Arquitecturas Empresariales y la plataforma J2EE

Pablo Basanta Val
Florina Almenares Mendoza
Basado en material de:
Natividad Martínez Madrid, Marisol García Valls y Simon Pickin
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
{pbasanta,florina,nati,mvalls,spickin}@it.uc3m.es

Objetivos didácticos

- Conocer qué se entiende por una aplicación empresarial basada en componentes
  - Qué requisitos la diferencian de una tradicional
  - Qué diferencias existen entre la programación basada en componentes y la tradicional basada en objetos
- Conocer las principales arquitecturas utilizadas a la hora de construir este tipo de sistemas, así como sus pros y contras
  - Cliente/Servidor, multi-capa tradicional, MVC y servidor de aplicaciones
- Comprender qué es un servidor de aplicaciones y el papel que juega dentro de J2EE
  - Funcionalidad que oferta y elementos que maneja internamente
  - Conocer la lista de servicios ofertados por el servidor de aplicaciones J2EE
  - Conocer los componentes y contenedores existente en J2EE
  - Encajar el EJB dentro de la estructura J2EE

Índice (1/2)

Índice (2/2)

Marco en el que se encuadra J2EE

- Heterogeneidad y descentralización: características de los sistemas de información actuales → distintas plataformas, sistemas operativos, protocolos de red, etc.
- Plataforma Java 2 Enterprise Edition (J2EE) (Sun Microsystems) → independencia de plataforma, portabilidad e interoperabilidad entre aplicaciones
- Objetivos básicos:
  - Abstracción de tareas críticas y repetitivas mediante *servicios con una interfaz uniforme*
  - Preparación de una *infraestructura uniforme* y de una arquitectura software basada en ella para aplicaciones empresariales

Requisitos de las aplicaciones empresariales

- ***Almacenamiento y acceso de datos (Back-end integration):*** empleo de sistemas de bases de datos (DBMS), conexión a la base de datos, representación de los datos en la base de datos
- ***Mapping de datos y persistencia*:** representación de los datos en los programas (clases) y correspondencia (mapping) con su representación en la base de datos, actualización de la base de datos tras cambios por el programa
- ***Consistencia de datos*:** control de acceso concurrente a los datos, monitores de transacción
- ***Interacción con el usuario*:** autentificación, control de acceso, coordinación de accesos concurrentes
- ***Acceso a datos comunes*:** aislamiento de los distintos accesos, cache de datos

