Introducción al acceso a datos.

# 1.- Introducción.

La inmensa mayoría de aplicaciones informáticas se pueden diferenciar, a grandes rasgos, en dos partes:

* Por un lado, **el programa** propiamente dicho, que realiza las operaciones deseadas con los datos necesarios.
* Por otro lado, **los datos** con los que opera le programa. Esos datos pueden ser obtenidos por el programa mediante diversos métodos: leídos mediante teclado, escaneados, leídos de algún soporte de almacenamiento secundario, etc.

En la mayoría de los casos, cuando programamos, nos interesa que el programa guarde los datos que le hemos introducido, o los resultados que dicho programa haya obtenido, es decir, **persistan**. Una forma tradicional de hacer esto es mediante la utilización de ficheros o de bases de datos que se guardarán en un dispositivo de memoria no volátil (normalmente un disco).

En esta unidad inicial, vas a ver una panorámica de los diversos métodos de [**persistencia**](file:///D:\OneDrive\OneDrive%20-%20Consejer%C3%ADa%20de%20Educaci%C3%B3n%20y%20Cultura\Documentos\FP\DAM\Curso%202\Materias\Acceso%20a%20datos\Evaluacion%201\Tema%201\Apuntes\AD01\web\html\1_introduccin.html#t8fdd7714-1c9a-6a36-9e5a-de9e9f438d35)que encontramos en el mercado.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Para saber más**: en este módulo y en el ciclo en general, nos decantamos por Java. Otra alternativa a Java es .NET. [.NET (desarrolloweb.com)](https://desarrolloweb.com/manuales/48)

# 2.- Acceso a datos.

Hay diversas estrategias de acceso a datos para gestionar la persistencia de los datos:

* Mediante **ficheros** (la forma más primitiva).
* **Bases de datos**, que pueden ser:
  + Relacionales (la forma más estandarizada),
  + Orientadas a objetos (relativamente nueva),
  + Objeto-relacionales (una extensión de las relacionales, es la opción preferida cundo queremos usar los conceptos de orientación a objetos).
* **Mapeo objeto relacional**(ORM).
* Bases de datos XML(eXtensible Markup Language).
* **Componentes**.

**Las bases de datos solucionaban los problemas** de guardar la información mediante ficheros:

* **Eliminan** la posibilidad de información **redundante**.
* **Eliminan** la información **inconsistente**.
* Globalizan y **centralizan la información**.
* Garantizan el mantenimiento de la **integridad en la información**. Únicamente se almacena la información correcta.
* **Independencia de datos**. La independencia de datos implica una separación entre programas y datos

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## 2.1.- Qué estrategia o método de acceso de datos usar.

No se puede afirmar que haya un método que sea el mejor de manera absoluta. Más bien, **la cuestión es tener claro qué tipo de aplicación hay que construir y, según eso, estudiar qué tipo de sistema de almacenamiento será mejor usar**: si una base de datos orientada a objetos, o una base de datos XML, etc.

Debemos **conocer el funcionamiento de las diferentes alternativas para poder comparar sus prestaciones**. **Cada una de las tecnologías tiene su propio origen y filosofía** para alcanzar el mismo fin y, por esta razón, no es fácil analizar sus ventajas y desventajas frente a las demás alternativas.

Por ejemplo: si voy a crear una base de datos para guardar mi colección de vídeos, probablemente no me va a interesar utilizar una base de datos Oracle, sino un producto mucho más barato, y sencillo de instalar y mantener.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# 3.- Ficheros.

Ejemplo de información contenida en un fichero:

**Antonio Pérez Pérez 30 C/ Morales nº 11 Madrid Madrid**

**Feliciano Gómez Sander 25 C/ Terreros nº 121 Vitoria Vitoria**

**Arturo Bueno Hernández 46 C/ Cocoliso nº 43 Murcia Murcia**

**Desventaja**: el programador tenía que conocer detalladamente las posiciones de los datos, para saber desde qué posición hasta qué otra posición, se guardaba el nombre y apellidos, etc.

Aún así, aunque las BB.DD supusieron un gran avance en este sentido, los ficheros siguen siendo útiles. Por ejemplo: un fichero de configuración, o un fichero log. Es decir, no siempre nos hace falta una base de datos para almacenar la información.

En Java, como en otros lenguajes de programación, hay diversas clases para el manejo de ficheros.

## 3.1.- Uso de ficheros en la actualidad.

La información guardada en ficheros, ya no sigue una estructura como la del ejemplo anterior, sino que actualmente, siguen un patrón y su estructura está bien definida.

Aquí los más importantes son los ficheros XML. No necesitan software propietario para ser interpretado, y tienen extensión xml.

Muchas veces se recurre al uso de ficheros XML para intercambiar información a través de varias plataformas de hardware o de software, o de varias aplicaciones. A veces se exporta de una base de datos a ficheros XML para trasladar la información a otra BB.DD que leerá esos XML.

