



# Ingenieria de software

*Diseño Arquitectónico*



# Diseño Arquitectónico

**El diseño arquitectónico se interesa por entender cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global de ese sistema. En el modelo del proceso de desarrollo de software, como se mostró en el capítulo 2, el diseño arquitectónico es la primera etapa en el proceso de diseño del software. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación.**



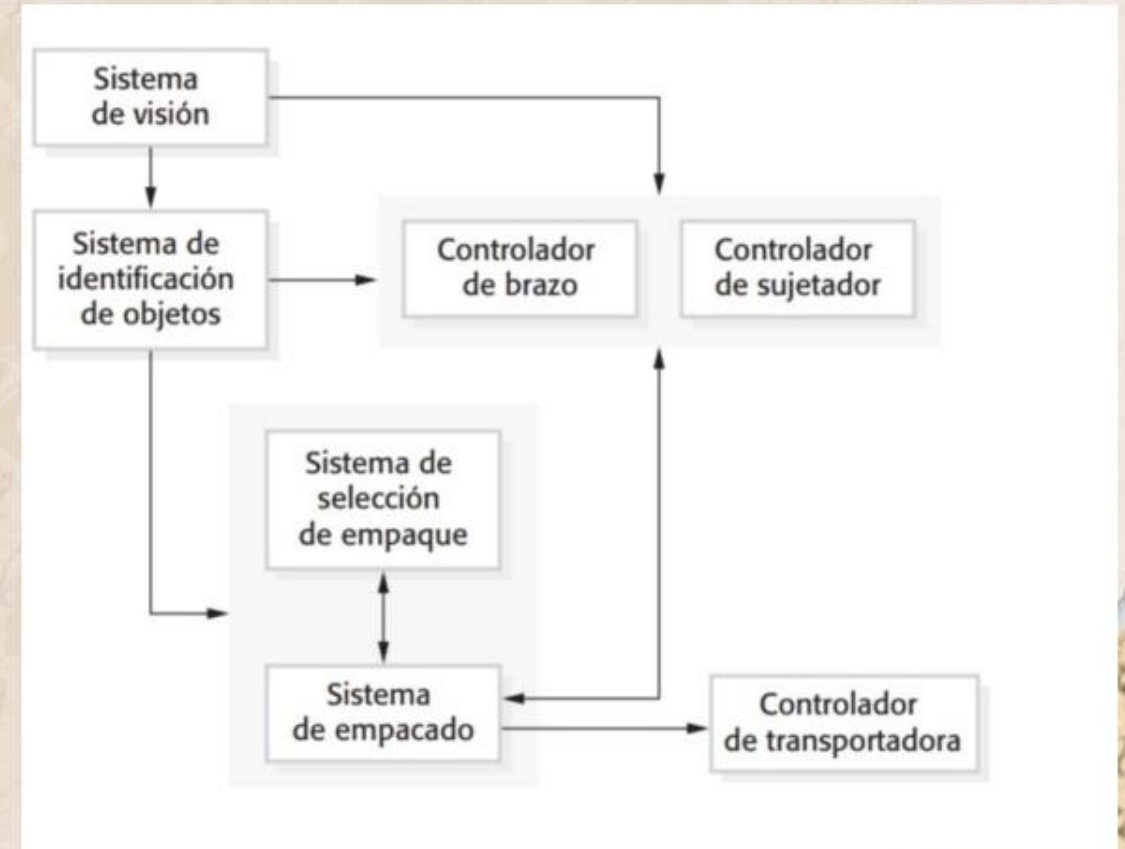


Las arquitecturas de software se diseñan en dos niveles de abstracción, que en este texto se llaman arquitectura en pequeño y arquitectura en grande:

