB
sort.dbx	06/06/2019 11:58	Archivo DBX	8 KB
structure.dbx	12/06/2019 12:30	Archivo DBX	1.068 KB
symbols.dbx	12/06/2019 12:04	Archivo DBX	9 KB
values.dbx	06/06/2019 11:58	Archivo DBX	8 KB

Elaboración propia (CC0)

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

Un XML Group Library es el equivalente a una BD XML.

☐ Verdadero ☐ Falso

Falso

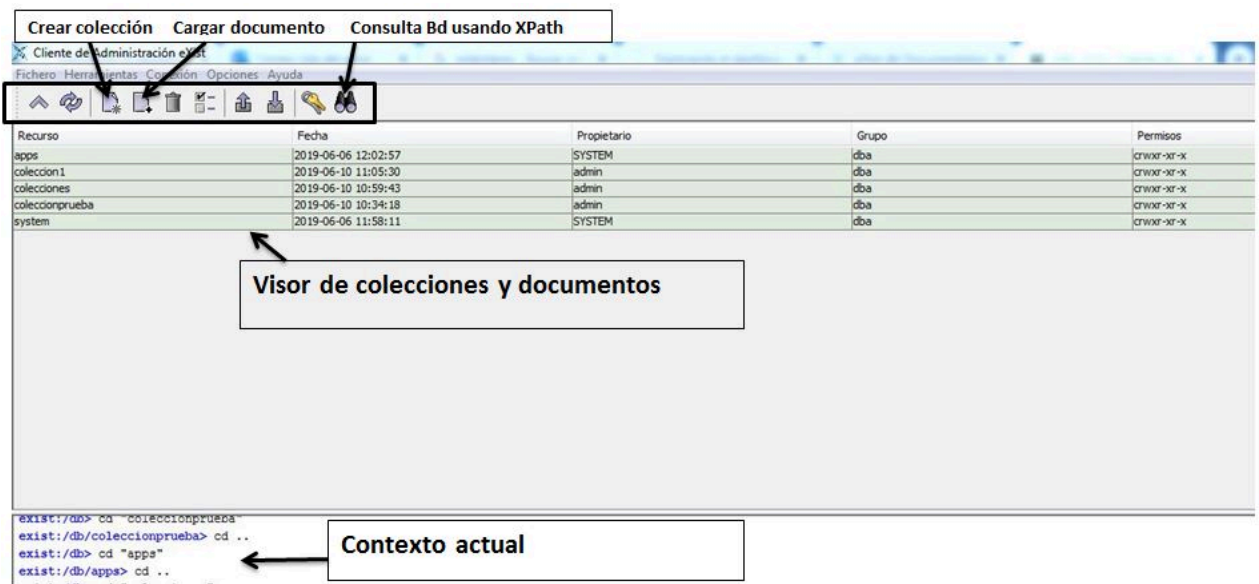
Es falso, el equivalente a una BD XML es una Library.

Para saber más

En el siguiente enlace puede encontrar información sobre el [modelo DOM](#)

3.2.1.- Cliente Java eXist

Al ejecutar el cliente eXist nos encontramos con el siguiente entorno:



Elaboración propia (CC0)

Crear colección: creamos una colección por ejemplo le ponemos el nombre coleccion1

Cargar documento: cargamos el documento xml sobre el que vamos a trabajar, en este caso el documento prueba.xml

comprobar contexto: debemos verificar que el documento se encuentra dentro del contexto sobre el que queremos trabajar en nuestro caso será **exist/db/coleccion1**

Para ejecutar consultas sobre esa colección tendremos que hacer doble clic sobre ella y pulsar sobre el icono



que nos abrirá la ventana para realizar consultas:

Elaboración propia (CC0)

Debes tener en cuenta que las consultas se están ejecutando sobre el contexto que has seleccionado para ello comprueba que en el dialogo de consulta en el cuadro Contexto aparece la colección que contiene el documento xml sobre el que vas a trabajar..

4.- El lenguaje de consultas XQuery.

Caso práctico

Ana está orgullosa en su nueva faceta de profesora. **Juan** y **María** la han felicitado y están satisfechos con la introducción que les ha dado sobre **eXist**. **Ana** les dice —Antes de empezar con la aplicación en Java, necesitamos familiarizarnos con el lenguaje de consultas XQuery.

A lo que **Juan** le pregunta —¿Conoces algo del lenguaje?"

Ana sonríe y contesta —Pues...sí. En clase vimos también algunos ejemplos sencillos de uso.

Juan, con cara de asombro dice —Está claro que el nuevo Ciclo Formativo de **DAM** incorpora muchos temas punteros que yo no domino. El próximo curso iniciaré sin falta el estudio de algunos módulos.



XQuery es un lenguaje de consulta diseñado para extraer información de colecciones de datos expresadas en XML. Entre las **principales características de XQuery** vamos a destacar las siguientes:

- ✓ Está **basado en el lenguaje XPath**, (XML Path Language), y se fundamenta en él para realizar la selección de información y la iteración a través del conjunto de datos XML.
- ✓ Es un **lenguaje declarativo**, lo que significa que, en vez de ejecutar una lista de comandos como un lenguaje procedimental clásico, cada consulta es una expresión que es evaluada y devuelve un resultado, al igual que en SQL.