**Por esta razón se emplea XML en tecnologías de comunicación** como, por ejemplo, en [**WML**](https://ead.murciaeduca.es/pluginfile.php/1431817/mod_resource/content/3/AD01_Contenidos_Web/31_uso_ficheros_en_la_actualidad.html#t684dd398-5fd8-ee5b-2066-a9f87351df03) (lenguaje de formato inalámbrico) y [**WAP**](https://ead.murciaeduca.es/pluginfile.php/1431817/mod_resource/content/3/AD01_Contenidos_Web/31_uso_ficheros_en_la_actualidad.html#t226016d4-1d17-bbfa-9aab-c539a776b412)(protocolo de aplicaciones inalámbricas).

Hay muchas librerías de conversión de la información almacenada en XML a otros formatos, como a PDF, texto, hojas de cálculo, etc. Hay muchos productos propietarios y también de código abierto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Para saber más**: [Tutorial XML](http://web.archive.org/web/20191204001517/http:/geneura.ugr.es/~jmerelo/xml/).

# 4.- Bases de datos.

Un sistema de bases de datos es un **sistema de información orientado hacia los datos**, que pretende recuperar y almacenar la información de manera eficiente y cómoda.

## 4.1.- Introducción:

Ventajas de las Bases de datos respecto de los archivos convencionales:

* Independencia de los datos respecto de los procedimientos.
* **Disminución de redundancias y** reducción de la posibilidad de **inconsistencia** de datos.
* **Mayor integridad de datos**: solo se almacenan los datos correctos, y una única vez.
* Mayor **disponibilidad** de los datos.
* Mayor **seguridad** de los datos.
* Mayor **eficiencia** en la recogida, codificación y entrada en el sistema.
* **Interfaz con el pasado y futuro**: la BB.DD siempre está abierta a reconocer información organizada físicamente por otro software.
* **Compartición de los datos**: los datos deben poder ser accedidos por varios usuarios simultáneamente sin comprometer la integridad de los mismos.

Podemos afirmar generalizando, que se usa un **sistema de ficheros convencional cuando la cantidad de datos a guardar es tan reducida**que no justifica las desventajas del uso de los sistemas de bases de datos.

Interfaz de usuario gráfica, Texto, Escala de tiempo

Descripción generada automáticamente con confianza media

## 4.2.- Bases de datos relacionales:

El **modelo de bases de datos relacional** fue propuesto en 1969 por Edgar Codd.

Utilizan un método declarativo para especificar datos y consultas.

**Ventajas**: adecuadas para manejar grandes cantidades de datos, compartir datos entre programas, realizar búsquedas rápidas, etc.

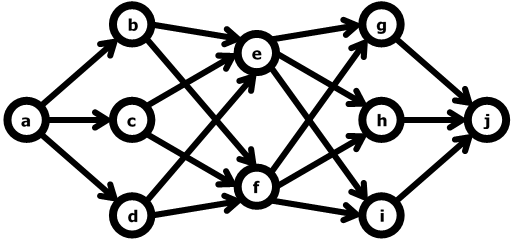
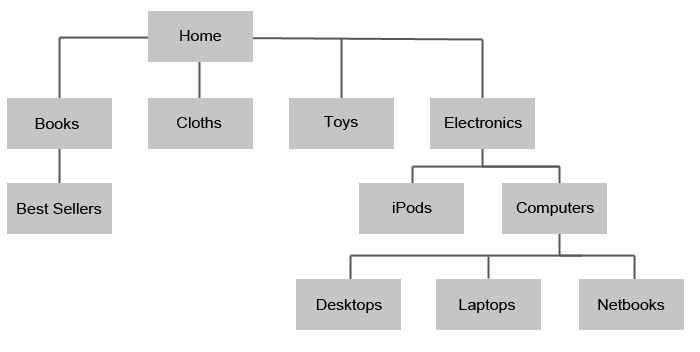
**Desventajas**: no presentan un buen modelo de las relaciones entre los datos, ya que todo se representa como tablas bidimensionales (ósea, filas y columnas).

Características de las bases de datos relacionales:

* Están muy **extendidas**.
* Son muy **robustas**.
* Permiten **interoperabilidad** (compartir datos) entre aplicaciones.
* Son el **común denominador de muchos sistemas** y tecnologías: Java, C++…

Otros modelos tradicionales

Los modelos **jerárquico** y **en red**. Algunos sistemas antiguos utilizan todavía estas arquitecturas, en centros de datos donde se manejan grandes volúmenes de información y la complejidad de los sistemas hace prohibitivo el coste que supondría migrar a sistemas que utilizaran otro modelo como el relacional.



