
Universidad
Tecmilenio®

Universidad Tecmilenio

Ingeniería de Software

Arquitectura de aplicación

Semana 7

Bienestar-mindfulness

Te invito a realizar la siguiente actividad de bienestar-mindfulness antes de comenzar a revisar el tema:

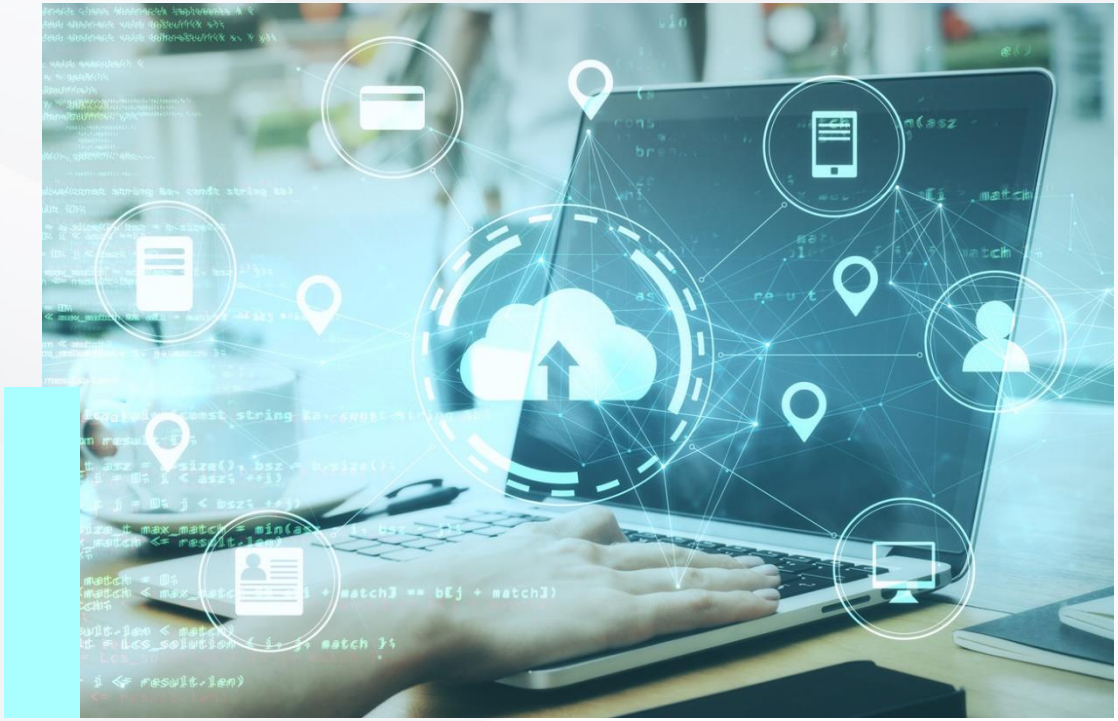
https://youtu.be/xC3k9R9FXo8

Introducción

La arquitectura de los sistemas de información es parte fundamental de la calidad del producto de software y dependerá del objetivo que se quiera alcanzar y el uso que se le quiera dar.

Explicación

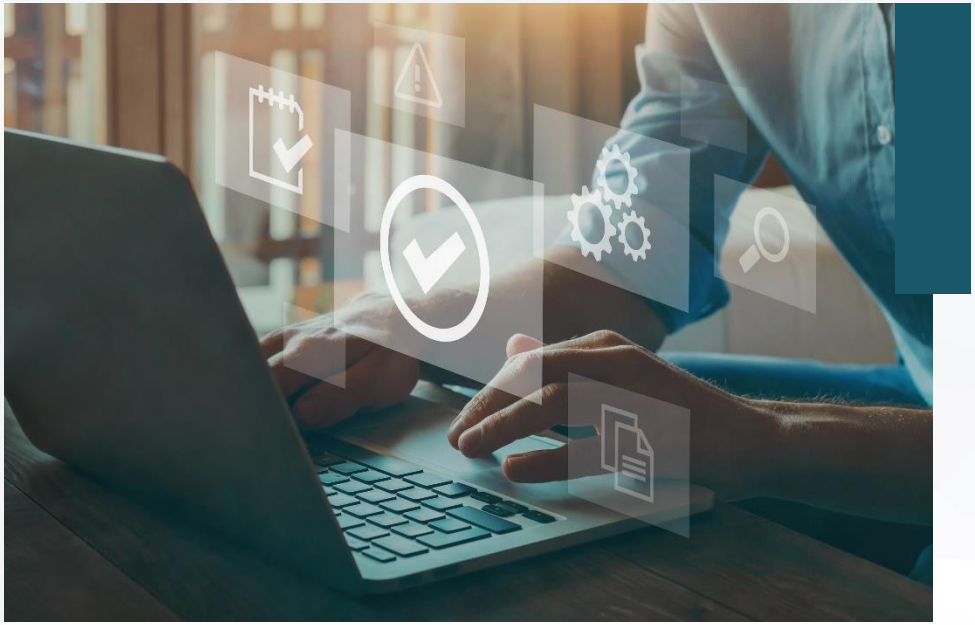
Una transacción de base de datos es una secuencia de operaciones que se trata como una sola unidad, es decir, una unidad atómica.