Podemos decir que **XQuery es a XML lo mismo que SQL es a las bases de datos relacionales**. Sin embargo, aunque XQuery y SQL puedan considerarse similares, el modelo de datos sobre el que se sustenta XQuery es muy distinto del modelo de datos relacional sobre el que sustenta SQL, ya que XML incluye conceptos como jerarquía y orden de los datos que no están presentes en el modelo relacional.

Por ejemplo, a diferencia de SQL, en XQuery es importante y determinante el orden en que se encuentren los datos, ya que no es lo mismo buscar una etiqueta nombre dentro de una etiqueta autor que todas las etiquetas nombre del documento (que pueden estar anidadas dentro de una etiqueta autor o fuera).

Los **principales tipos de expresiones de XQuery** son:

- ✓ Expresiones XPath, para navegar por los documentos.
- ✓ Expresiones FLWOR (For, Let, Where, Order, Return) para iterar por los elementos de un conjunto de datos.

Pero XQuery también admite:

- ✓ Constructores para generar nodos y contenido dinámico.
- ✓ Condicionales (IF, THEN ELSE) para construir el resultado en base a alguna condición.
- ✓ Cuantificadores (SOME, ANY) para chequear la existencia de algún elemento que cumpla una condición.
- ✓ Listas a las que se pueden aplicar operadores (UNION,...) y funciones de agregación (AVG, COUNT,...).

Para saber más

En este enlace puedes profundizar sobre el lenguaje XQuery.

[Especificación completa de XQuery](#)

4.1.- Modelo de datos.

El modelo de datos en que se sustenta XQuery es el modelo de datos de XPath.

XPath modela un documento XML como una estructura jerárquica en forma de árbol. El árbol está formado por nodos, y hay siete tipos de nodos: raíz, elemento, texto, atributo, espacio de nombres, instrucción de procesamiento y comentario.



Los **principales nodos** de la estructura jerárquica o en árbol en un documento XML son: (puedes verlos en la imagen ampliable superior)

- ✓ **Nodo raíz o /**. Es el primer nodo del documento.
- ✓ **Nodo elemento**. Cualquier elemento de un documento XML. Cada nodo elemento posee un padre y puede o no tener hijos. En el caso de que no tenga hijos, es un nodo hoja.
- ✓ **Nodo texto**. Cualquier elemento del documento que no esté marcado con una etiqueta de la DTD del documento XML.
- ✓ **Nodo atributo**. Un nodo elemento puede tener etiquetas que complementen la información de ese elemento. Esto sería un nodo atributo.

Debes conocer

En el siguiente enlace tienes una ilustración gráfica más detallada del paso de un documento XML a un árbol de nodos XML, así como la explicación en detalle de cada tipo de nodo. (Apartado 2. Modelo de datos XPath)

[Nodos de un árbol XML.](#)

Autoevaluación

Señala la opción correcta. Entre los nodos de la estructura jerárquica de un árbol XML están:

- ☐ Nodo etiqueta y nodo atributo.
- ☐ Nodo padre.
- ☐ Nodo hijo.
- ☐ Nodo raíz y nodo elemento.

Es incorrecto, prueba de nuevo.

No es correcto, deberías haber leído mejor.

No, ese método no existe.

Muy bien, vamos por buen camino.

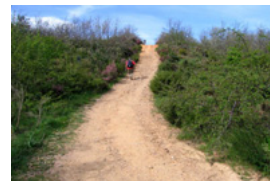
Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

4.2.- Caminos de localización.

¿Cómo se localiza cada uno de esos nodos en el árbol XML? Mediante una expresión XPath conocida como camino o ruta de localización. **Un camino de localización:**

- ✓ Selecciona un conjunto de nodos relativo al nodo de contexto.
- ✓ Puede contener recursivamente expresiones utilizadas para filtrar conjuntos de nodos.
- ✓ Al ser evaluado, devuelve el conjunto de nodos seleccionados por el camino de localización.
- ✓ Se construye siguiendo unas reglas de sintaxis y semántica.



Hay dos **tipos de caminos de localización**:

- ✓ **Caminos relativos.** Son una secuencia de uno o más pasos de localización separados por /.
 - Los pasos se componen de izquierda a derecha.
- ✓ **Caminos absolutos.** Consiste en / seguido, opcionalmente, por un camino de localización relativo.
 - Una / por sí misma selecciona el nodo raíz del documento que contiene al nodo contextual.

Los siguientes, son **algunos ejemplos de caminos de localización**:

- ✓ **cuadro** selecciona los elementos cuadro hijos del nodo contextual.
- ✓ **cuadro/titulo** selecciona los elementos titulo descendientes de los elementos cuadro hijos del nodo contextual.
- ✓ ***** selecciona todos los elementos hijos del nodo contextual.
- ✓ **@año** selecciona el atributo año del nodo contextual
- ✓ **@*** selecciona todos los atributos del nodo contextual
- ✓ **cuadro[1]** selecciona el primer hijo cuadro del nodo contextual
- ✓ **cuadro[@año=1907]** selecciona todos los hijos cuadro del nodo contextual que tengan un atributo año con valor 1907.

Debes conocer

En el siguiente enlace tienes ejemplos de caminos de localización en la sintaxis abreviada de XPath (Es el apartado 2.5 Sintaxis abreviada)

[Caminos de localización XPath.](#)

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

El camino: //curso selecciona los elementos curso descendientes del nodo contextual.

☐ Verdadero ☐ Falso

Falso

Es falso, selecciona todos los descendientes curso del raíz del documento.

Citas para pensar

Razonar y convencer, ¡qué difícil, largo y trabajoso! ¿Sugestionar? ¡Qué fácil, rápido y barato!