El modelo relacional fue el primer modelo de bases de datos definido de manera formal. Posteriormente como modelos informales, los modelos jerárquico y en red. Sin embargo, ya existían este tipo de bases de datos antes que las bases de datos relacionales, **pero fueron descritas como modelo después**.

**Para saber más**: [Tipos de Bases de Datos: Modelos, Usos y Beneficios (tecnologias-informacion.com)](https://www.tecnologias-informacion.com/basesdedatos.html)

## 4.3.- Bases de datos orientadas a objetos (BDOO).

¿Por qué surgen este tipo de bases de datos?

* **La existencia de problemas al representar cierta información y modelar ciertos aspectos del mundo real**. Los modelos clásicos (relacional) permiten representar gran cantidad de datos, pero las operaciones que se pueden realizar sobre ellos son bastante simples.
* **Para que pasar del modelo de objetos al modelo relacional no sea un obstáculo**. Esto genera dificultades en BDR que, en el caso de las BDOO, no surgen, ya que el modelo es el mismo. Los datos de los programas escritos en lenguaje orientado a objetos se pueden almacenar directamente, sin conversión alguna, en las BDOO.

Los sistemas de bases de datos orientadas a objetos soportan un **modelo de objetos puro**, a diferencia del modelo objeto-relacional, que se basa en una extensión del modelo relacional.

Por ello, una característica general es que **el lenguaje de programación y el esquema de la base de datos utilizan las mismas definiciones de tipos**.

**Para saber más**: qué debe cumplir una BDOO:

|  |  |
| --- | --- |
| **Características obligatorias** | 1. Deben soportarse **objetos complejos**. 2. Deben soportarse **mecanismos de identidad de los objetos**. 3. Debe soportarse la **encapsulación**. 4. Deben soportarse **tipos y clases**. 5. Los tipos o clases deben ser capaces de **heredar** de sus ancestros. 6. Debe **soportarse** **sobrecarga**, **sobreescritura** y **enlace dinámico**. 7. El DML debe ser **computacionalmente completo**. 8. El conjunto de todos los tipos de datos debe ser **ampliable**. 9. Debe proporcionarse **persistencia** a los datos. 10. El SGBD debe ser capaz de gestionar bases de datos de muy gran tamaño. 11. El SGBD debe soportar a usuarios concurrentes. 12. El SGBD debe ser capaz de recuperarse de fallos hardware y software. 13. El SGBD debe proporcionar una forma simple de consultar los datos. |
| **Características opcionales** | 1. Herencia múltiple. 2. Comprobación de tipos e inferencia de tipos. 3. Sistema de bases de datos distribuido. 4. Soporte de versiones. |

El acceso a los datos **es más rápido con las bases de datos orientadas a objetos que con las bases de datos tradicionales**, porque no hay necesidad de utilizar JOINS, ya que basta con recuperar un objeto simplemente, siguiendo punteros.

Cada vez más, las **necesidades de las aplicaciones actuales con respecto a las bases de datos son**:

* Soporte para **objetos complejos** y **datos multimedia**.
* Jerarquías de objetos o **tipos** y **herencia**.
* **Gestión de versiones**.
* Modelos extensibles mediante **tipos de datos definidos por el usuario**.
* **Integración de los datos con sus procedimientos** **asociados**.
* **Manipulación navegacional** (en vez de declarativa) y de conjunto de registros.
* Interconexión e interoperabilidad.

**Ejemplos de Sistemas Gestores de Bases de Datos Orientados a Objetos**:

* [Base de datos Jasmine (cai.com)](http://www.cai.com/)
* [Base de datos ObjectStore (odi.com)](http://www.odi.com/)
* [Base de datos Gemstone (wordpress.com)](https://smalltalkuy.wordpress.com/2009/10/19/gemstones-el-oracle-de-las-oodbms/)

**Para saber más**:

[Interoperabilidad - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Interoperabilidad)

[Control de versiones - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Control_de_versiones)

Tabla

Descripción generada automáticamente

## 4.4.- Comparativa entre bases de datos relacionales y orientadas a objetos.

Los sistemas de bases de datos orientados a objetos (ODBMS) han demostrado ser superiores para cierto tipo de tareas, ya que muchas operaciones se realizan utilizando interfaces navegacionales más que declarativos, y el acceso a datos navegacional es normalmente implementado muy eficientemente.

