Hola, te saluda Ubaldo Acosta. Bienvenida o bienvenido nuevamente. Espero que estés listo para comenzar con esta lección.

Vamos a estudiar el tema de cómo crear una capa de acceso a datos utilizando el API de JDBC.

¿Estás listo? Ok, ¡Vamos!

Vamos a revisar a continuación el tema de las buenas prácticas y el concepto de patrones de diseño.

En una arquitectura Java Empresarial es común que la aplicación se divida en varias capas y cada una es responsable de realizar tareas específicas. Debido a la complejidad de los sistemas y los problemas a los que nos enfrentamos diariamente existe el concepto de buenas prácticas y el concepto de patrones de diseño con el objetivo de reducir los problemas mencionados.

Las buenas prácticas incluyen temas de codificación, división de responsabilidades en capas lógicas, entre otros temas. A su vez los patrones de diseño, como su nombre lo dice, resuelven un problema que se presenta de manera recurrente (patrón) en el desarrollo de sistemas, en particular en sistemas orientados a objetos.

Un tema fundamental en el diseño de sistemas es la **cohesión y el acoplamiento.** Este tema entra en juego cuando manejamos varias capas lógicas ya que cada capa es responsable de cierta funcionalidad.

Como podemos observar en la figura, tenemos la capa de presentación, la capa de negocio y la capa de datos, cada una de estas capas tiene ciertas responsabilidades.

La **Capa de Presentación** se encarga de administrar el flujo entre las distintas pantallas del sistema, así como procesar los datos del usuario (formularios) y desplegar la información al mismo.

Por otro lado la **Capa de Negocio** se encarga de procesar la lógica del negocio y/o los servicios que debe manejar nuestro sistema.

Por último la **Capa de Datos** es la encargada de la comunicación con la base de datos, archivos entre otras fuentes de información.

Las capas se intercomunican para procesar la información, esto es, desde que un usuario hace una petición, hasta que el sistema responde al usuario de nueva cuenta.

A continuación vamos a describir algunas características de cada uno de estos temas.

Vamos a revisar el tema de cohesión y acoplamiento, los cuales juegan un rol central en el diseño del software.