Santiago Ramón y Cajal

4.3.- Primeras consultas XQuery.

Una **consulta XQuery** es una expresión que lee una secuencia de datos en XML y devuelve como resultado otra secuencia de datos en XML, donde:

- ✓ Una **secuencia** es un conjunto ordenado de cero o más ítems.
- ✓ Un **ítem** es cualquier tipo de nodo del árbol XML o un valor atómico.

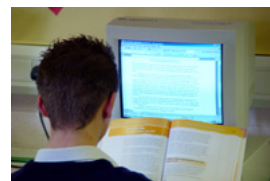
Las funciones que se pueden invocar para **referirnos a colecciones y documentos dentro de la BD** son las siguientes:

- ✓ `collection(/ruta)`: indicamos el camino para referirnos a una colección.
- ✓ `doc(/ruta/documento.xml)`: indicamos el camino de un documento dentro de una colección.

En la siguiente presentación hay diferentes ejemplos de consultas XQuery (algunas incluyen expresiones FLWOR que vemos en el siguiente apartado) ejecutadas directamente sobre una sencilla BD XML denominada 'Cursillos'. Necesitarás descargar el archivo:

[Acceso a los ejemplos.](#) (0.01 MB)

[Presentación consultas eXist](#) (zip - 287.9 KB).



Debes conocer

Si no indicamos esas funciones la BD busca los elementos en el contexto actual.

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

La expresión `/curso[aula=2]/profesor` devuelve los profesores con cursos en el aula 2.

☐ Verdadero ☐ Falso

Verdadero

Devuelve todos los profesores que dan cursos en el aula 2.

Para saber más

Te recomendamos que visites el siguiente enlace para profundizar en el concepto y uso de los namespaces.

[Espacios de nombres o namespaces en XML.](#)

Además en el siguiente documento podrás profundizar más sobre el lenguaje Xquery

[Xquery.](#)

4.4.- Expresiones FLWOR.

Los ejemplos de consultas XQuery vistos en el apartado anterior, son la manera más sencilla de realizar búsquedas y selecciones de nodos concretos en una BD XML. Pero existe otra manera mucho más potente de realizar este trabajo, mediante lo que se denomina consultas o **expresiones FLWOR** (leído como flower), siendo FLWOR las siglas de For, Let, Where, Order y Return.

Se trata de una **expresión que permite la unión de variables sobre conjuntos de nodos y la iteración sobre el resultado**. Las diferentes **cláusulas de una expresión FLWOR** son:

- ✓ **For**. Permite seleccionar los nodos que se quieren consultar, guardándose su valor en una variable (identificador que comienza por \$). Al conjunto de valores de la variable se le llama **tupla**.
- ✓ **Let**. (opcional). Asocia valores a variables.
- ✓ **Where** (opcional). Permite filtrar los resultados según una condición.
- ✓ **Order** (opcional). Permite ordenar la secuencia de valores o resultados.
- ✓ **Return**. Genera los valores de salida o devueltos.

La sintaxis general de una estructura **FLWOR** es esta:

for in

let

where

order by

return

For: se usa para seleccionar nodos y almacenarlos en una variable, es algo parecido a la cláusula **from** de SQL. A continuación hay que especificar una variable XPath que es la que se va a encargar de seleccionar los nodos. Hay que tener en cuenta que si se especifica más de una variable en el for lo que se va a realizar es el producto cartesiano de ambas. Para especificar variables estas deben ir precedidas del símbolo \$.

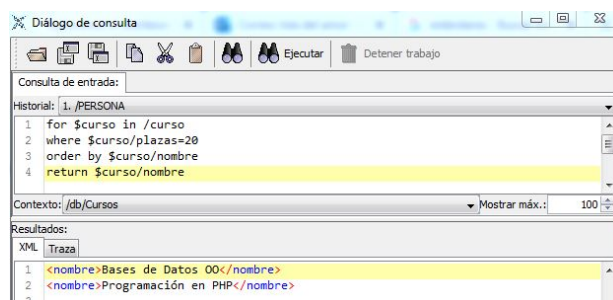
let: sirve para asignar valores resultantes de las expresiones XPath a variables. Es posible poner varias líneas let separadas por comas por cada variable.

Where: aquí se especifican las condiciones que se quiera que cumplan los valores seleccionados.

Order by: ordena los datos según el criterio que especifiquemos.

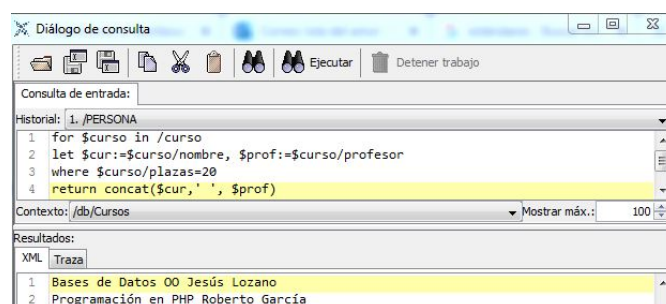
Return: se utiliza para devolver el resultado. Aquí podemos especificar etiquetas XML encerrándolas entre llaves {}. Otra opción que presenta el return es la posibilidad de utilizar cláusulas condicionales **if- then- else**, hay que tener en cuenta que si utilizamos esta cláusula es obligatorio poner siempre el **else**. Esto es debido a la naturaleza de las consultas XQuery en las que es obligatorio devolver siempre un valor. En el caso de que la consulta no devolviese ningún valor por no cumplir con las condiciones entonces devolvemos una secuencia vacía con **else()**.