Explicación

Los sistemas de información que procesan las transacciones diarias son llamados transaction processing systems (TPS).

Permiten registrar las operaciones diarias de la empresa, como por ejemplo: ventas, pagos, nómina, facturación, registros de entradas y salidas de inventarios, por mencionar algunos.



Explicación

Sistemas de procesamiento de datos por lotes (Batch data processing systems): estas arquitecturas obtienen o ingresan datos en grupos llamados lotes, desde o hacia archivos o base de datos.

Ejercicio

Describe el funcionamiento de la arquitectura de un sistema de procesamiento basado en eventos (Event processing system).

Cierre

Las organizaciones han apostado por desarrollos de sistemas que sean un apoyo a los procesos de la cadena de valor. Las arquitecturas de software deben responder a las necesidades de las empresas generando diseños versátiles, robustos, modificables e innovadores.

Bibliografía

- Reese, M. (2015). Natural Language Processing with Java. Inglaterra: Packt Publishing.
- Laudon, K y Laudon J. (2012). Sistemas de información gerencial (12ª ed.). EE. UU: Pearson.
- Sommerville, I. (2008). Event processing systems. Recuperado de http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Architecture/AppArch/EventProc.html
- Sommerville, I. (2011). Ingeniería de Software (9ª ed.). México: Pearson.

Universidad Tecmilenio

Ingeniería de Software

Otros patrones

Semana 7

Introducción

Un sistema embebido se encuentra en una infinidad de instrumentos o aparatos electrónicos como un horno o un reloj digital.

Explicación

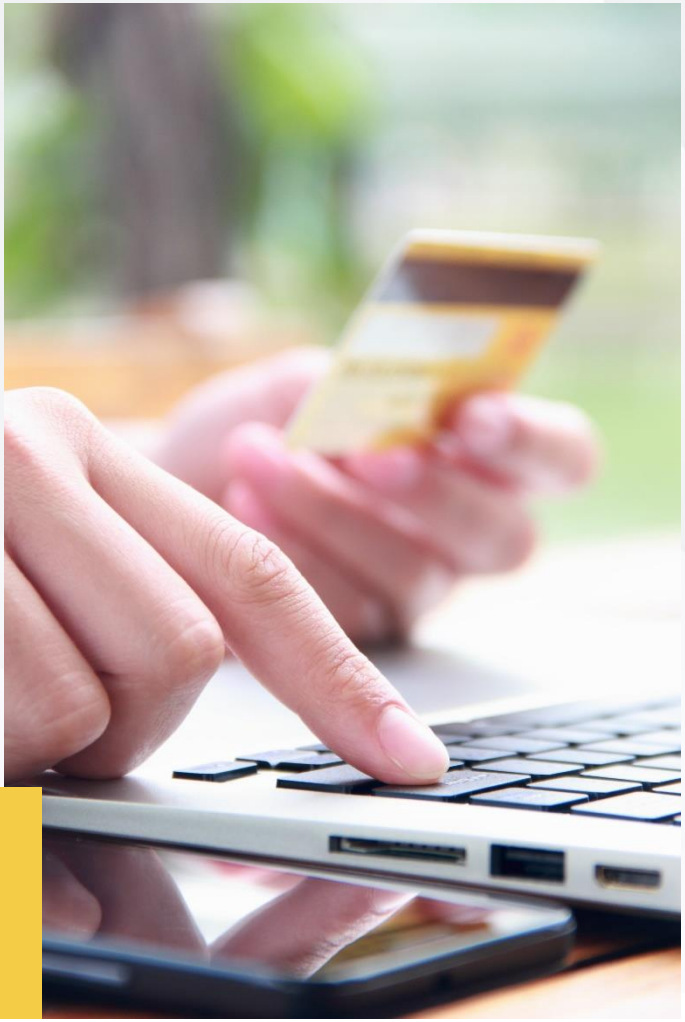
Un sistema embebido es aquel que se encuentra almacenado en el hardware de la memoria ROM en espera de responder en tiempo real (en un plazo de tiempo) a eventos del entorno de sistema.