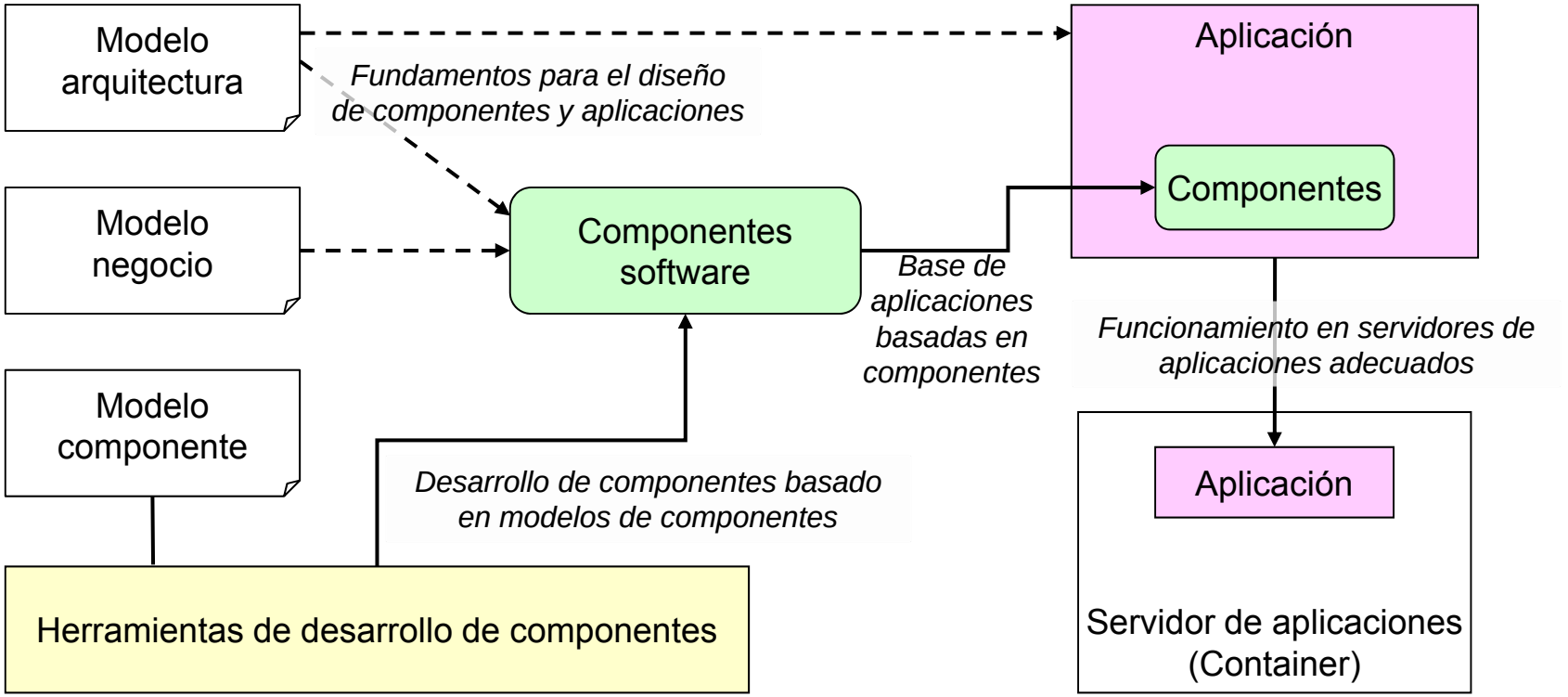
Requisitos de las aplicaciones empresariales

- ***Performance*:** tiempo de respuesta, interacción eficiente entre los distintos componentes
- ***Escalabilidad*:** posibilidad de incorporar nuevos servidores, distribución de carga
- ***Disponibilidad*:** seguridad frente a caídas de la aplicación (ideal disponibilidad 24 x 7), sistemas de tolerancia a fallos, *clustering* de servidores y datos
- ***Diseño software*:** mantenibilidad y portabilidad → modularidad, diseño en niveles, reducción de dependencias externas (por ejemplo, en la base de datos)

Programación basada en componentes vs. Programación O-O

- Objetivos comunes: ocultación de información, abstracción, bajo acoplamiento externo y alta cohesión interna
- Los objetos son de una granularidad más pequeña que los componentes
- Los componentes son elementos más asociados a la aplicación