En la siguiente imagen, puedes ver un ejemplo de una consulta (izquierda) donde aparecen las 5 cláusulas. La consulta devuelve los nombres de los cursos con 20 plazas, ordenados por nombre de curso (derecha).



Elaboración propia. (CC0)

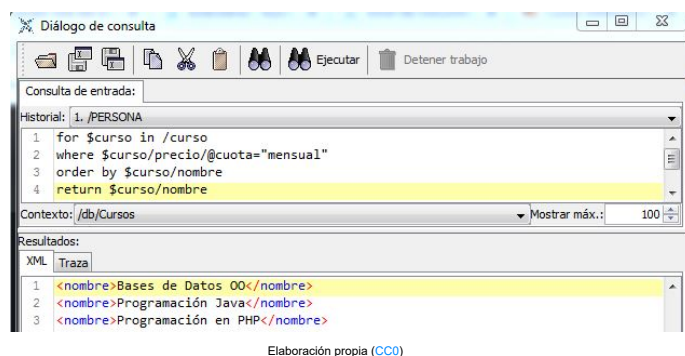
Otro ejemplo de consulta: en este caso la consulta devuelve además del nombre del curso el nombre del profesor que lo imparte, para esto hemos utilizado dos variables la primera almacena el nombre del curso (\$cur) y la segunda almacena el nombre del profesor (\$prof). Utilizamos concat para concatenar el resultado de la consulta.



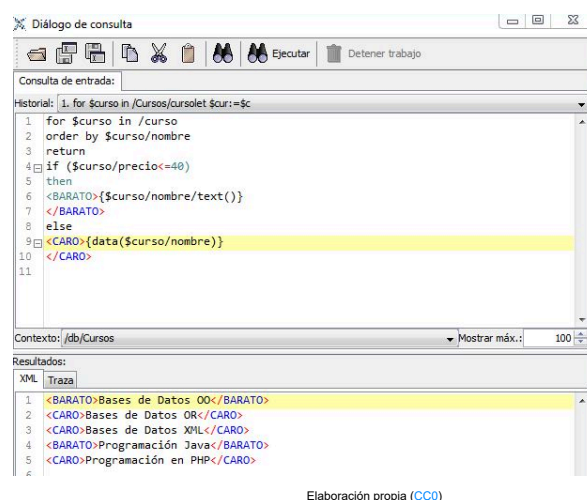
Elaboración propia. (CC0)

Una expresión FLWOR vincula variables a valores con las cláusulas for y let y utiliza esos vínculos para crear nuevas estructuras de datos XML.

A continuación se muestra otro ejemplo de consulta XQuery. En este caso se devuelven los nombres de los cursos con cuota mensual. Como cuota es un atributo del elemento precio, recuerda que se le antecede con un carácter @.



En el siguiente ejemplo puedes ver la utilización en el return de la cláusula condicional **if-then-else**. En esta consulta se obtienen los nombres de los cursos cuyo precio sea menor o igual que 40 con la etiqueta xml **BARATO** y en si tienen un precio superior aparecerán los nombres de los cursos con la etiqueta **CARO**.



En el siguiente recurso didáctico dispones de más ejemplos de consultas XQuery, utilizando expresiones FLWOR.

[Consultas XQuery con expresiones FLWOR](#) (zip - 163.96 KB) .

Para saber más

En el siguiente enlace tienes muchos ejemplos de consultas XQuery con expresiones FLWOR.

[Ejemplos XQuery con expresiones FLWOR](#)

Autoevaluación

La cláusula **for** se usa para seleccionar y almacenarlos en una Las variables deben ir precedidas del símbolo .

La cláusula **let** asocia a variables.

La cláusula **return** sirve para devolver el resultado se pueden especificar etiquetas encerrándolas entre { }. También podemos utilizar cláusulas condicionales if-then-else teniendo en cuenta que es obligatorio poner la cláusula .

Enviar

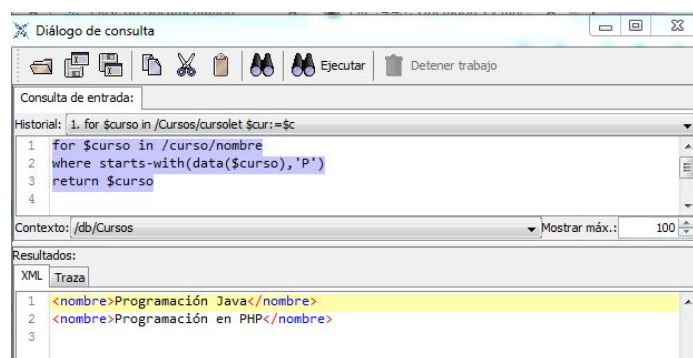
4.4.1.-Operadores y funciones XQuery

Xquery soporta operadores y funciones matemáticas, con cadenas, comparaciones con fechas, conversión de tipos.

OPERADORES MATEMÁTICOS	+, -, *, /
COMPARATIVOS	=, !=, >, <, <=, >=
AGRUPACIÓN	count(), min(), max(), sum().
CADENAS	concat(), string(), string-length(), starts-with, end-with
COMENTARIOS	Para poner comentarios en XQuery se pone: (: escribo mi comentario:)

A continuación algunos ejemplos utilizando algunas funciones:

En esta consulta se seleccionan los cursos cuyo nombre empieza por P.