Explicación

La arquitectura multiniveles permite distribuir funciones del sistema en subgrupos ya sea para facilitar la operación de la infraestructura o bien por razones administrativas, en las que ciertos grupo de personas se hacen responsables de los equipos (hardware) que ofrecen un conjunto de servicios al resto de la organización.

Explicación

Los sistemas embebidos son una pieza fundamental de la tecnología actual: cajeros automáticos, decodificadores de señal de televisión, sistemas de radares, componentes satelitales, sensores de movimiento, parquímetros digitales, son parte de una lista enorme de hardware que requiere de una arquitectura de software que proporcionará las instrucciones necesarias para dar respuesta en tiempo real a estímulos externos.



Ejercicio

Enlista 5 ventajas de la arquitectura multi-niveles (multi-tier).

Cierre

El patrón arquitectónico multi-tier permite diseñar sistemas complejos que requieren ser divididos en funciones similares, lo que da oportunidad de ser asignados a diferentes equipos de personas con competencias específicas.

Bibliografía

- Bass, L., Clements, P. y Kazman, R. (2012). Software Architecture in Practice. EE. UU: Addison - Wesley Professional.
- Sommerville, I. (2011). Ingeniería de Software (9ª ed.). México: Pearson.

Universidad Tecmilenio

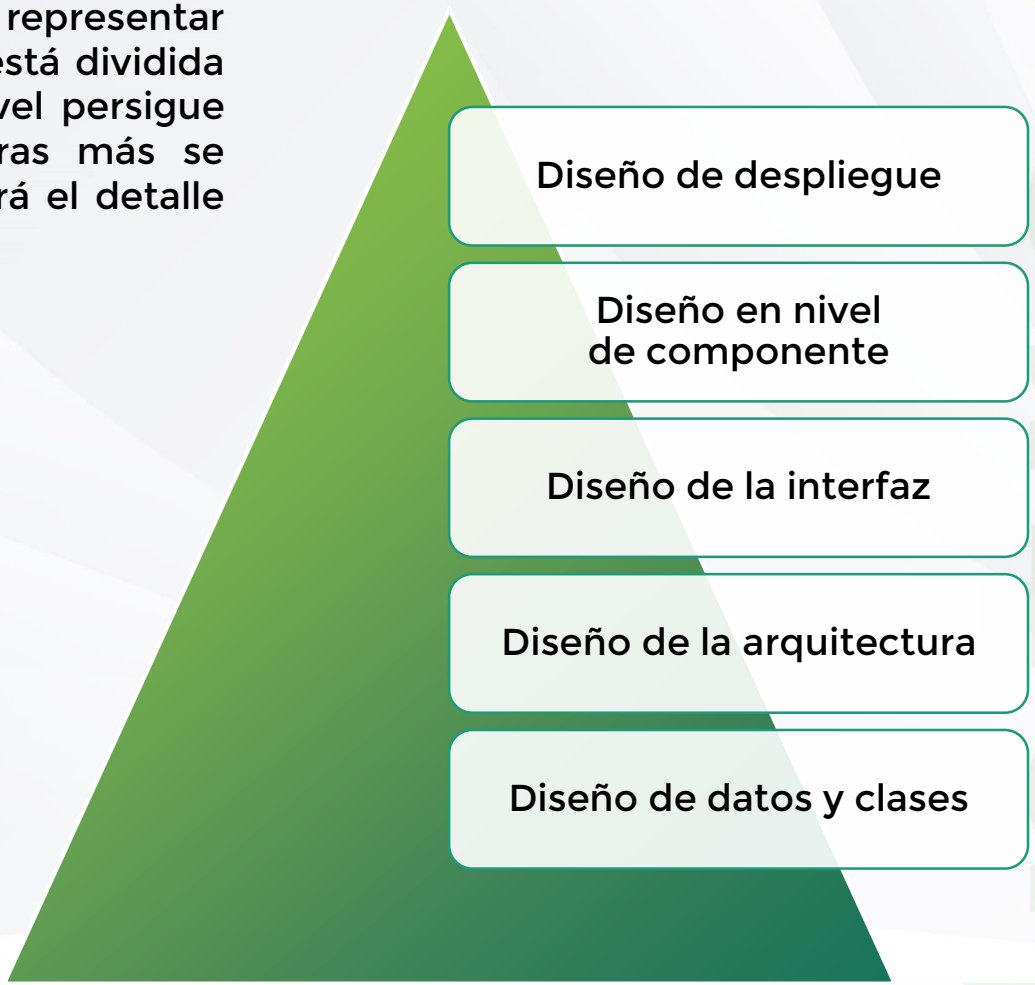
Ingeniería de Software

Fundamentos del diseño de componentes

Semana 7

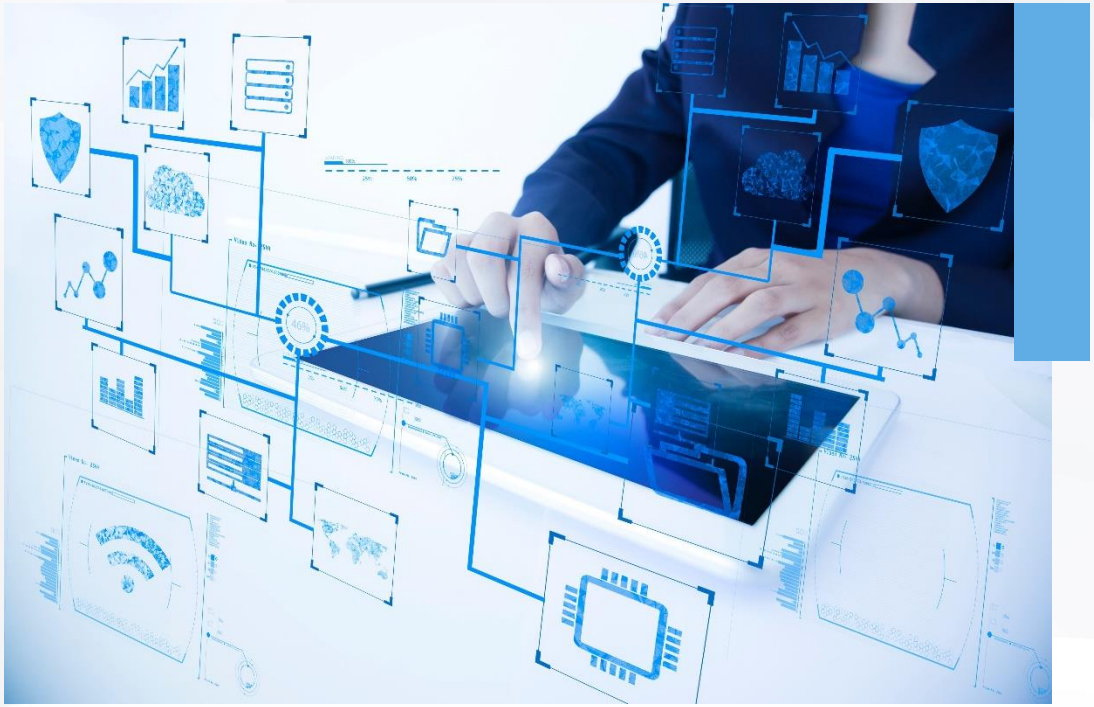
Introducción

El diseño de software se puede representar a través de una pirámide que está dividida en niveles del diseño. Cada nivel persigue un objetivo específico. Mientras más se acerquen a la punta, mayor será el detalle que debe incluirse.



Explicación

Un componente de software es **cualquier parte de un sistema que sea modular, desplegable y sustituible**, que requiere de un conjunto de interfaces para comunicarse con otros componentes internos o externos de un sistema.



Explicación

Un componente es elemento funcional de un programa que incorpora lógica de procesamiento, A este componente se le conoce como módulo y se le pueden asignar tres funciones diferentes.