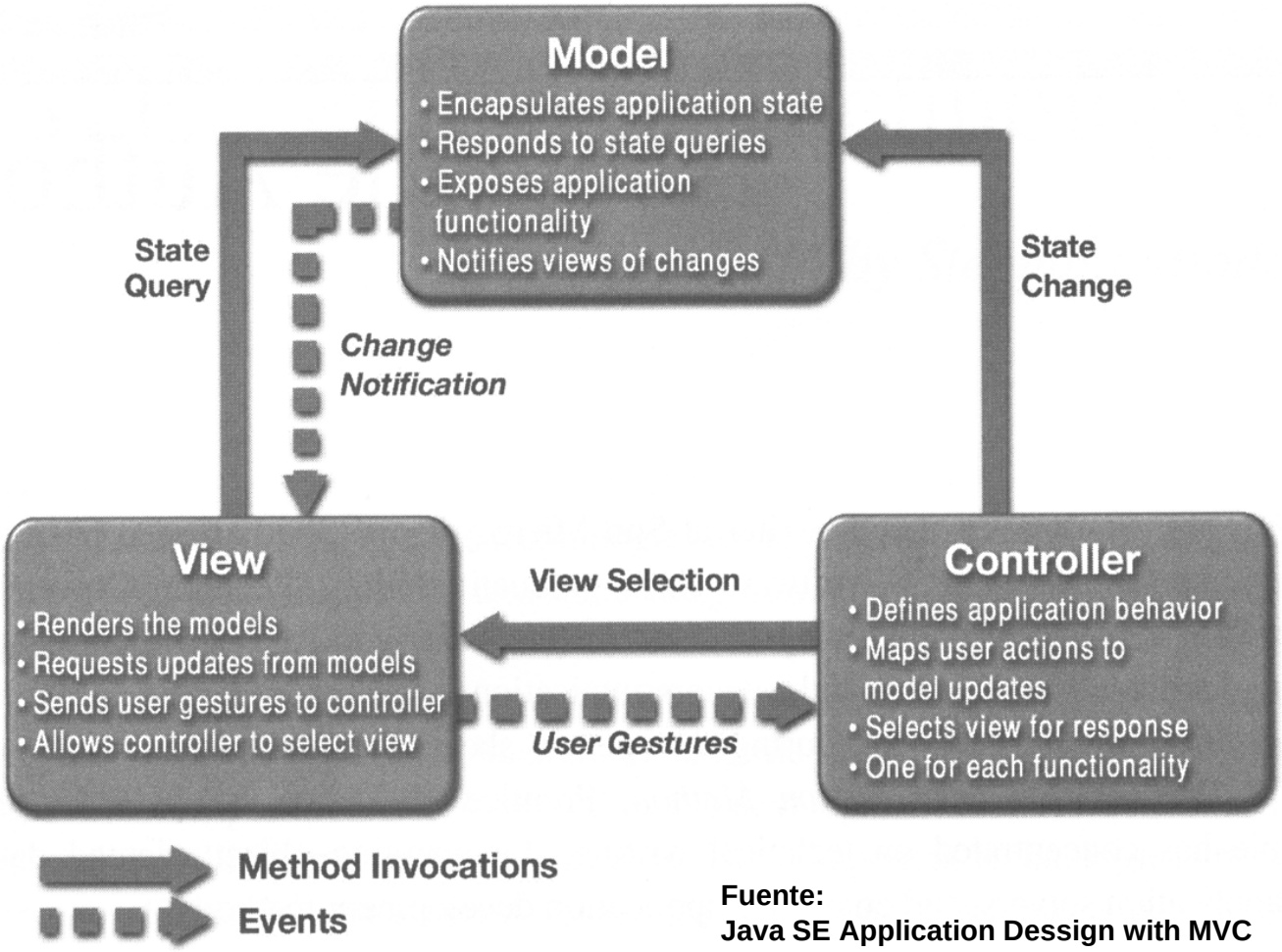
Relación entre componentes, modelos y entornos de ejecución



Arquitecturas para aplicaciones empresariales

- Las aplicaciones complejas basadas en componentes se subdividen en niveles lógicos
- Cada nivel cubre un área de tareas y puede estar compuesto de una o varias partes
- La división en niveles es una abstracción lógica

Patrón de diseño Model-View-Controller (MVC)



Fuente:
Java SE Application Dessign with MVC
http://www.oracle.com/technetwork/articles/javase/mvc-136693.html