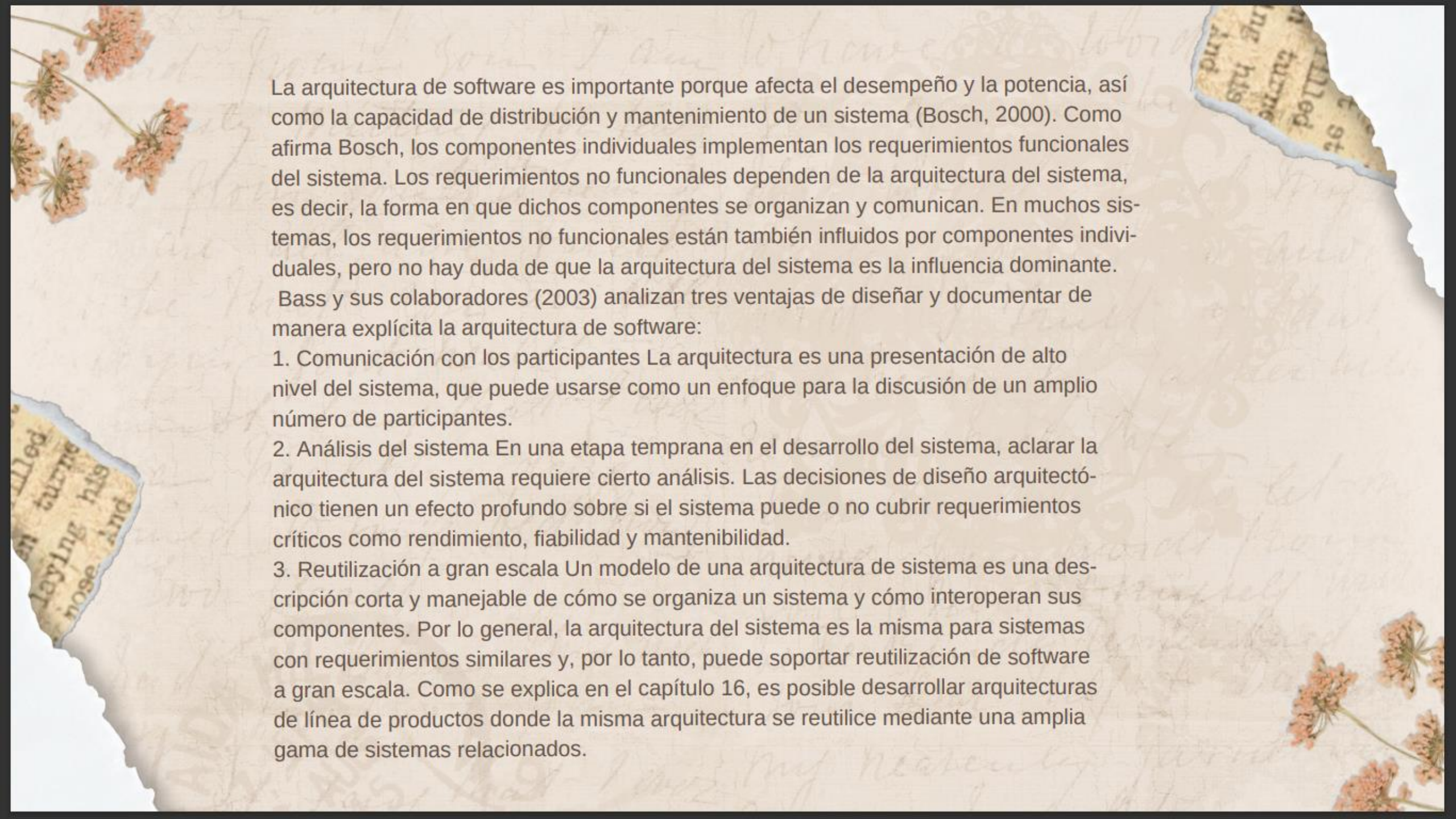
1. La arquitectura en pequeño se interesa por la arquitectura de programas individuales. En este nivel, uno se preocupa por la forma en que el programa individual se separa en componentes. Este capítulo se centra principalmente en arquitecturas de programa.

2. La arquitectura en grande se interesa por la arquitectura de sistemas empresariales complejos que incluyen otros sistemas, programas y componentes de programa.

Tales sistemas empresariales se distribuyen a través de diferentes computadoras, que diferentes compañías administran y poseen. En los capítulos 18 y 19 se cubren las arquitecturas grandes; en ellos se estudiarán las arquitecturas de los sistemas distribuidos.







La arquitectura de software es importante porque afecta el desempeño y la potencia, así como la capacidad de distribución y mantenimiento de un sistema (Bosch, 2000). Como afirma Bosch, los componentes individuales implementan los requerimientos funcionales del sistema. Los requerimientos no funcionales dependen de la arquitectura del sistema, es decir, la forma en que dichos componentes se organizan y comunican. En muchos sistemas, los requerimientos no funcionales están también influidos por componentes individuales, pero no hay duda de que la arquitectura del sistema es la influencia dominante.