- Control
- Dominio
- Infraestructura

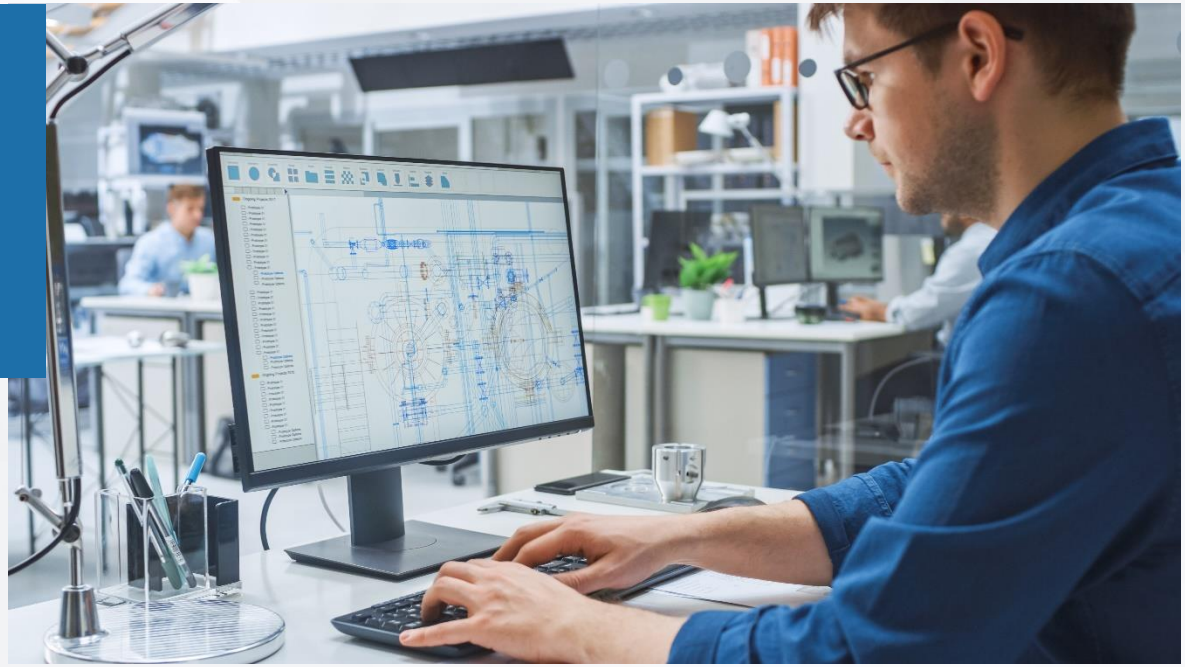
Explicación

Los principios de diseño de componentes están orientados a objetos que ayudan a evitar que se introduzcan defectos al software y facilitan la modificación posterior:

- Abierto-cerrado
- Sustitución de Loskov
- Inversión de la dependencia
- Segregación de la interfaz

Ejercicio

Describe los tres principales lineamientos o recomendaciones para el diseño de componentes.



Cierre

El diseño de componentes es parte fundamental de todo diseño modular; además, permite la reutilización de código, ya sea de otros proyectos anteriores o bien, un código que puede servir para futuros proyectos.

Bibliografía

- Freeman, E., Freeman E., Sierra, K. y Bates, B. (2004). Head First Design Patterns. USA: O'Reilly.
- Martin, R. (2000). Design principles and Design Patterns Recuperado de http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf
- Pressman, R. (2010). Ingeniería de Software. Un enfoque práctico (7ª ed.). México: McGraw-Hill.