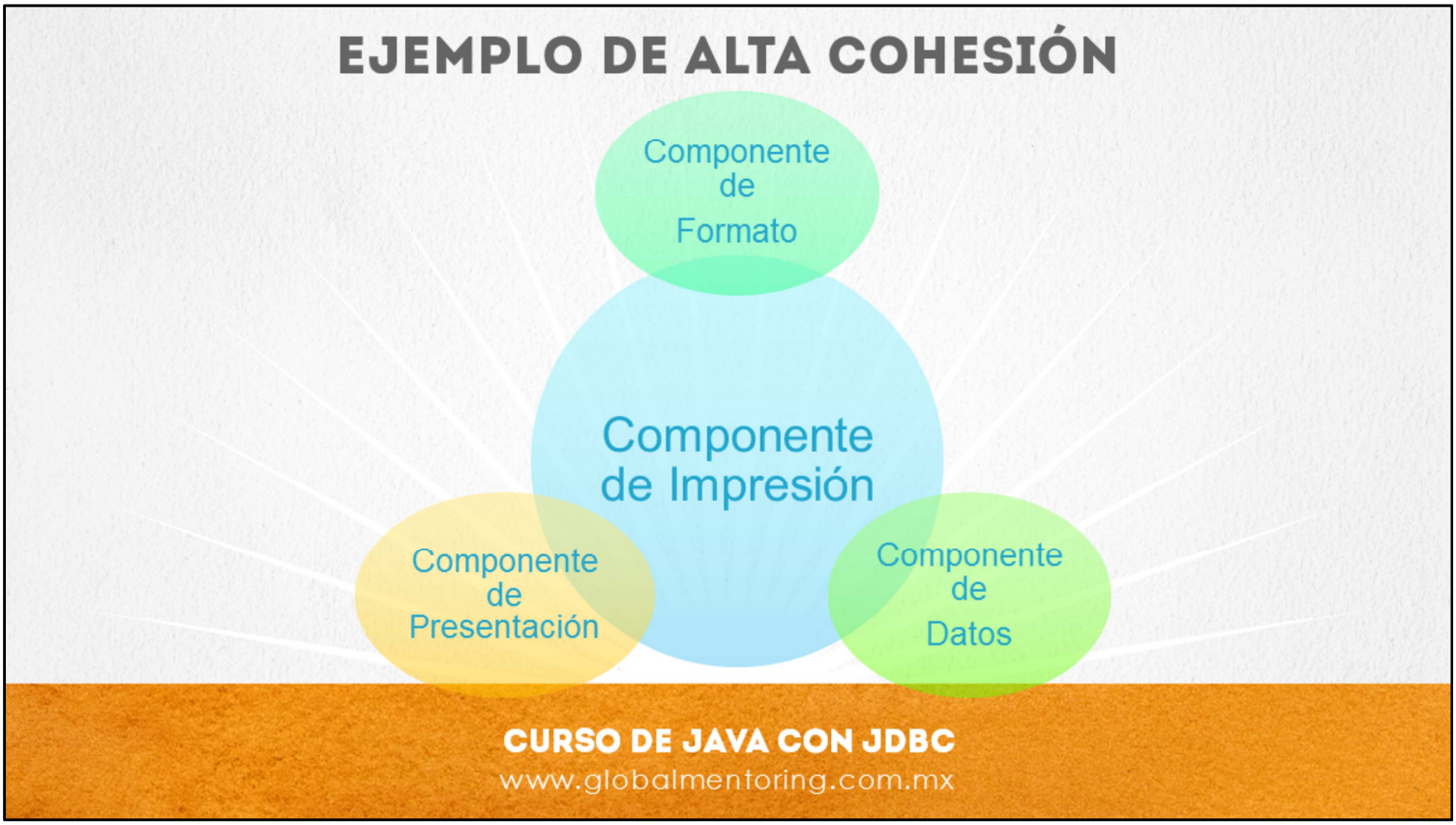
Al diseñar nuestros módulos de software seguramente requeriremos de cambios posteriores conforme las necesidades de la aplicación vayan cambiando. Por lo que el diseño de nuestro sistema puede impactar de manera directa en el tiempo y costo asociado para realizar dichos cambios, por lo tanto vamos a revisar estos conceptos de cohesión y acoplamiento.

Podemos entender la cohesión como la medida en la que un componente se dedica a realizar solamente la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes.

El acoplamiento por otro lado, mide el grado de dependencia entre dos o más elementos, estos elementos pueden ser módulos clases o cualquier otro componente de software.

El objetivo final del desarrollo de software es crear sistemas con bajo acoplamiento y alta cohesión, estas son 2 características que debemos de tener en cuenta al crear nuestras aplicaciones.

La división en capas de manera lógica en una arquitectura Java Empresarial (Java EE), introduce un bajo acoplamiento y una alta cohesión de manera automática, debido a que permite que el número de relaciones entre cada capa sea el menor posible y aumente la cohesión en cada capa, debido a que tenemos una división de responsabilidades de manera muy específica y clara.