Bass y sus colaboradores (2003) analizan tres ventajas de diseñar y documentar de manera explícita la arquitectura de software:

1. Comunicación con los participantes La arquitectura es una presentación de alto nivel del sistema, que puede usarse como un enfoque para la discusión de un amplio número de participantes.
2. Análisis del sistema En una etapa temprana en el desarrollo del sistema, aclarar la arquitectura del sistema requiere cierto análisis. Las decisiones de diseño arquitectónico tienen un efecto profundo sobre si el sistema puede o no cubrir requerimientos críticos como rendimiento, fiabilidad y mantenibilidad.
3. Reutilización a gran escala Un modelo de una arquitectura de sistema es una descripción corta y manejable de cómo se organiza un sistema y cómo interoperan sus componentes. Por lo general, la arquitectura del sistema es la misma para sistemas con requerimientos similares y, por lo tanto, puede soportar reutilización de software a gran escala. Como se explica en el capítulo 16, es posible desarrollar arquitecturas de línea de productos donde la misma arquitectura se reutilice mediante una amplia gama de sistemas relacionados.



Las aparentes contradicciones entre práctica y teoría arquitectónica surgen porque hay dos formas en que se utiliza un modelo arquitectónico de un programa:

1. Como una forma de facilitar la discusión acerca del diseño del sistema. Una visión arquitectónica de alto nivel de un sistema es útil para la comunicación con los participantes de un sistema y la planeación del proyecto, ya que no se satura con detalles. Los participantes pueden relacionarse con él y entender una visión abstracta del sistema. En tal caso, analizan el sistema como un todo sin confundirse por los detalles. El modelo arquitectónico identifica los componentes clave que se desarrollarán, de modo que los administradores pueden asignar a individuos para planear el desarrollo de dichos sistemas.

2. Como una forma de documentar una arquitectura que se haya diseñado. La meta aquí es producir un modelo de sistema completo que muestre los diferentes componentes en un sistema, sus interfaces y conexiones. El argumento para esto es que tal descripción arquitectónica detallada facilita la comprensión y la evolución del sistema.

Los diagramas de bloque son una forma adecuada para describir la arquitectura del sistema durante el proceso de diseño, pues son una buena manera de soportar las comunicaciones entre las personas involucradas en el proceso. En muchos proyectos, suele ser la única documentación arquitectónica que existe. Sin embargo, si la arquitectura de un sistema debe documentarse ampliamente, entonces es mejor usar una notación con semántica bien definida para la descripción arquitectónica. No obstante, tal como se estudia en la sección 6.2, algunas personas consideran que la documentación detallada ni es útil ni vale realmente la pena el costo de su desarrollo.





# Decisiones en el diseño Arquitectónico

Durante el proceso de diseño arquitectónico, los arquitectos del sistema deben tomar algunas decisiones estructurales que afectarán profundamente el sistema y su proceso de desarrollo. Con base en su conocimiento y experiencia, deben considerar las siguientes preguntas fundamentales sobre el sistema:

1. ¿Existe alguna arquitectura de aplicación genérica que actúe como plantilla para el sistema que se está diseñando?
2. ¿Cómo se distribuirá el sistema a través de algunos núcleos o procesadores?
3. ¿Qué patrones o estilos arquitectónicos pueden usarse?
4. ¿Cuál será el enfoque fundamental usado para estructurar el sistema?
5. ¿Cómo los componentes estructurales en el sistema se separarán en subcomponentes?
6. ¿Qué estrategia se usará para controlar la operación de los componentes en el sistema?
7. ¿Cuál organización arquitectónica es mejor para entregar los requerimientos no funcionales del sistema?
8. ¿Cómo se evaluará el diseño arquitectónico?
9. ¿Cómo se documentará la arquitectura del sistema?