Elaboración propia (CC0)

Cursos cuyo número de plazas sean mayores que 20 y menores que 30.

Elaboración propia (CC0)

Para saber más

En el siguiente enlace podrás practicar las funciones y operadores más utilizadas en XQuery, puedes como ejercicio probarlos en eXist.

[Ejemplos Xquery.](#)

4.5.- Actualización en eXist

En este caso nos vamos a centrar en la manera de insertar, eliminar y modificar nodos y elementos en documentos XML.

Todas las sentencias de actualización comienzan con la palabra **UPDATE** y a continuación:

INSERT: para insertar nodos, seguido de:

- **INTO**: si queremos añadir el contenido al último hijo especificado.
- **FOLLOWING**: si queremos añadir el contenido inmediatamente después de los nodos especificados.
- **PRECEDING**: si queremos añadir el contenido inmediatamente antes de los nodos especificados.

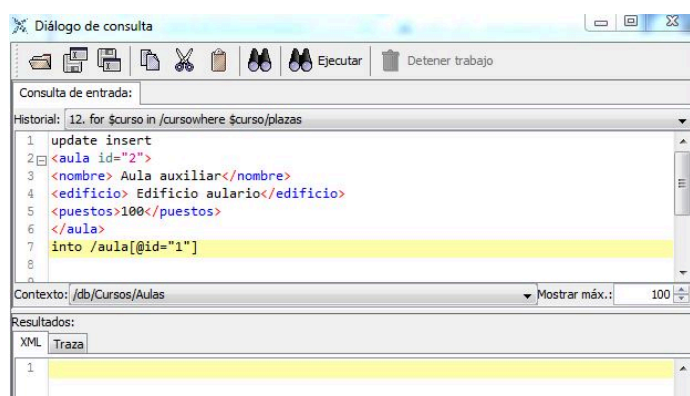
La sintaxis para **INSERT** sería:

update insert CONTENIDO into EXPRESIÓN XPATH

update insert CONTENIDO following EXPRESIÓN XPATH

update insert CONTENIDO preceding EXPRESIÓN XPATH

En el siguiente ejemplo hemos insertado un aula dentro del aula 1 con la clausula **INTO**.



Elaboración propia (CC0)

Elaboración propia (CC0)

REPLACE: sustituye bien un nodo especificado con el valor nuevo que le especifiquemos. Hay que tener en cuenta que nodo debe devolver un único elemento.

La sintaxis para **REPLACE** es:

update replace NODO with VALOR_NUEVO, siendo NODO el nodo a remplazar que debe devolver un único valor y VALOR_NUEVO el nuevo contenido al que queremos actualizar.

En el siguiente ejemplo vamos a modificar el nombre del nodo por el del y el contenido del nombre por el de Aula primera.

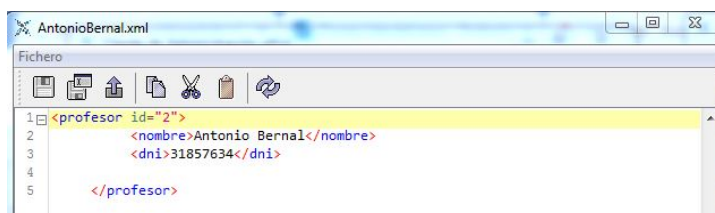
Elaboración propia ([CC0](#))

Elaboración propia ([CC0](#))

DELETE: elimina los nodos que le indiquemos en la expresión: **update delete expr xpath**

En el siguiente ejemplo podemos ver como hemos eliminado el nodo email al profesor con id=1.

Elaboración propia ([CC0](#))



Elaboración propia ([CC0](#))

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

Mediante update insert podemos insertar un nodo nuevo, pero si especificamos INTO el contenido se añade antes de los nodos especificados.

☐ Verdadero ☐ Falso

Falso

Es falso, con INTO el contenido se añade como último hijo de los nodos especificados.

Para saber más

En el siguiente enlace puedes otros ampliar el contenido de actualización con eXist:

[XQuery Update Extension](#)

5.- Trabajar con colecciones y documentos desde Java.

Caso práctico

—¡Ha llegado la hora de manejar la base de datos desde Java! —exclama **Ana**, y añade— ¿queréis que os indique cómo crear colecciones y añadir documentos?

Juan contesta —Pues claro **Ana**, ¿sabes cual es el API principal que proporciona este gestor para trabajar con Java?

Ana contesta —De memoria no lo recuerdo, pero sí recuerdo que la propia distribución dispone de datos de prueba que podemos utilizar en nuestros primeros ejemplos.

María, que escucha con atención dice —¡Perfecto!, vamos a ello.



En este apartado nos centraremos en las librerías utilizadas para acceder desde Java a sistemas XML nativos.

XML:DB; es la librería Java de acceso a sistemas XML más conocida y utilizada desde 2001, y la que más compatibilidad tiene con la mayoría de SGBD. Su objetivo principal es definir un método común de acceso a sistemas nativos que permita realizar consultas, creación y modificaciones. Podemos considerarla como una equivalencia de los sistemas ODBC y JDBC.

Su estructura está compuesta de:

Driver: encapsula la lógica de acceso a una BD XML. Cada sistema debe implementar el suyo, son proporcionados por el proveedor y deben ser registrado.

Collection: es el contenedor de recursos y otras subcolecciones.