Ventajas de las BBDDOO:

* La **transparencia** (no ensucia la construcción de una aplicación). El desarrollador así se debe preocupar de los objetos de su aplicación, en lugar de como los debe almacenar y recuperar de un medio físico.
* Permiten mayor **capacidad de modelado**: ya que permite modelar el mundo real de una forma mucho más fiel. Esto es porque:
  + Un objeto permite **encapsular tanto un estado, como un comportamiento**.
  + Un objeto puede **almacenar todas las relaciones** que tenga con otros objetos.
  + Los objetos pueden **agruparse para formar objetos complejos** (herencia).
* **Extensibilidad**, debido a que:
  + Se pueden **crear nuevos tipos de datos** a partir de los ya existentes.
  + Agrupación de propiedades comunes en superclases (herencia), lo que reduce la redundancia.
  + **Reusabilidad de clases**, lo que repercute en una mayor facilidad de mantenimiento y un menor tiempo de desarrollo.
* Desarrollo de un **lenguaje de consulta más expresivo**: gracias al **acceso navegacional** desde un objeto al siguiente (mediante punteros).
* **Adaptación a aplicaciones** avanzadas de base de datos. Por ejemplo, en sistemas de diseño CAD (Diseño asistido por ordenador), CASE (Conjunto de herramientas de diseño por ordenador), sistemas multimedia, etc. Donde aquí los SGBDR no han tenido éxito.
* **Prestaciones**: los SGBDOO proporcionan **mejoras significativas de rendimiento** con respecto a los SGBDR. Sin embargo, los SGBDR siguen teniendo mejor rendimiento en el procesamiento de transacciones en línea.
* **Reglas de acceso**: en las bases de datos relacionales, los atributos se manipulan mediante operadores relacionales predefinidos. En cambio, en las orientadas a objetos, se procede mediante las interfaces que se crean a tal efecto de las clases.
* **Clave**: los objetos no necesitan una representación visible del identificador (lo que sería la clave primaria en el modelo relacional).

**Para saber más**: [Microsoft Word - bdOO.doc (iessanvicente.com)](https://iessanvicente.com/colaboraciones/bdOO.pdf)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Para saber más**: [curso introductorio sobre SQL](http://web.archive.org/web/20210514103802/https:/www.aulaclic.es/sql/).

Desventajas de las bases de datos orientadas a objetos frente a las relacionales

* **Reticencia del mercado**, tanto para desarrolladores como usuarios, a este tipo de bases de datos.
* **Carencia de un modelo de datos universal**: la mayoría de modelos (que además, ninguno está universalmente aceptado) carecen de una base teórica. El modelo de objetos aún no tiene una teoría matemática coherente que le sirva de base.
* **Carencia de experiencia**: al ser una tecnología relativamente nueva, no se dispone de experiencia.
* **Panorama actual**: en su lugar, tenemos los sistemas de bases de datos objeto-relacionales, los cuales están muchísimo más extendidos. SQL es un estándar aprobado y ODBC y JDBC son estándares de facto (como conectores/drivers de bases de datos).
* **Dificultadesen optimización**: la optimización de consultas necesita realizar una compresión de la implementación (procedimientos) de los objetos para poder acceder a la base de datos de manera eficiente. Sin embargo, esto compromete el concepto de encapsulación.

## 4.5.- Bases de datos objeto-relacionales.

Es un tipo de Base de datos que ha evolucionado desde el modelo relacional a otro extendido que incorpora conceptos del paradigma orientado a objetos. Tiene lo mejor de los dos mundos: relacional y orientación a objetos.

En una Base de Datos Objeto Relacional el diseñador puede **crear sus propios tipos de datos** y **crear métodos** para esos tipos de datos.

Un ejemplo de creación de un tipo de datos:

CREATE TYPE persona\_t AS OBJECT (

nif VARCHAR2(9),

nombre VARCHAR2(30),

direccion VARCHAR2(40),

telefono VARCHAR2(15),

fecha\_nac DATE);

En la base de datos Oracle, a partir de la versión 8i, fue extendido con conceptos del modelo de bases de datos orientadas a objetos.

Se reconoce el concepto de objetos, de tal manera que **un objeto tiene un tipo**, se almacena en cierta fila de cierta tabla **y tiene un identificador único (OID)**. **Estos identificadores se pueden utilizar para referenciar a otros objetos y así representar relaciones de asociación y agregación**.

La ventaja es que las aplicaciones construidas sobre bases de datos relacionales, no es necesario volver a reescribirlas. Más tarde, se pueden ir adaptando las aplicaciones y bases de datos para que utilicen las funciones orientadas a objetos.

**Aportaciones que hace el modelo orientado a objetos a las bases de datos objeto-relacionales**:

* Mayor variedad de tipos, gracias a poder crear los nuestros propios.
  + **Además, se soportan tipos complejos como**: registros, conjuntos, referencias, listas, pilas, colas y vectores.
* Se pueden **compartir** varias **librerías de clases** ya existentes (**reusabilidad**).

**Como productos comerciales de bases de Datos Objeto-Relacional, podemos destacar**:

* DB2 Universal Database de IBM (International Business Machines).
* Universal Server de Informix (ahora de IBM),
* INGRES II, de Computer Associates.
* ORACLE de Oracle Corporation,
* SyBASE

Como productos de código abierto, destacamos PostGreSQL.

