



# TEMA-15-ACCESO-A-DATOS.pdf



**user\_4383848**



**Acceso a datos**



**1º Desarrollo de Aplicaciones Multiplataforma**



**Estudios España**

**Google Gemini:**  
**Plan Pro a 0€ durante 1 año.**  
**Tu ventaja por ser estudiante.**

Oferta válida hasta el 9 de diciembre de 2025

**Consigue la oferta**

Después 21,99€/mes



Convierte tus apuntes en podcasts.

Generar un resumen de audio



resumen temario

PDF



Deep Research

Canvas



Google Gemini:  
Plan Pro a 0€ durante 1 año.  
Tu ventaja por ser estudiante.



Oferta válida hasta el 9 de diciembre de 2025

Consigue la oferta

Después 21,99€/mes

Domina cualquier tema con el Aprendizaje Guiado.

Puedes explicarme como se crea un eclipse lunar completo y sus fases?

¡Claro vamos paso a paso para que lo entiendas a la perfección!



Aprendizaje Guiado

## TEMA 15 - ACCESO A DATOS

### PROGRAMACIÓN DE COMPONENTES DE ACCESO A DATOS

#### 1. CONCEPTO Y CARACTERÍSTICAS DE UN COMPONENTE:

Un componente es una **unidad de software** que encapsula un segmento de código con **ciertas funciones**. Los estilos de estos componentes pueden ser estilos proporcionados por el entorno de desarrollo (incluidos en la interfaz de usuario) o pueden ser no visuales, siendo sus funciones similares a las de las bibliotecas remotas. Los componentes del software tienen las siguientes características principales:

- **Un componente es una unidad ejecutable** que se puede instalar y utilizar de forma independiente.
- Puede **interactuar y operar con otros componentes desarrollados por terceros**, es decir, empresas o desarrolladores pueden utilizar componentes y agregarlos a las operaciones que se realizan, lo que significa que pueden estar compuestos por estos componentes.
- **No tienen estado**, al menos su estado no es visible desde 'fuera'.
- La programación orientada a componentes, **se puede asociar a los distintos engranajes electrónicos que**, en su conjunto, forman un sistema más grande.

Podemos afirmar que un componente está formado por:

1. **Atributos, operaciones y eventos:** Es parte de la interfaz del componente. Todo en su conjunto representaría el nivel sintáctico.
2. **Comportamiento:** Representa la parte semántica.
3. **Protocolos y escenarios:** Representa la **compatibilidad de las secuencias de los mensajes con otros componentes**, y la forma de actuar que tendrá el componente en diferentes escenarios donde llegará a ser ejecutado.
4. **Propiedades:** Las diferentes características que puede tener el componente.

#### 2. PROPIEDADES Y ATRIBUTOS DE UN COMPONENTE: INDEXADAS Y SIMPLES:

Las propiedades de los componentes se dividen en simples, indexadas, compartidas y restringidas. **Se puede verificar y modificar las propiedades del componente accediendo al método del componente o la función de acceso.** Hay **dos tipos de estas funciones**:

- **El método get** se utiliza para consultar o leer el valor de un atributo. La sintaxis general es la siguiente:
- **El método set** se utiliza para asignar o cambiar el valor de un atributo. Su sintaxis general en Java es:

Centrándonos en las propiedades o atributos indexados y simples, podemos decir, que **los atributos simples son atributos que representan un solo valor.**

Además de las propiedades de los valores simples y únicos, existe otro tipo de propiedades **más complejas que son muy similares a un conjunto de valores: los índices.** Todos los elementos de este tipo de atributo comparten el mismo tipo y se puede acceder a ellos por índice. Concretamente, **se puede acceder mediante el método de acceso que se mencionó anteriormente (get / set)**, aunque la llamada de estos métodos variará, ya que solo se puede acceder a cada propiedad a través de su índice.

### 3. PROPIEDADES Y ATRIBUTOS DE UN COMPONENTE: COMPARTIDAS Y RESTRINGIDAS:

Los **atributos compartidos** se refieren a los datos mediante los que informa a todos los interesados sobre el atributo, cuando cambian, por lo que solo se les notifica cuando cambia el atributo. El mecanismo de notificación **se basa en eventos**, es decir, hay un componente de origen que notifica al componente receptor cuando un atributo compartido cambia a través de un evento. Debe quedar claro que estas propiedades no son bidireccionales, por lo que el componente receptor no puede responder.

Para que un **componente permita propiedades compartidas**, debe admitir dos métodos para registrar componentes que estén interesados en cambios de propiedad (uno para agregar y otro para eliminar). Su sintaxis general en Java sería:

`addPropertyChangeListener (PropertyChangeListener x)` y `removePropertyChangeListener (PropertyChangeListener x)`.

Las **propiedades restringidas** buscarán la aprobación de otros componentes antes de cambiar su valor. Al igual que con los atributos compartidos, se deben proporcionar dos métodos de registro para el receptor.

`addVetoableChangeListener (VetoableChangeListener x)` y  
`removeVetoableChangeListener (VetoableChangeListener x)`

### 4. EVENTOS:

La iteración entre componentes se denomina **control activo**, es decir, el funcionamiento de un componente se activa mediante la llamada de otro componente. Además del control activo (la forma más común de invocar operaciones), existe otro método llamado **control reactivo**, que se refiere a eventos de componentes, como los permitidos por el modelo de componentes EJB.