# Vistas arquitectónicas

Existen diferentes opiniones relativas a qué vistas se requieren. Krutchen (1995), en

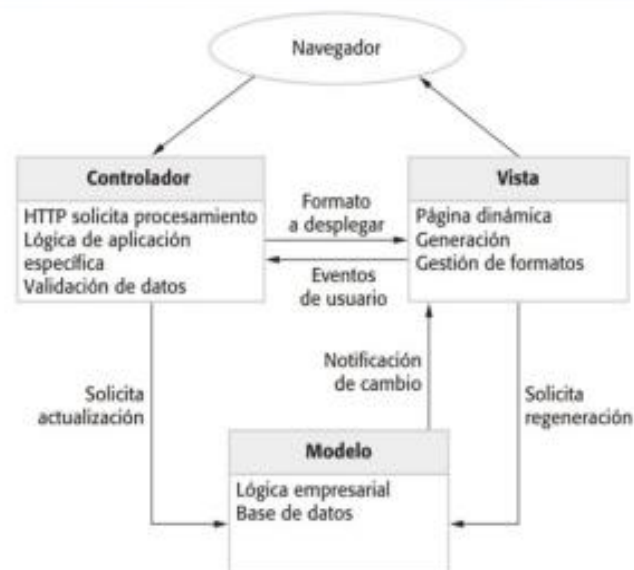
su bien conocido modelo de vista 4+1 de la arquitectura de software, sugiere que deben existir cuatro vistas arquitectónicas fundamentales, que se relacionan usando casos de uso o escenarios. Las vistas que él sugiere son:

1. Una vista lógica, que indique las abstracciones clave en el sistema como objetos o clases de objeto. En este tipo de vista se tienen que relacionar los requerimientos del sistema con entidades.
2. Una vista de proceso, que muestre cómo, en el tiempo de operación, el sistema está compuesto de procesos en interacción. Esta vista es útil para hacer juicios acerca de las características no funcionales del sistema, como el rendimiento y la disponibilidad.
3. Una vista de desarrollo, que muestre cómo el software está descompuesto para su desarrollo, esto es, indica la descomposición del software en elementos que se implementen mediante un solo desarrollador o equipo de desarrollo. Esta vista es útil para administradores y programadores de software.
4. Una vista física, que exponga el hardware del sistema y cómo los componentes de software se distribuyen a través de los procesadores en el sistema. Esta vista es útil para los ingenieros de sistemas que planean una implementación de sistema.

# Patrones arquitectónicos

Un patrón arquitectónico se puede considerar como una descripción abstracta estilizada de buena práctica, que se ensayó y puso a prueba en diferentes sistemas y entornos. De este modo, un patrón arquitectónico debe describir una organización de sistema que ha tenido éxito en sistemas previos. Debe incluir información sobre cuándo es y cuándo no es adecuado usar dicho patrón, así como sobre las fortalezas y debilidades del patrón.

En una breve sección de un capítulo general es imposible describir todos los patrones genéricos que se usan en el desarrollo de software. En cambio, se presentan algunos ejemplos seleccionados de patrones que se utilizan ampliamente y captan buenos principios de diseño arquitectónico. En las páginas Web del libro se incluyen más ejemplos acerca de patrones arquitectónicos genéricos.





## Arquitectura de capas

Este enfoque en capas soporta el desarrollo incremental de sistemas. Conforme se desarrolla una capa, algunos de los servicios proporcionados por esta capa deben quedar a disposición de los usuarios. La arquitectura también es cambiable y portátil. En tanto su interfaz no varíe, una capa puede sustituirse por otra equivalente. Más aún, cuando las interfaces de capa cambian o se agregan nuevas facilidades a una capa, sólo resulta afectada la capa adyacente. A medida que los sistemas en capas localizan dependencias de máquina en capas más internas, se facilita el ofrecimiento de implementaciones multiplataforma de un sistema de aplicación. Sólo las capas más internas dependientes de la máquina deben reimplantarse para considerar las facilidades de un sistema operativo o base de datos diferentes.

158 Capítulo 6 ■ Diseño arquitectónico

Nombre	Arquitectura en capas
Descripción	Organiza el sistema en capas con funcionalidad relacionada con cada capa. Una capa da servicios a la capa de encima, de modo que las capas de nivel inferior representan servicios núcleo que es probable se utilicen a lo largo de todo el sistema. Véase la figura 6.6.
Ejemplo	Un modelo en capas de un sistema para compartir documentos con derechos de autor se tiene en diferentes bibliotecas, como se ilustra en la figura 6.7.
Cuándo se usa	Se usa al construirse nuevas facilidades encima de los sistemas existentes; cuando el desarrollo se dispersa a través de varios equipos de trabajo, y cada uno es responsable de una capa de funcionalidad; cuando existe un requerimiento para seguridad multinivel.
Ventajas	Permite la sustitución de capas completas en tanto se conserve la interfaz. Para aumentar la confiabilidad del sistema, en cada capa pueden incluirse facilidades redundantes (por ejemplo, autenticación).
Desventajas	En la práctica, suele ser difícil ofrecer una separación limpia entre capas, y es posible que una capa de nivel superior deba interactuar directamente con capas de nivel inferior, en vez de que sea a través de la capa inmediatamente abajo de ella. El rendimiento suele ser un problema, debido a múltiples niveles de interpretación de una solicitud de servicio mientras se procesa en cada capa.