**Para saber más**: [Bases de datos Objeto-relacional (wikipedia.org)](https://es.wikipedia.org/wiki/Base_de_datos_objeto-relacional)

# 5.- Acceso a bases de datos mediante conectores.

Para gestionar la información de las bases de datos hay unos estándares que facilitan a los programas informáticos manipular la información almacenada en ellas.

En Java existe una **API** basada en estos estándares que **permite conectarse a bases de datos relacionales**: **JDBC** (Java Database Connectivity, conectividad de bases de datos de Java).

La mayoría de las aplicaciones importantes de una empresa están respaldadas por una arquitectura normalizada y optimizada de **bases de datos relacionales** (basadas en sentencias SQL).

**Este modelo** continúa teniendo una gran importancia estratégica y **es la base para el continuo crecimiento del mapeo Objeto-Relacional (O/R)** y está asociado a los mecanismos de persistencia.

El API **JDBC** **define interfaces y clases** para conectarse a bases de datos desde una aplicación.

Mediante JDBC se pueden enviar sentencias SQL y PL/SQL a una base de datos relacional. JDBC permite embeber SQL dentro de código Java.

Diagrama

Descripción generada automáticamente con confianza baja

La ilustración muestra una representación de los diferentes mecanismos de mapeo O/R y cómo se relacionan con el código de la aplicación. Se observa claramente la función crítica que desempeña el driver JDBC, ya que es la base de todos los frameworks.

**Ventaja de usar conectores JDBC**: es independiente de la base de datos que utilices.

No obstante hay un trabajo de traducción para mapear los campos devueltos por cada consulta a la colección de objetos correspondiente. Esto constituye una razón para tratar de buscar alternativas menos costosas en tiempo de desarrollo: los frameworks.

**Más información**:

ODBC (Open Database Connectivity) es la interfaz de programación de bases de datos que utiliza Microsoft para acceder a distintas bases de datos relacionales en diversas plataformas.

También existe un estándar que sirve de puente entre JDBC-ODBC en las versiones Solaris y Windows de la plataforma Java, para que se pueda utilizar ODBC desde un programa Java.

# 6.- Mapeo objeto relacional (ORM).

Cuando se programan sistemas orientados a objetos, utilizando una base de datos relacional, se han de convertir los objetos del lenguaje de programación a registros de la base de datos. De igual forma, para hacer la operación inversa, se necesita convertir esos registros de la BB.DD a objetos del lenguaje.

El **mapeo objeto-relacional** (Object-Relational Mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el sistema utilizado en una base de datos relacional.

**El modelo relacional** trata con relaciones y conjuntos, **es de naturaleza matemática**. Por el contrario, **el paradigma orientado a objetos trata con objetos**, atributos y asociaciones de unos con otros.

Cuando se requiere almacenar la información de los objetos utilizando una base de datos relacional, se produce lo que se conoce como **desfase objeto-relacional** (problema de compatibilidad entre ambos modelos).

Por ello, **para ahorrar trabajo al programador, se puede utilizar un [framework](https://ead.murciaeduca.es/pluginfile.php/1431817/mod_resource/content/3/AD01_Contenidos_Web/6_mapeo_objeto_relacional_orm.html" \l "t38643fbc-ccb9-5562-c7d6-af44415d474b" \o ") que se encargue de realizar estas tareas de modo transparente**, de modo que el programador no tenga por qué usar JDBC ni SQL.

Tabla

Descripción generada automáticamente

## 6.1.- Capa de persistencia y framework de mapeo.

La **capa de persistencia** de una aplicación es la **pieza que permite almacenar, recuperar, actualizar y eliminar el estado de los objetos** que necesitan persistir en un sistema gestor de datos.

El **framework de mapeo ORM** es una capa que permite **relacionar objetos con un modelo relacional**, ocultando todo el mecanismo de conexión al **motor de base de datos,** así como las sentencias SQL.

La **capa de persistencia** traduce entre los dos modelos de datos: desde objetos a registros, y desde registros a objetos. Para grabar un objeto, el programa llama al **motor de persistencia** para que traduzca el objeto a registro y lo guarde en la base de datos.

De este modo, el programa solo ve que puede guardar y recuperar objetos, como si estuviese programado para una base de datos orientada a objetos.

Hay varias alternativas de mapeadores O/R:

|  |  |
| --- | --- |
| **De organizaciones basadas en el estándar** | * EJB 3.0 (Enterprise JavaBean) |
| **De código abierto** | * Hibernate * Spring |
| **Implementaciones comerciales** | * TopLink |

Cada uno de los mecanismos de mapeo O/R tiene una dependencia particular en el conector JDBC para poder comunicarse con la base de datos de una forma eficiente. Por ello **es importante elegir el conector JDBC que participa en la comunicación óptimo, el que mejor se adapte a la aplicación**.