Universidad Tecmilenio

Ingeniería de Software

Proceso de diseño por componentes

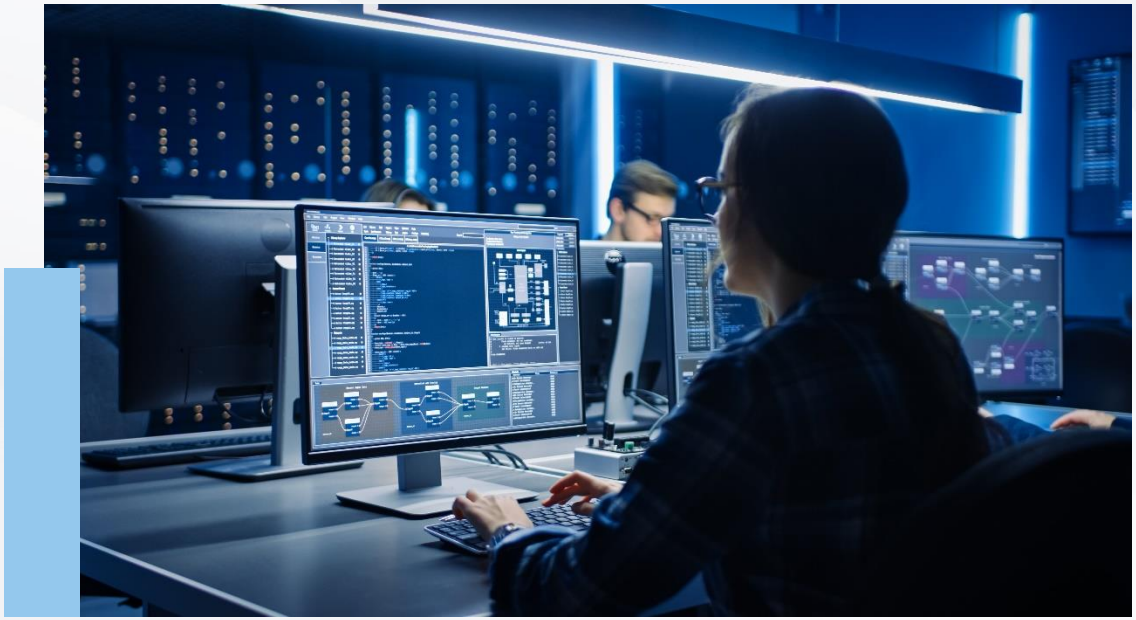
Semana 7

Introducción

El diseño de software basado en componentes es parecido a construir un sistema sin tener que empezar desde abajo, es decir, permite aprovechar la reutilización de módulos prefabricados que puedas tomar cada vez que requieras, y si acaso no encuentras el componente que necesitas, puedes crear uno con la idea de hacerlo reutilizable en caso de que en un futuro requieras de ese componente.

Explicación

En todo diseño basado en componentes deben existir dos características que permitan establecer la capacidad que tiene cada componente de interactuar entre sí: la cohesión y el acoplamiento.