# Arquitectura de reposito

Los patrones de arquitectura en capas y MVC son ejemplos de patrones en que la vista presentada es la organización conceptual de un sistema.

a base de datos o un repositorio compartido. Por lo tanto, este modelo es adecuada para aplicaciones en las que un componente genere datos y otro los use. Los ejemplos de este tipo de sistema incluyen sistemas de comando y control, sistemas de información administrativo, sistemas CAD y entornos de desarrollo interactivo para software.

Nombre	Arquitectura en capas
Descripción	Organiza el sistema en capas con funcionalidad relacionada con cada capa. Una capa da servicios a la capa de encima, de modo que las capas de nivel inferior representan servicios núcleo que es probable se utilicen a lo largo de todo el sistema. Véase la figura 6.6.
Ejemplo	Un modelo en capas de un sistema para compartir documentos con derechos de autor se tiene en diferentes bibliotecas, como se ilustra en la figura 6.7.
Cuándo se usa	Se usa al construirse nuevas facilidades encima de los sistemas existentes; cuando el desarrollo se dispersa a través de varios equipos de trabajo, y cada uno es responsable de una capa de funcionalidad; cuando exista un requerimiento para seguridad multivél.
Ventajas	Permite la sustitución de capas completas en tanto se conserve la interfaz. Para aumentar la confiabilidad del sistema, en cada capa pueden incluirse facilidades redundantes (por ejemplo, autenticación).
Desventajas	En la práctica, suele ser difícil ofrecer una separación limpia entre capas, y es posible que una capa de nivel superior deba interactuar directamente con capas de nivel inferior, en vez de que sea a través de la capa inmediatamente abajo de ella. El rendimiento suele ser un problema, debido a múltiples niveles de interpretación de una solicitud de servicio mientras se procesa en cada capa.

# Arquitectura cliente-servidor

Un sistema que sigue el patrón cliente-servidor se organiza como un conjunto de servicios y servidores asociados, y de clientes que acceden y usan los servicios. Los principales componentes de este modelo son:

1. Un conjunto de servidores que ofrecen servicios a otros componentes. Ejemplos de estos incluyen servidores de impresión; servidores de archivo que brindan servicios de administración de archivos, y un servidor compilador, que proporciona servicios de compilación de lenguaje de programación.
2. Un conjunto de clientes que solicitan los servicios que ofrecen los servidores. Habrá usualmente varias instancias de un programa cliente que se ejecuten de manera concurrente en diferentes computadores.
3. Una red que permite a los clientes acceder a dichos servicios. La mayoría de los sistemas cliente-servidor se implementan como sistemas distribuidos, conectados mediante protocolos de Internet.

Nombre	Cliente-servidor
Descripción	En una arquitectura cliente-servidor, la funcionalidad del sistema se organiza en servicios, y cada servicio lo entrega un servidor independiente. Los clientes son usuarios de dichos servicios y para utilizarlos ingresan a los servidores.
Ejemplo	La figura 6.11 es un ejemplo de una biblioteca y videoteca (videos/DVD) organizada como un sistema cliente-servidor.
Cuándo se usa	Se usa cuando, desde varias ubicaciones, se tiene que ingresar a los datos en una base de datos compartida. Como los servidores se pueden replicar, también se usan cuando la carga de un sistema es variable.
Ventajas	La principal ventaja de este modelo es que los servidores se pueden distribuir a través de una red. La funcionalidad general (por ejemplo, un servicio de impresión) estará disponible a todos los clientes, así que no necesita implementarse en todos los servicios.
Desventajas	Cada servicio es un solo punto de falla, de modo que es susceptible a ataques de rechazo de servicio o a fallas del servidor. El rendimiento resultará impredecible porque depende de la red, así como del sistema. Quizá haya problemas administrativos cuando los servidores sean propiedad de diferentes organizaciones.