Resource: representa el recurso en sí. La API define dos:

- ✦ **XMLResource** : representa un documento XML o un fragmento de documento, seleccionado por una consulta XPath ejecutada previamente.
- ✦ **BinaryResource**.: representa una secuencia binaria, como un mapa de bits.
- ✦ Hay que tener en cuenta que los recursos utilizados posteriormente se deben cerrar.

Service: Los servicios se solicitan para tareas especiales, como por ejemplo ejecutar consultar una colección XPath o administrar una colección.

En la siguiente presentación puede ver los primeros pasos para integrar eXist en Netbeans

[Integración en NetBeans](#) (zip - 70.36 KB) .

En el siguiente fichero .zip puedes encontrar las librerías necesarias para la integración de eXist en Netbeans, de todas formas también las tienes en la instalación de eXist en la carpeta lib/core y en la carpeta de instalación.

[Librerías XML:DB](#) (zip - 5.95 MB) , (zip - 6.72 MB) .

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

XML:DB puede considerarse como una equivalencia en los sistemas XML nativos a alternativas como ODBC y JDBC.

☐ Verdadero ☐ Falso

Verdadero

Es verdadero, es la librería de acceso a sistemas XML nativos más conocida y utilizada desde 2001

5.1.- Creación de la BD XML. Establecer conexiones.

En general, la conexión de una aplicación Java a una base de datos XML se realizará mediante el API proporcionado por el propio gestor XML.

Para establecer la conexión la conexión con XML:DB, son necesarios los siguientes elementos:

Registrar el driver proporcionado por el proveedor del producto. El driver viene expresado de la siguiente forma:

• **driver="org.eXist.xmlldb.DatabaseImpl"**

Cargar el driver en memoria.

• **Class c1 = Class.forName(driver);**

Creamos una instancia de la base de datos

• **Database database = (Database) c1.newInstance();**

Registramos la instancia.

• **DatabaseManager.registerDatabase(database);**

Ahora ya podemos acceder a la colección que queramos. La ruta de la conexión se especifica mediante unaURI, (ruta de acceso al servicio) y la ruta de la colección.

• Para una BD local sería así: **URI="xmlldb:exist://localhost:8080/eXist/xmlrpc";**

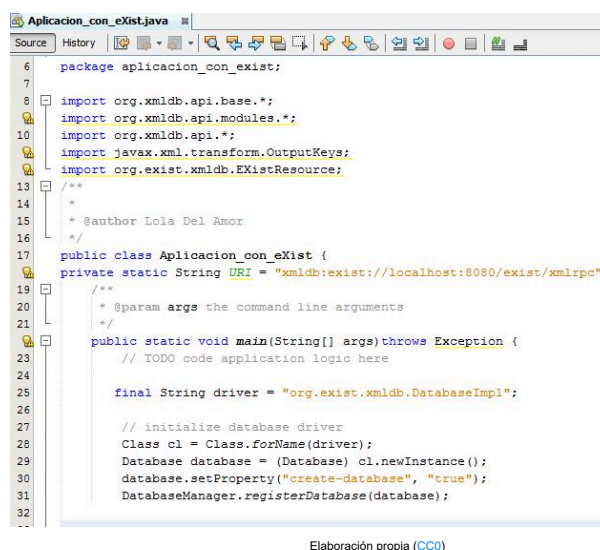
• Para acceder a la colección (por ejemplo a la colección Aulas) **collection= "/db/Cursos/Aulas";**

• Si pertenece a un usuario hay que indicar como parámetros usuario y contraseña:

• **usuario="admin"; pwd="admin";**

• Quedando la conexión como: **Collection col = DatabaseManager.getCollection(URI, usuario, pwd);**

En el siguiente ejemplo puedes ver el código de la conexión.



```

6 package aplicacion_con_exist;
7
8 import org.xmlldb.api.base.*;
9 import org.xmlldb.api.modules.*;
10 import org.xmlldb.api.*;
11 import javax.xml.transform.OutputKeys;
12 import org.exist.xmlldb.EXistResource;
13
14 /**
15  * @author Lola Del Amor
16  */
17 public class Aplicacion_con_eXist {
18     private static String URI = "xmlldb:exist://localhost:8080/exist/xmlrpc";
19
20     /**
21      * @param args the command line arguments
22      */
23     public static void main(String[] args) throws Exception {
24         // TODO code application logic here
25
26         final String driver = "org.exist.xmlldb.DatabaseImpl";
27
28         // initialize database driver
29         Class c1 = Class.forName(driver);
30         Database database = (Database) c1.newInstance();
31         database.setProperty("create-database", "true");
32         DatabaseManager.registerDatabase(database);
33     }
34 }

```

Elaboración propia (CC0)

5.2.- Operaciones sobre colecciones

La API XML:DB nos permite también crear y eliminar colecciones y documentos.

Para crear una nueva colección, hay que llamar al método `createCollection` del servicio `CollectionManagementService`.

En el siguiente código puedes ver como se **crea la colección** `NEW_COLLECTION` dentro de la colección `col`:

```
CollectionManagementService colserv = (CollectionManagementService)
    col.getService("CollectionManagementService", "1.0");
colserv.createCollection("NEW_COLLECTION");
```



Del mismo modo para borrar una colección:

```
CollectionManagementService colserv = (CollectionManagementService)
    col.getService("CollectionManagementService", "1.0");
colserv.removeCollection("NEW_COLLECTION");
```

Citas para pensar

"El sabio puede sentarse en un hormiguero, pero sólo el necio se queda sentado en él."

Proverbio chino.