**Debes conocer**: [Persistencia (informática) - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Persistencia_%28inform%C3%A1tica%29)

Tabla

Descripción generada automáticamente con confianza media

# 7.- Bases de datos XML.

Estas bases de datos almacenan los datos estructurados como XML sin la necesidad de traducir los datos a una estructura relacional o de objeto.

Podemos distinguir entre:

* **Bases de datos nativas XML**: consiste en un modelo lógico para documentos XML.
  + **Ejemplos de sistemas gestores de datos**: eXist, Apache Xindice, Berkeley dbXML, **BaseX**…

Texto

Descripción generada automáticamente

* + Permiten trabajar con **XQL** (eXtensible Query Language), el cuál sirve un propósito similar a SQL en una base de datos relacional.
  + Las bases de datos nativas son consultadas mediante expresiones XPath o XQuery.
  + **Involucra dos pasos básicos**:
    - **Describir** los datos mediante **Definiciones de Tipos de Datos** (Document Type Definitions, **DTD**) o esquemas XML y
    - **Definir un nuevo esquema de base de datos XML** nativa o Mapa de Datos a usar para almacenar y obtener datos.
* **Bases de datos compatibles con XML (XML-enabled)**: son bases de datos, generalmente objeto-relacionales, que admiten entradas de datos en forma de XML, y proporcionan salidas en este mismo formato.
  + **Ejemplos de sistemas gestores de datos**: Oracle, DB2 (Viper), SQL Server…
  + Texto

    Descripción generada automáticamente

**Para saber más**: [Tutorial de XML (archive.org)](http://web.archive.org/web/20191204001517/http:/geneura.ugr.es/~jmerelo/xml/)

Tabla

Descripción generada automáticamente

# 8.- Desarrollo de componentes.

**Para saber más**: [Curso gratuito - Programacion Orientada a Objetos - Ciberaula (archive.org)](http://web.archive.org/web/20190921131347/http:/java.ciberaula.com/articulo/tecnologia_orientada_objetos)

## 8.1.- Definición de componente.

Un componente es una **unidad de software que realiza una función bien definida y posee una interfaz bien definida**. Además, puede interactuar con otros componentes.

Un **interfaz** es un **punto de acceso a los componentes**: permite a los clientes acceder a los servicios proporcionados por un componente.

La idea de **dividir u organizar en trozos el software**, o sea, dividirlo en componentes surge para **reducir la complejidad del software**. Así se permite la **reutilización** y **se acelera** el proceso de **ensamblaje del software** y de su **mantenimiento**.

La **interoperabilidad entre componentes** de distintos fabricantes:

* Aumenta la competencia.
* Reduce los costes.
* Facilita la construcción de estándares.

Un componente software también debe especificar sus necesidades para funcionar correctamente, lo que se denominan las dependencias de contexto:

* Interfaces requeridas
* Entorno de ejecución: máquina necesaria, sistema operativo a utilizar, etc.

En el entorno Java, contamos con los JavaBeans.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## 8.2.- JavaBeans.

El **origen** de los JavaBeans lo podemos encontrar en un par de necesidades que Java tenía:

* **Disponer de una tecnología de objetos y componentes reutilizables**.
* **Mejorar el proceso para crear interfaces de usuario**, acercándose a la facilidad de uso que otros productos permitían como Visual Basic de Microsoft.

Un **JavaBean** es un **componente software reutilizable** basado en la especificación JavaBean de Sun (ahora Oracle) que **se puede manipular visualmente con una herramienta de desarrollo**.

**Propiedades de los JavaBeans**:

* **Portabilidad**: que los beans puedan usarse en otras plataformas y entornos.
* **Reusabilidad**: que se puedan usar como bloques en la construcción de aplicaciones complejas.
* **Introspección**: los IDE reconocen ciertas pautas de diseño, nombres de las funciones miembros o métodos y definiciones de las clases.
* **Personalización**: en tiempo de diseño, con el IDE que se utilice, se pueden modificar las características de apariencia y comportamiento de un bean.
* **Persistencia**: un bean puede guardar su estado y recuperarlo posteriormente. Esta capacidad se logra mediante la **serialización**. Para recuperarlo, se logra mediante la deserialización.
* **Comunicación entre eventos**: los eventos son un **mecanismo de notificación entre objeto fuente y objeto(s) receptor(es)**. Las herramientas de desarrollo pueden examinar un bean para determinar qué eventos puede enviar y cuáles puede recibir.

**Reglas básicas para diseñar un JavaBean:**

* Se define como clase pública.
* Debe tener un constructor vacío por defecto, aunque puede tener más constructores.
* Se debe implementar el interface Serializable para poder implementar la persistencia.
* Se deben seguir las convenciones de nombres establecidas, como por ejemplo los metodos get...() y set...() se usan para obtener y establecer propiedades de un Bean.