# Arquitectura de tubería y filtro

Este es un modelo de la organización en tiempo de operación de un sistema, donde las transformaciones funcionales procesan sus entradas y producen salidas. Los datos fluyen de uno a otro y se transforman conforme se desplazan a través de la secuencia. Cada paso de procesamiento se implementa como un transformador. Los datos de entrada fluyen por medio de dichos transformadores hasta que se convierten en salida. Las transformaciones pueden ejecutarse secuencialmente o en forma paralela. Es posible que los datos se procesen por cada transformador ítem por ítem o en un solo lote.

Nombre	Tubería y filtro (pipe and filter)
Descripción	El procesamiento de datos en un sistema se organiza de forma que cada componente de procesamiento (filtro) sea discreto y realice un tipo de transformación de datos. Los datos fluyen (como en una tubería) de un componente a otro para su procesamiento.
Ejemplo	La figura 6.13 es un ejemplo de un sistema de tubería y filtro usado para el procesamiento de facturas.
Cuándo se usa	Se suele utilizar en aplicaciones de procesamiento de datos (tanto basadas en lotes [batch] como en transacciones), donde las entradas se procesan en etapas separadas para generar salidas relacionadas.
Ventajas	Fácil de entender y soporta reutilización de transformación. El estilo del flujo de trabajo coincide con la estructura de muchos procesos empresariales. La evolución al agregar transformaciones es directa. Puede implementarse como un sistema secuencial o como uno concurrente.
Desventajas	El formato para la transferencia de datos debe acordarse entre las transformaciones que se comunican. Cada transformación debe analizar sus entradas y sintetizar sus salidas al formato acordado. Esto aumenta la carga del sistema, y puede significar que sea imposible reutilizar transformaciones funcionales que usen estructuras de datos incompatibles.



# Arquitecturas de aplicación

La arquitectura de aplicación puede reimplantarse cuando se desarrollen nuevos sistemas, pero, para diversos sistemas empresariales, la reutilización de aplicaciones es posible sin reimplementación. Esto se observa en el crecimiento de los sistemas de planeación de recursos empresariales (ERP, por las siglas de Enterprise Resource Planning) de compañías como SAP y Oracle, y paquetes de software vertical (COTS) para aplicaciones especializadas en diferentes áreas de negocios.

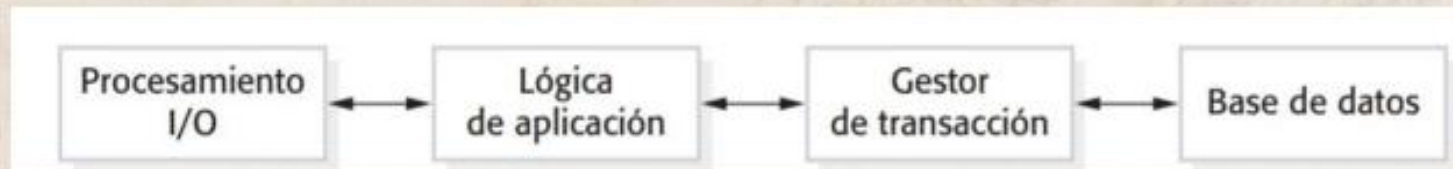
En dichos sistemas, un sistema genérico se configura y adapta para crear una aplicación empresarial específica.

Como diseñador de software, usted puede usar modelos de arquitecturas de aplicación en varias formas:

1. Como punto de partida para el proceso de diseño arquitectónico Si no está familiarizado con el tipo de aplicación que desarrolla, podría basar su diseño inicial en una arquitectura de aplicación genérica. Desde luego, ésta tendrá que ser especializada para el sistema específico que se va a desarrollar, pero es un buen comienzo para el diseño.
2. Como lista de verificación del diseño Si usted desarrolló un diseño arquitectónico para un sistema de aplicación, puede comparar éste con la arquitectura de aplicación genérica y luego, verificar que su diseño sea consistente con la arquitectura genérica.
3. Como una forma de organizar el trabajo del equipo de desarrollo Las arquitecturas de aplicación identifican características estructurales estables de las arquitecturas del sistema y, en muchos casos, es posible desarrollar éstas en paralelo. Puede asignar trabajo a los miembros del grupo para implementar diferentes componentes dentro de la arquitectura.
4. Como un medio para valorar los componentes a reutilizar Si tiene componentes por reutilizar, compare éstos con las estructuras genéricas para saber si existen componentes similares en la arquitectura de aplicación.
5. Como un vocabulario para hablar acerca de los tipos de aplicaciones Si discute acerca de una aplicación específica o trata de comparar aplicaciones del mismo tipo, entonces puede usar los conceptos identificados en la arquitectura genérica para hablar sobre las aplicaciones.