Autoevaluación

El servicio para crear y borrar colecciones se llama:

- ☐ `CollectionManagementService`.
- ☐ `CreatecollectionService`
- ☐ `DeleteCollectionService`
- ☐ No hay ningún servicio que haga esto

Exacto!!

No es cierto, deberías haber leído mejor.

No, ese servicio no existe.

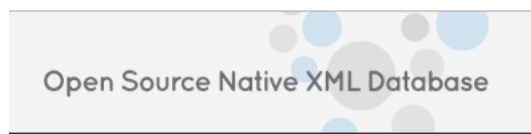
Si si que lo hay!!

Solución

1. Opción correcta
2. Incorrecto
3. Incorrecto
4. Incorrecto

5.3.-Operaciones sobre documentos.

De la misma forma que existe la posibilidad de crear y borrar colecciones existe también la posibilidad de crear y borrar documentos.



Elaboración propia (CC0)

Para ello se utilizan los siguientes métodos:

CreateResource: para crear nuevos documentos.

RemoveResource: para borrar documentos.

En el siguiente código podemos ver cómo crear un nuevo documento:

```
import java.io.File;
File fichero= new File("NUEVO_DOC.xml");//Crea el nuevo documento.
if (!fichero.canRead())
    System.out.println("Error al leer el fichero");
else{
    Resource nuevoRecurso=col.createResource(fichero.getName(), "XMLResource");//Creamos el recurso
    nuevoRecurso.setContent(fichero);//Asigno el fichero al recurso.
    col.storeResource(nuevoRecurso);//Lo asigno a la coleccion
}
```

De la misma forma vemos ahora como borrar un documento ya existe de una colección:

```
try
{
    Resource borrarRecurso=col.getResource("NUEVOS_CURSOS.xml");
    col.removeResource(borrarRecurso);
}
catch(NullPointerException e){
    System.out.println("No se puede borrar,no se encuentra");
}
```

Debes conocer

Consulta la documentación de la API de eXist en la página oficial:

[API-XML:DB](#)

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

El método para borrar documentos de una colección es **DeleteResource**

☐ Verdadero ☐ Falso

Falso

Es falso, ese no es el nombre del método, es **RemoveResource**

5.4.- Consultar documentos

¿Se utiliza el lenguaje XQuery para realizar consultas a la BD XML desde una aplicación Java? La respuesta es sí, pero será necesario realizar los siguientes **pasos para procesar una consulta desde Java**:

El método siguiente ejecuta una consulta que se encuentra en un fichero consulta.xq y almacenada en la carpeta del proyecto. El método va a recibir el nombre del fichero a ejecutar. ¿Cómo funciona?.

- Se lee el fichero
- Se recorre hasta el final y se almacena en una cadena
- Se ejecuta la consulta.

Internamente, eXist no distingue entre expresiones XPath y XQuery. XQueryService por lo tanto, se asigna a la misma clase de implementación que XPathQueryService. Sin embargo, proporciona algunos métodos adicionales. Lo más importante, cuando se habla con una base de datos incrustada, es que XQueryService permite que la expresión XQuery se compile en una representación interna, que luego se puede reutilizar. El código completo puede consultarse en el siguiente enlace: [código completo](#)

```
try {
    col = DatabaseManager.getCollection(URI + args[0]);
    XPathQueryService xpqs = (XPathQueryService)col.getService("XPathQue
ryService", "1.0");
    xpqs.setProperty("indent", "yes");
    ResourceSet result = xpqs.query(args[1]);
    ResourceIterator i = result.getIterator();
    Resource res = null;
    while(i.hasMoreResources()) {
        try {
            res = i.nextResource();
            System.out.println(res.getContent());
        } finally {
            //dont forget to cleanup resources
            try { ((EXistResource)res).freeResources(); } catch(XMLDBExc
eption xe) {xe.printStackTrace();}
        }
    }
} finally {
    //dont forget to cleanup
    if(col != null) {
        try { col.close(); } catch(XMLDBException xe) {xe.printStackTrac
e();}
    }
}
```

eXist-DB ([CC0](#))

[Código descargable](#) (java - 1.77 KB).

Autoevaluación

Señala si la siguiente afirmación es verdadera o falsa.

eXist no distingue internamente entre expresiones XPath y XQuery

☐ Verdadero ☐ Falso

Verdadero

Efectivamente, No hay distinción interna.

6.- Tratamiento de excepciones.

Caso práctico

Ada se ha vuelto a reunir con **María** y **Juan**. Les ha insistido en que, como siempre, no deben descuidar la robustez de la aplicación. Ya saben que los clientes son muy exigentes y que una aplicación que falla de manera abrupta y se bloquea, no es el sello de la empresa BK Programación.

Juan y **María**, al unísono le han contestado —Tranquila **Ada**, la aplicación va a capturar y procesar todas las posibles excepciones. No habrá error que se le resista.



Una excepción es un suceso excepcional que ocurre durante la ejecución de un programa y que interrumpe su ejecución o flujo normal de sentencias. El origen o causa de una excepción es un error en el programa.

Gestionando de manera adecuada las posibles excepciones del programa, podemos construir programas robustos, que pueden continuar su ejecución sin que el problema les afecte, o bien finalizar de manera no abrupta.

A lo largo del tema han aparecido algunas excepciones bien a nivel de métodos utilizando la palabra clave **throws** para luego ser tratadas en `main`, o bien utilizando bloques **try-catch** dentro de los métodos.