En el **control reactivo**, los componentes pueden generar eventos correspondientes a solicitudes de invocación de operaciones. Posteriormente, otros componentes del sistema recolectarían estas solicitudes, y activarían llamadas a operaciones para sus propósitos de procesamiento.

### 5. INTROSPECCIÓN:

La introspección, por tanto, **se puede definir como un mecanismo a través del cual se pueden descubrir las propiedades, métodos y eventos contenidos en el componente**. Los componentes admiten la introspección de dos formas:

- **Mediante el uso de convenciones de nomenclatura específicas**, que se conocen al nombrar las características de los componentes. Para EJB, la clase **Introspector** comprueba el EJB para encontrar esos patrones de diseño y descubrir sus características.
- Proporcionando información clara sobre **atributos, métodos o eventos de clases** relacionadas.

En particular, **EJB admite múltiples niveles de introspección**. En un nivel bajo, esta introspección se puede lograr mediante la posibilidad de reflexión. Estas características permiten a los objetos Java descubrir información sobre métodos públicos, campos y constructores de clases cargados durante la ejecución del programa. La reflexión permite la introspección de todos los componentes del software, y todo lo que se tiene que hacer es declarar el método o variable como público para que pueda ser descubierto a través de la reflexión.

## 6. PERSISTENCIA DEL COMPONENTE:

Para EJB, **la persistencia se proporciona con la biblioteca JPA de Java**. Para usar JPA, se requiere el uso al menos del **JDK 1.5 en adelante**, ya que amplía las nuevas especificaciones del lenguaje Java, como son las anotaciones.

Las clases que componen JPA son las siguientes:

- **Persistence**. La clase `javax.persistence.Persistence` contiene un método auxiliar estático, el cual usamos para tener un **objeto EntityManagerFactory** de forma independiente del que se obtiene a través de JPA.
- **EntityManagerFactory**. La clase `javax.persistence.EntityManagerFactory` extraemos **objetos de tipo EntityManager** usando el conocido patrón de Factory.
- **EntityManager**. La clase `javax.persistence.EntityManager` es la interfaz JPA principal para la persistencia de aplicaciones. Cada EntityManager puede crear, leer, modificar y eliminar un grupo de objetos persistentes.
- **Entity**. La clase Entity es una **anotación Java**. La encontramos al mismo nivel que las clases Java serializables. Cada una de estas entidades las representamos como registros diferentes en base de datos.
- **EntityTransaction**. Permite operaciones conjuntas con datos persistentes, de tal forma, que pueden crear grupos para persistir distintas operaciones. Si todo el grupo realiza las operaciones sin ningún problema se persistirá, de lo contrario, todos los intentos fallarán. En caso de error, la base de datos realizará rollback y volverá al estado anterior.
- **Query**. Cada proveedor de JPA ha implementado la interfaz `javax.persistence.Query` para encontrar objetos persistentes procesando ciertas condiciones de búsqueda. JPA utiliza JPQL y SQL para estandarizar el soporte de consultas. Podremos obtener un **objeto Query** a través del EntityManager.

## 7. EMPAQUETADO DE COMPONENTES:

Un módulo J2EE consta de uno o más componentes J2EE del mismo tipo de contenedor, y descriptores de implementación de componentes de este tipo. **El descriptor de despliegue del módulo EJB** especifica los distintos atributos de nuestra transacción y la autorización de seguridad del EJB. Los módulos JavaEE sin descriptores de implementación de aplicaciones se pueden implementar como módulos separados.

Antes de EJB 3.1, todos los EJB debían estar empaquetados en estos archivos.

Como buena parte de todas las aplicaciones **J2EE, contienen un front-end web y un back-end EJB, lo que significa, que se debe crear un .ear que contenga una aplicación con dos módulos: .war y ejb-jar.**

# Google Gemini: Plan Pro a 0€ durante 1 año. Tu ventaja por ser estudiante.



Oferta válida hasta el 9 de diciembre de 2025

Consigue la oferta

Después 21,99€/mes

Domina cualquier tema con el Aprendizaje Guiado.

Puedes explicarme como se crea un eclipse lunar completo y sus fases?

¡Claro vamos paso a paso para que lo entiendas a la perfección!



Aprendizaje Guiado

Ésta es una buena práctica en el sentido de establecer una clara separación estructural entre la parte 'delantera' y la 'trasera.' No obstante, esta diferenciación será complicado delimitarla en aplicaciones simples.

Hay **cuatro tipos de módulos J2EE** para aplicaciones web EJB:

- **Módulos EJB**, que contienen archivos con clases EJB y descriptores de despliegue EJB. El módulo EJB está empaquetado como un archivo JAR con extensión .jar.
- **El módulo Web**, que contiene archivos con Servlet, archivos JSP, archivos de soporte de clases, archivos GIF y HTML y descriptores de implementación de aplicaciones web. El módulo web está empaquetado como un archivo JAR con una extensión .war (archivo web).
- **El módulo de la aplicación cliente**, que contiene archivos con clases y descriptores de la aplicación cliente. El módulo de la aplicación cliente está empaquetado como un archivo JAR con extensión .jar.
- **Módulo adaptador de recursos**, que contiene todas las interfaces, clases, bibliotecas nativas y otros documentos de Java y sus descriptores de implementación del adaptador de recursos.