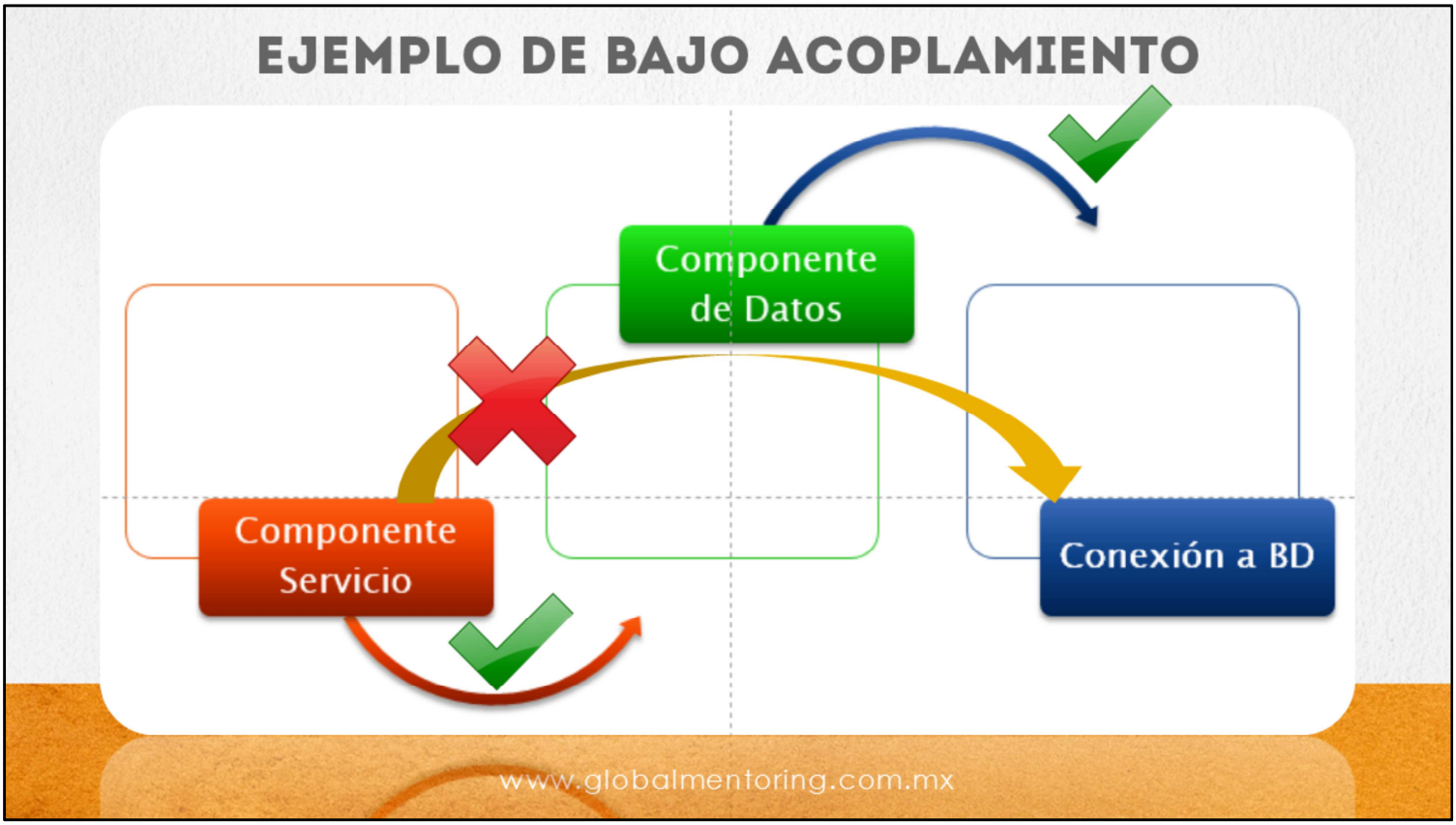


En este ejemplo, podemos observar el componente de impresión, el cual únicamente tiene como responsabilidad imprimir cierta información. Para realizar las demás tareas se apoya de otros componentes, por lo que se dice que tenemos una alta cohesión, debido a que cada componente se dedica solo a la tarea para la cual fue creado.

Para lograr la alta cohesión nos vamos apoyar de componentes como puede ser el componentes de datos para obtener la información (de alguna base de datos o algún archivo), el componente para presentar la información y un componente para dar formato a la información que seleccione el usuario.

El componente de impresión en el diseño de clases no es necesariamente el diseño más importante, ya que podemos observar que al imprimir no necesariamente debemos de tener todas la tareas siendo ejecutadas por este componente, si no que nos podemos apoyar de más componentes para realizar finalmente la tarea de impresión. Algunos de estos componentes incluso pueden ser parte de otras aplicaciones y podemos reutilizar código y componentes de otras aplicaciones, siempre y cuando los componentes tengan pocas dependencias y el código haya sido diseñado teniendo en cuenta una alta cohesión.

Este tipo de diseños permite que los cambios sean más fáciles de identificar y de realizar, pero genera componentes más pequeños, es decir, con menos código, lo cual podría incrementar el acoplamiento y según comentamos esto no es deseable. Para corregir esto se debe de manejar un **balance** entre el diseño de componentes y tratar de que no sean módulos demasiado pequeños. También debemos de tener cuidado en que los componentes no sean muy grandes, es decir, que no tengan demasiado código, de lo contrario los errores van hacer cada vez más difíciles de resolver y/o identificar.



Como podemos observar en la figura, estamos mostrando componentes que presentan un flujo para la extracción de información con una base de datos.