XMLDBEXCEPTION

XMLDBException se ejecuta cuando se produce una excepción en la **API XML:DB**. Contiene dos códigos de error:

- código de error XML de API
- código de error definido por el proveedor.

Cuando se produce un error con **XMLDBException** podemos capturar la información usando el método **getMessage()**, que devuelve la cadena del error.

En el siguiente enlace puedes ver algunos ejemplos sobre el tratamiento de las excepciones:

[Excepciones XMLDBException](#)

6.1.- Excepciones en el procesamiento de consultas.

Cuando se procesa una consulta XQuery o una consulta de actualización XQuery, pueden ocurrir errores en el proceso de compilación y evaluación de la expresión de consulta, que suponen el lanzamiento de excepciones.

CompileException

EvaluateException












Para saber más

En el siguiente enlace puede ver un ejemplo de tratamiento de excepciones con Qizx en un entorno Web, aunque estemos trabajando con eXist es interesante conocer de su existencia.

[Ejemplo de tratamiento de excepciones con Qizx en entorno Web.](#)

Anexo.- Licencias de recursos.

Licencias de recursos utilizados en la Unidad de Trabajo.

Recurso (1)	Datos del recurso (1)	Recurso (2)	Datos del recurso (2)
	Autoría: Emma Gracia Lor. Licencia: CC-by-nc-sa. Procedencia: http://www.flickr.com/photos/paideiaeducacion/5102909303/		Autoría: fsse8info. Licencia: CC BY-SA. Procedencia: http://www.flickr.com/photos/fsse-info/4194980532/
	Autoría: Howard Stanbury. Licencia: CC BY-NC-SA. Procedencia: http://www.flickr.com/photos/stanbury/5852623652/		Autoría: Justin See. Licencia: CC BY. Procedencia: http://www.flickr.com/photos/koalazymonkey/3651288422/
	Autoría: Adam Prince. Licencia: CC BY-NC-SA. Procedencia: http://www.flickr.com/photos/adam_prince/3908404628/		Autoría: Tantek Çelik. Licencia: CC BY-NC. Procedencia: http://www.flickr.com/photos/tantek/539497686/
	Nombre: AD05_CONT_R02_xml Título: Rótulo XML Autoría: Isabel M. Cruz Granados Licencia: Uso educativo-no comercial. Procedencia: Elaboración propia		Título: Documento de texto y datos Autoría: Isabel M. Cruz Granados Licencia: Uso educativo-no comercial. Procedencia: Elaboración propia Descripción: Dibujo que muestra los rótulos Datos (en azul) y Texto (en rojo). Bajo los rótulos una figura blanca de en forma de pergamino incluyendo un documento XML entrado en datos y centrado en texto respectivamente.
	Título: Almacenar XML Autoría: Isabel M. Cruz Granados Licencia: Uso educativo-no comercial. Procedencia: Elaboración propia Descripción: Dibujo que representa las posibilidades de almacenamiento de documentos XML (En fichero, en Base de datos existente como BLOB o mapeado, y en base de datos nativa)		Título: Jerarquía BD nativa XML Autoría: Isabel M. Cruz Granados Licencia: Uso educativo-no comercial. Procedencia: Elaboración propia Descripción: Dibujo que muestra la jerarquía de una BB nativa XML, dibujando el nodo raíz en la parte superior y después el árbol de colecciones, en rojo, y documentos, en celeste.
	Título: Árbol XML Autoría: Isabel M. Cruz Granados Licencia: Uso educativo-no comercial. Procedencia: Elaboración propia Descripción: Dibujo que muestra a la izquierda un documento XML y a la derecha su transformación en árbol con nodos.		Título: Camino. Autoría: ITE. Juan F. Morillo. Licencia: Uso educativo-nc. Procedencia: idITE=182864 Descripción: Foto de un camino de tierra entre matorrales.
	Título: Excepciones de consultas Autoría: Isabel M. Cruz Granados Licencia: Uso educativo-no comercial. Procedencia: Elaboración propia Descripción: Dibujo que muestra un cartel blanco con las palabras: CompileException en verde oliva, y EvaluateException en granate.		Título: Foto de Ada. Autor: Ministerio de Educación. Licencia: Uso Educativo no comercial. Procedencia: Elaboración propia. Descripción: Foto de Ada mirando al frente.

Condiciones y términos de uso de los materiales

Materiales desarrollados inicialmente por el Ministerio de Educación, Cultura y Deporte y actualizados por el profesorado de la Junta de Andalucía bajo licencia Creative Commons BY-NC-SA.



Antes de cualquier uso leer detenidamente el siguiente [Aviso legal](#)

Historial de actualizaciones

Versión: 02.00.00	Fecha de actualización: 19/06/19	Autoría: María Dolores Amor Gómez
Ubicación: Toda la unidad Mejora (tipo 3): La herramienta Qizx Studio ha sido comprada por Qualcomm y ha dejado de ser libre y gratuita. Se recomienda sustituir esta herramienta en la que está basada toda la unidad por la herramienta eXist Ubicación: A lo largo de unidad Mejora (Mapa conceptual): Se ha modificado el mapa conceptual de acuerdo a las modificaciones y actualizaciones de la unidad. Ubicación: A lo largo de la unidad Mejora (Orientaciones del alumnado): Apartados que han sido cambiados en la unidad con respecto a la sugerencia		
Versión: 01.00.00	Fecha de actualización: 11/11/13	
Versión inicial de los materiales.		