Diagrama

Descripción generada automáticamente con confianza media

* Una propiedad es de solo lectura o solo escritura si falta uno de los métodos de acceso anteriores (get o set).

**Para saber más**: [Componentes Java Beans (uchile.cl)](https://users.dcc.uchile.cl/~lmateu/CC60H/Trabajos/fpicero/Index.htm)

Una captura de pantalla de un celular

Descripción generada automáticamente

# 9.- Herramientas de trabajo necesarias para el módulo.

* Eclipse.
* NetBeans
* Plugin de Hibernate
* JDBC o conectores Java para bases de datos.
* **Sistemas gestores de bases de datos relacionales (SGBDR)**: Oracle, MySQL y SQLite entre otros.
* **Sistemas gestores de bases de datos orientadas a documentos** como: Exists o Mongo.

## 9.1.- NetBeans.

[Descargar NetBeans 8.2](https://netbeans.org/downloads/8.2/rc/)

[Más sobre NetBeans](http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/1-introduccioi81n-a-netbeans.pdf)

Los pasos a seguir para instalar NetBeans 8.2 son los que se muestran a continuación:

* Descarga desde la web oficial de Oracle, Java SE e instálalo en tu equipo. Sigue los pasos que te indica dando la opción por defecto. Este producto lo puedes descargar en el siguiente enlace https://netbeans.org/community/releases/82/index.html
* Aunque no es necesario, si hubiera problemas configura las variables PATH (Equipo/propiedades/configuración avanzada del sistema y busca la variable path e introduce aquí ;C:\Program Files\Java\jdk1.7.0\bin;) y CLASSPATH (haz lo mismo que antes e introduce ;C:\Program Files\Java\jdk1.7.0\lib;).
* Aunque tampoco es necesario, si quieres, comprueba a través de la consola de comandos, que has realizado correctamente la instalación y configuración del JDK y JRE.
* Arranca NetBeans desde el icono instalado en el escritorio y crea un nuevo proyecto. Por defecto los proyectos se guardan en Documentos

## 9.2.- Eclipse.

**Características**:

* **Perspectivas, editores y vistas**: en Eclipse el concepto de trabajo está basado en las perspectivas, que no es otra cosa que una preconfiguración de ventanas y editores, relacionadas entre sí, y que nos permiten trabajar en un determinado entorno de trabajo de forma óptima.
* Se basa en proyectos.
* Depurador de código.
* Extensa colección de plug-ins.

<https://www.genbeta.com/desarrollo/eclipse-ide>

## 9.3.- Plugin de Hibernate.

Hibernate Tools es un conjunto de herramientas para Hibernate implementado como un paquete integrado de complementos de Eclipse, junto con una tarea Ant unificada para su integración en el ciclo de compilación.

**Características integradas de Hibernate en Eclipse**:

* **Editor de mapeo**:
  + Trae un editor para los archivos de mapeo XML de Hibernate
  + Es compatible con autocompletado y resaltado de sintaxis para nombres de clases, nombres de propiedades / campos, nombres de tablas y columnas.
* **Consola**: la consola de Hibernate permite configurar conexiones de bases de datos, proporciona visualización de clases y sus relaciones y permite ejecutar consultas HQL interactivamente contra su base de datos y examinar los resultados de la consulta.
* **Ingeniería inversa**: característica más poderosa de Hibernate Tools. Permite generar clases de modelo de dominio y archivos de mapeo Hibernate, Beans de entidad EJB3 anotados, documentación HTML o incluso una aplicación JBoss Seam completa en segundos.
* **Wizards**: Se proporcionan varios asistentes, incluidos los asistentes para generar rápidamente archivos de configuración de Hibernate (cfg.xml) y las configuraciones de la consola de Hibernate.
* **Tarea Ant**: Las herramientas de Hibernate incluyen una tarea Ant unificada que le permite ejecutar generación de esquema, generación de mapeo o generación de código Java como parte de su compilación.

**Para saber más**: [como instalar Hibernate en NetBeans](https://becommerce.es/web/java-anadir-soporte-hibernate-para-acceso-a-base-de-datos/).

## 9.4.- JDBC.

**Java Database Connectivity (JDBC)**es una interfaz de programación de aplicaciones (**API**) para Java, que **define** **como** un cliente puede **acceder a una base de datos**.

Es **parte de la plataforma Java Standard Edition**, de Oracle Corporation. Proporciona **métodos para consultar y actualizar datos** en una base de datos y está **orientado a bases de datos relacionales**.

Para conectarse con distintos gestores de bases de datos (MySQL, Oracle, SQLite, etc.), **JDBC requiere controladores para cada base de datos**.

El controlador o **driver JDBC proporciona la conexión a la base de datos e implementa el protocolo para transferir la consulta y el resultado** entre el cliente y la base de datos.