Hay muchos tipos de sistema de aplicación y, en algunos casos, parecerían muy diferentes. Sin embargo, muchas de estas aplicaciones distintas superficialmente en realidad tienen mucho en común y, por ende, suelen representarse mediante una sola arquitectura de aplicación abstracta. Esto se ilustra aquí al describir las siguientes arquitecturas de dos tipos de aplicación:

1. Aplicaciones de procesamiento de transacción Este tipo de aplicaciones son aplicaciones centradas en bases de datos, que procesan los requerimientos del usuario mediante la información y actualizan ésta en una base de datos. Se trata del tipo más común de sistemas empresariales interactivos. Se organizan de tal forma que las acciones del usuario no pueden interferir unas con otras y se mantiene la integridad de la base de datos. Esta clase de sistema incluye los sistemas bancarios interactivos, sistemas de comercio electrónico, sistemas de información y sistemas de reservaciones.
2. Sistemas de procesamiento de lenguaje Son sistemas en los que las intenciones del usuario se expresan en un lenguaje formal (como Java). El sistema de procesamiento de lenguaje elabora este lenguaje en un formato interno y después interpreta dicha representación interna. Los sistemas de procesamiento de lenguaje mejor conocidos son los compiladores, que traducen los programas en lenguaje de alto nivel dentro de un código de máquina. Sin embargo, los sistemas de procesamiento de lenguaje se usan también en la interpretación de lenguajes de comandos para bases de datos y sistemas de información, así como de lenguajes de marcado como XML (Harold y Means, 2002. Hunter et al., 2007).

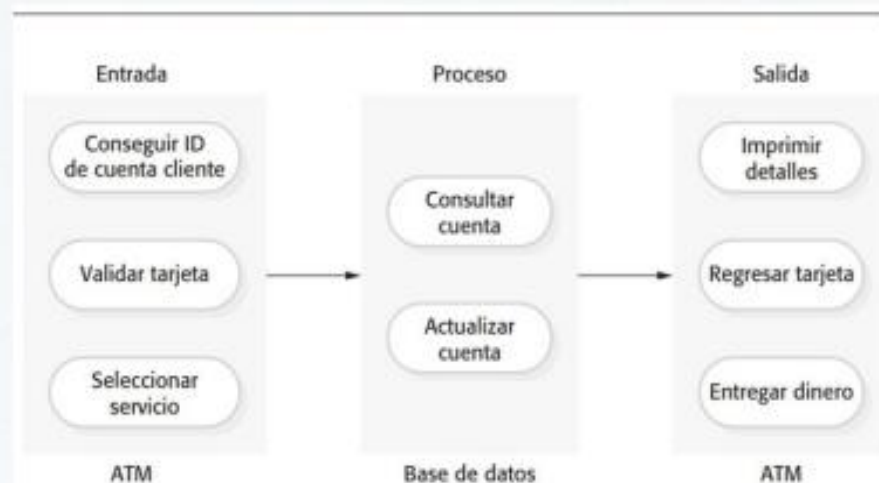




# Sistema de procesamiento de transiciones

Los sistemas de procesamiento de transacciones (TP, por las siglas de Transaction Processing) están diseñados para procesar peticiones del usuario mediante la información de una base de datos, o los requerimientos para actualizar una base de datos (Lewis et al., 2003). Técnicamente, una transacción de base de datos es una secuencia de operaciones que se trata como una sola unidad (una unidad atómica). Todas las operaciones en una transacción tienen que completarse antes de que sean permanentes los cambios en la base de datos. Esto garantiza que la falla en las operaciones dentro de la transacción no conduzca a inconsistencias en la base de datos.

Los sistemas de procesamiento de transacción pueden organizarse como una arquitectura "tubería y filtro" con componentes de sistema responsables de entradas, procesamiento y salida. Por ejemplo, considere un sistema bancario que permite a los clientes consultar sus cuentas y retirar dinero de un cajero automático. El sistema está constituido por dos componentes de software cooperadores: el software del cajero automático y el software de procesamiento de cuentas en el servidor de la base de datos del banco. Los componentes de entrada y salida se implementan como software en el cajero automático y el componente de procesamiento es parte del servidor de la base de datos del banco.