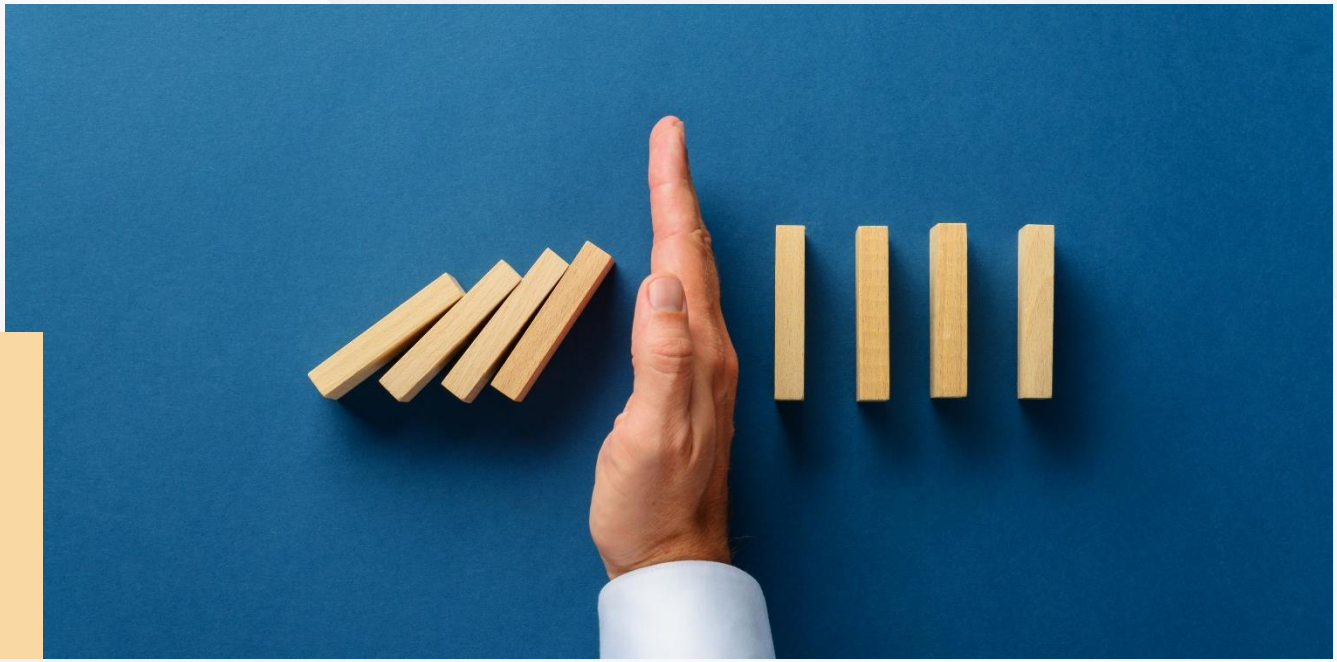


Explicación

El acoplamiento es la medición cuantitativa del grado en el que las clases se conectan una con otra.

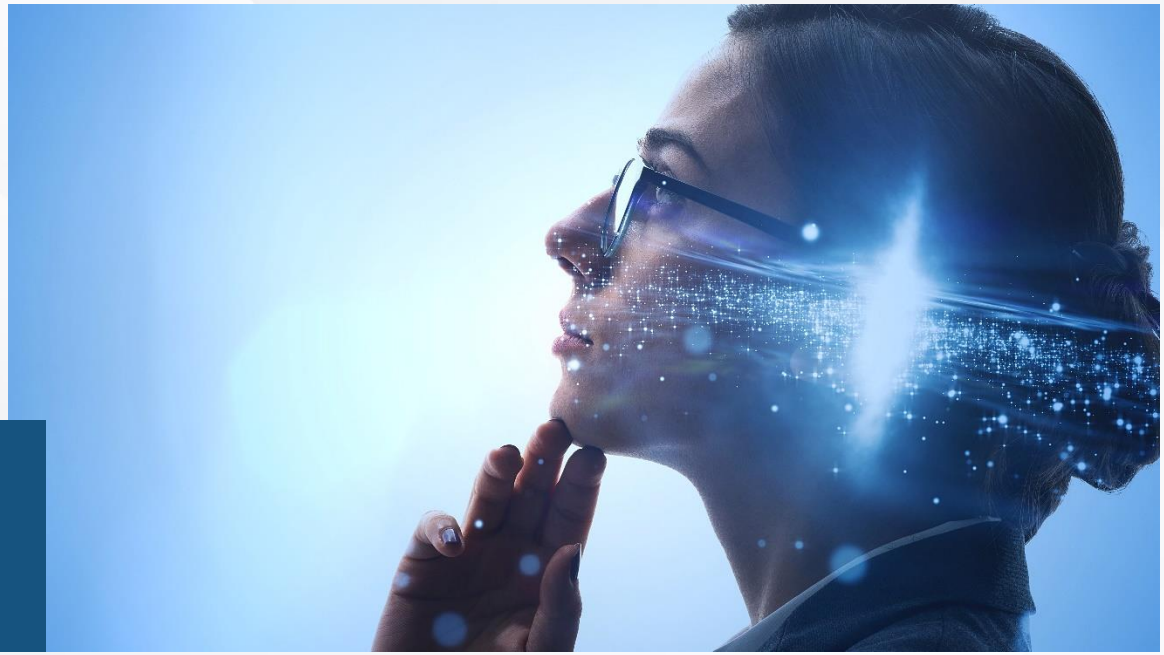
Explicación

La cohesión es la relación que mantienen los componentes o clases a través de sus atributos y operaciones.



Ejercicio

Describe los siete pasoso del diseño a nivel de componente.



Cierre

El diseño basado en componentes ofrece grandes ventajas a los desarrollos de sistemas que se espera crezcan a través del tiempo y reduzcan el tiempo de entrega.

Generar un código que pueda ser reutilizable es una tarea que requiere un mayor esfuerzo por parte del programador, sin embargo, los beneficios de reducir el tiempo que trae consigo serían razón suficiente para hacerlo.

Bibliografía

- Pressman, R. (2010). Ingeniería de Software. Un enfoque práctico (7ª ed.). México: McGraw-Hill.
- Sommerville, I. (2011). Ingeniería de Software (9ª ed.). México: Pearson.
- Tracz, W. (1995). Third International Conference on Software Reuse – Summary.
  ACM Software. Engineering Notes. Vol. 20, Núm. 2. pp. 21-22.