## 9.5.- Sistemas Gestores de Bases de Datos Relacionales.

Un gestor de base de datos (DataBase Management System) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarios para el almacenamiento y búsqueda de la información del modo más eficiente posible.

|  |  |
| --- | --- |
| **Oracle** | Tradicionalmente, Oracle ha sido el SGBS por excelencia, **considerado siempre como el más completo y robusto**. |
| **Ventajas**:   * Soporte de transacciones. * Escalabilidad. * Estabilidad. * Multiplataforma. * Existe una versión express gratis para pequeñas instalaciones y usuarios personales. |
| **Instalación de Oracle**:  [Descargar Oracle Database](https://www.oracle.com/es/database/technologies/112010-win64soft.html)  [Tutorial de Oracle Database (oracletutorial.com)](https://www.oracletutorial.com/)  [Sistema Gestor de Base de Datos Relacional de Oracle](http://www.oracle.com/technetwork/es/database/express-edition/downloads/index.html).  Entorno de desarrollo Oracle -> [SQL Developer](https://www.oracle.com/es/tools/downloads/sqldev-v192-downloads.html)  Con la instalación se crean el usuario SYSTEM y SYS, puedes asignar las contraseñas de ambos usuarios.  [Vídeo explicativo de la instalación de Oracle](https://www.youtube.com/watch?v=IKJDd2i3k8g). |
| **MySQL** | Es un sistema de gestión de base de datos **relacional, multihilo y multiusuario** seguramente el **más usado en aplicaciones** creadas como **software libre**. |
| Por un lado, se ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. |
| **Ventajas:**   * Velocidad al realizar las operaciones * Bajo costo en requerimientos para la elaboración de bases de datos * Facilidad de configuración e instalación. |
| **Instalación de MySQL**:  Sistema gestor de base de datos de MySQL -> [MySQL Workbench](https://dev.mysql.com/downloads/workbench/)  Vídeo explicativo: [Video explicativo de instalación (youtube.com)](https://www.youtube.com/watch?v=YKxJBHf0ntY) |
| Vamos a utilizar únicamente el **usuario root con la contraseña root**, este será el usuario administrador.  **Anota el puerto en el que escucha el servicio**. |
| **SQLite** | SQLite es un motor de base de datos SQL incorporado, contenida en una pequeña ​ biblioteca escrita en C.  **No tiene un proceso de servidor por separado**. SQLite lee y escribe directamente en archivos de disco ordinarios. |
| **Ventajas**:   * **SQLite no necesita ser "instalado"** antes de ser utilizado. * **No hay un proceso de servidor que deba iniciarse**, detenerse o configurarse. * Utiliza registros de longitud variable. * Su tamaño alrededor de 500K (es ligero) * Implementa  ACID (atomicity, consistency, isolation and durability) properties in SQLite to maintain database |
| Este sistema gestor de base de datos está **basado en archivos**, es de **codigo abierto**, **multiplataforma** y de acceso libre. |
| **Instalación de SQLite**:  Descargar SQLite: [Descargar gestor SQLite](https://www.sqlite.org/download.html).  Sistema gestor: [DB Browser for SQLite](https://sqlitebrowser.org/dl/).  Manual de instalación (en inglés): [Installing and Using SQLite on Windows | John Atten](http://johnatten.com/2014/12/07/installing-and-using-sqlite-on-windows/) |

## 9.6.- Bases de datos orientadas a documentos.

### 9.6.1.- Bases de datos XML.

Se utiliza para almacenar la gran cantidad de información en formato XML.

[Ya hemos visto que hay de dos tipos](#_7.-_Bases_de).

### 9.6.2.- Bases de datos MongoDB.

MongoDB viene de la palabra en inglés “homongous”, que significa enorme.

* No tiene **transacciones**. Puede simularlas usando alguna técnica, pero no tiene esa capacidad de forma nativa.
* Solo garantiza **operaciones atómicas a nivel de documento**.
* **Sistema de base de datos NoSQL orientado a documentos**, desarrollado bajo el concepto de código abierto.
* **En lugar de guardar los datos en registros, los guarda en documentos**, almacenados en BSON (representación binaria de JSON).
* A diferencia de las bases de datos relacionales**, no es necesario seguir un esquema**.
* Los documentos de una misma colección (colección = similar a una tabla en BB.DD relacionales), pueden tener esquemas diferentes.
* Cualquier aplicación que necesite **almacenar datos semi estructurados** puede usar **MongoDB**. Es el caso de las **típicas aplicaciones CRUD** o de muchos de los desarrollos web actuales.
* **Ventajas**:
  + Velocidad
  + Rico, pero sencillo **sistema de consulta** de los contenidos de la base de datos.
  + Balance perfecto entre rendimiento y funcionalidad.