# Sistemas de información

Todos los sistemas que incluyen interacción con una base de datos compartida se consideran sistemas de información basados en transacciones. Un sistema de información permite acceso controlado a una gran base de información, tales como un catálogo de biblioteca, un horario de vuelos o los registros de pacientes en un hospital. Cada vez más, los sistemas de información son sistemas basados en la Web, cuyo acceso es mediante un navegador Web.

1. La capa superior es responsable de implementar la interfaz de usuario. En este caso, a UI se implementó con el uso de un navegador Web.

2. La segunda capa proporciona la funcionalidad de interfaz de usuario que se entrega a través del navegador Web. Incluye componentes que permiten a los usuarios ingresar al sistema, y componentes de verificación para garantizar que las operaciones utilizadas estén permitidas de acuerdo con su rol. Esta capa incluye componentes de gestión de formato y menú que presentan información a los usuarios, así como componentes de validación de datos que comprueban la consistencia de la información.

3. La tercera capa implementa la funcionalidad del sistema y ofrece componentes que componen en operación la seguridad del sistema, la creación y actualización de la información del paciente, la importación y exportación de datos del paciente desde otras bases de datos, y los generadores de reporte que elaboran informes administrativos.

4. Finalmente, la capa más baja, que se construye al usar un sistema comercial de gestión de base de datos, ofrece administración de transacciones y almacenamiento constante de datos.

La organización de servidores en dichos sistemas refleja usualmente el modelo genérico en cuatro capas presentadas en la figura 6.16. Dichos sistemas se suelen implementar como arquitecturas cliente-servidor de multinivel, como se estudia en el capítulo 18:

1. El servidor Web es responsable de todas las comunicaciones del usuario, y la interfaz de usuario se pone en función mediante un navegador Web;

2. El servidor de aplicación es responsable de implementar la lógica específica de la aplicación, así como del almacenamiento de la información y las peticiones de recuperación;

3. El servidor de la base de datos mueve la información hacia y desde la base de datos y, además, manipula la gestión de transacciones.





# Sistema de procesamiento de lenguaje

Los sistemas de procesamiento de lenguaje convierten un lenguaje natural o artificial en otra representación del lenguaje y, para lenguajes de programación, también pueden ejecutar el código resultante. En ingeniería de software, los compiladores traducen un lenguaje de programación artificial en código de máquina. Otros sistemas de procesamiento de lenguaje traducen una descripción de datos XML en comandos para consultar una base de datos o una representación XML alternativa. Los sistemas de procesamiento de lenguaje natural pueden transformar un lenguaje natural a otro, por ejemplo, francés a noruego.



Figura 6.19 Arquitectura de repositorio para un sistema de procesamiento de lenguaje

Los compiladores de lenguaje de programación que forman parte de un entorno de programación más general tienen una arquitectura genérica (figura 6.19) que incluye los siguientes componentes:

1. Un analizador léxico, que toma valores simbólicos (tokens) y los convierte en una forma interna.
2. Una tabla de símbolos, que contiene información de los nombres de las entidades (variables, de clase, de objeto, etcétera) usados en el texto que se traduce.
3. Un analizador de sintaxis, el cual verifica la sintaxis del lenguaje que se va a traducir. Emplea una gramática definida del lenguaje y construye un árbol de sintaxis.
4. Un árbol de sintaxis es una estructura interna que representa el programa a compilar.
5. Un analizador semántico que usa información del árbol de sintaxis y la tabla de símbolos, para verificar la exactitud semántica del texto en lenguaje de entrada.
6. Un generador de código que "recorre" el árbol de sintaxis y genera un código de máquina abstracto.







**Thank  
you**

The card is a simple white rectangle with a gold-colored circular fastener at the top center. It is placed over a background collage. The collage includes a piece of aged, yellowed paper with faint, handwritten text in cursive. Some legible words include 'breast', 'bowl', 'and', 'ld el', 'saw', 'wh', 'h', 'w', 'ne', 'his', 'nose', and 'And'. There are also small, dried orange flowers in the bottom left corner and a small yellow flower in the bottom right corner. The overall aesthetic is vintage and artistic.