Podemos observar el menor número de relaciones posibles, ya que estamos evitando la comunicación directa entre la capa de servicio y la capa que crea la conexión a la base de datos (bajo acomplamiento).

Si tuviéramos una relación entre el componente de servicio y la conexión de base de datos, estaríamos generando más relaciones de las necesarias y por lo tanto generaríamos mayor dependencia entre cada uno de los componentes.

Uno de los objetivos en el diseño de software es hacer los componentes reutilizables, por lo que entre mayor dependencia existe entre cada componente más difícil va a ser tomar el módulo y adecuarlo para utilizarlo en otra aplicación.

Vamos a revisar ahora el tema de patrones de diseño. Un patrón de diseño es una guía que puede involucrar varias clases y que a su vez nos permite resolver un problema que se presenta de manera repetitiva.

Cuando hablamos de las capas de una arquitectura Java EE, en este caso la capa de presentación, la capa de servicio y la capa de acceso a datos, cada capa puede tener varios patrones de diseño y la capa de presentación podemos observar patrones como el MVC. Este patrón significa ***modelo-vista-controlador*** y su objetivo es dividir las responsabilidades en estos tres rubros, un modelo, una vista y un controlador. El patrón Front Controller nos permite proporcionar una entrada única al usuario y por lo tanto aquí podemos aplicar varias características como pueden ser algún tipo de seguridad, validaciones, entre otras validaciones.

El patrón DTO (data transfer object) representa un objeto del dominio del problema, en ocasiones puede ser una clase de entidad, esto es, una clase que se persiste o guarda en una base de datos. Este patrón aparece en las tres capas como podemos observar, debido a que se utiliza para transferir una entidad o una lista de entidades de cierto tipo entre las distintas capas de la aplicación, por ejemplo un usuario puede estar solicitando un listado de personas, entonces la capa de presentación procesa la petición y solicita posteriormente a la capa de servicio que ejecute el método encontrar listado de personas, posteriormente la capa de servicios accede a la capa de datos para que podamos recuperar el listado de personas y posteriormente vamos a regresar un arreglo de tipos DTO de tipo Persona. El objeto DTO en este caso es un objeto de tipo persona y entonces comenzamos a regresar la información hasta que damos una respuesta al usuario con el listado de personas que ha solicitado.

En la capa de negocio tenemos varios patrones, como pueden ser el patrón Business Delegate, el cual se encarga de los detalles de llamar algún método del servicio y a su vez también tenemos el patrón Services Locater, el cual es utilizado por el patrón Business Delegate para localizar los servicios si es que se encuentran en algún directorio como pueden ser , algún directorio de tipo JNDI (Java Naming and Directory Interface). Estos patrones los veremos en cursos más avanzados, de momento sólo es importante saber que existe una separación de responsabilidades en cada una de las capas de una aplicaciones Java Empresarial.

Por último, en la capa de datos tenemos el patrón DAO, este patrón se encarga de extraer y almacenar información en la base de datos, y este es uno de los patrones que utilizaremos en este curso.

Cabe mencionar que existe un catálogo extenso de patrones de diseño para Java, por lo que si requieres más información de este tema pueden consultar: http://www.oracle.com/technetwork/java/patterns-139816.html

Cabe mencionar que las capas que estamos mostrando son las más comunes que se utilizan en el desarrollo de aplicaciones Java, pero no son las únicas e incluso podríamos tener menos capas, en ocasiones únicamente tenemos una capa de presentación que se comunica con la base de datos, aunque este tipo de prácticas no son recomendables, a menos que sean prototipos. El tipo de arquitectura final dependerá realmente del problema que queramos resolver, pero nuestra recomendación es que utilicemos como mínimo estas tres capas para que los cambios y el mantenimiento en nuestros sistemas sean muchos más sencillos de realizar.

EJERCICIOS CURSO FUNDAMENTOS DE JAVA

- **ABRIR LOS ARCHIVOS DE EJERCICIOS EN PDF.**
- **EJERCICIO**: Ejercicio Creación Capa de Datos con JDBC.

CURSO DE JAVA CON JDBC
www.globalmentoring.com.mx

En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- Lógica de Programación
- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration
- Android con Java
- HTML5 y CSS3

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx
Email: informes@globalmentoring.com.mx