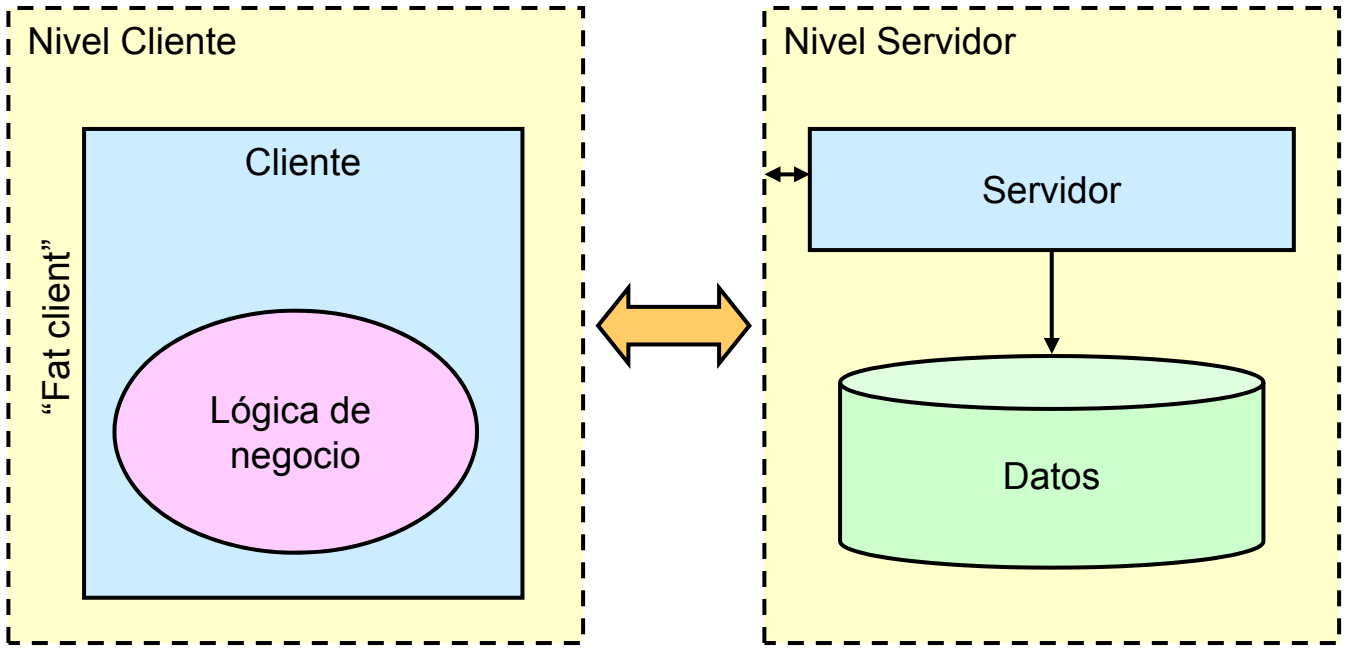
Arquitecturas cliente/servidor

- Reparto funcional y físico de la aplicación en dos niveles

1. Parte del cliente, en el ordenador del usuario:
   - Elementos de un programa clásico: lógica de ejecución, preparación y presentación de información, interacción con el usuario
2. Parte del servidor:
   - Datos de negocio (en una base de datos o, en general, en el *Enterprise Information System* (EIS))

- Cliente y servidor están débilmente acoplados y se comunican solamente mediante mensajes
- La solicitud de servicio (iniciación de la comunicación) proviene siempre del cliente. El servidor reacciona mediante una respuesta
- Gran parte de la aplicación corre en el lado del cliente (“Fat Client”) → modelo descentralizado

Arquitecturas cliente/servidor



Inconvenientes arquitectura cliente/servidor

- En general, falta de escalabilidad
- Problemas de integridad en la base de datos
- Alta carga en la red:
  - Gran número de pasos de comunicación necesarios
  - Cantidad de resultados devueltos por la base de datos mayor de lo necesario
- Costosa distribución y actualización del software (cientos o miles de clientes)
- El diseñador de la aplicación precisa un conocimiento profundo de muchas áreas críticas
  - Control de transacciones
  - Modelo de seguridad
  - Acceso a datos eficiente

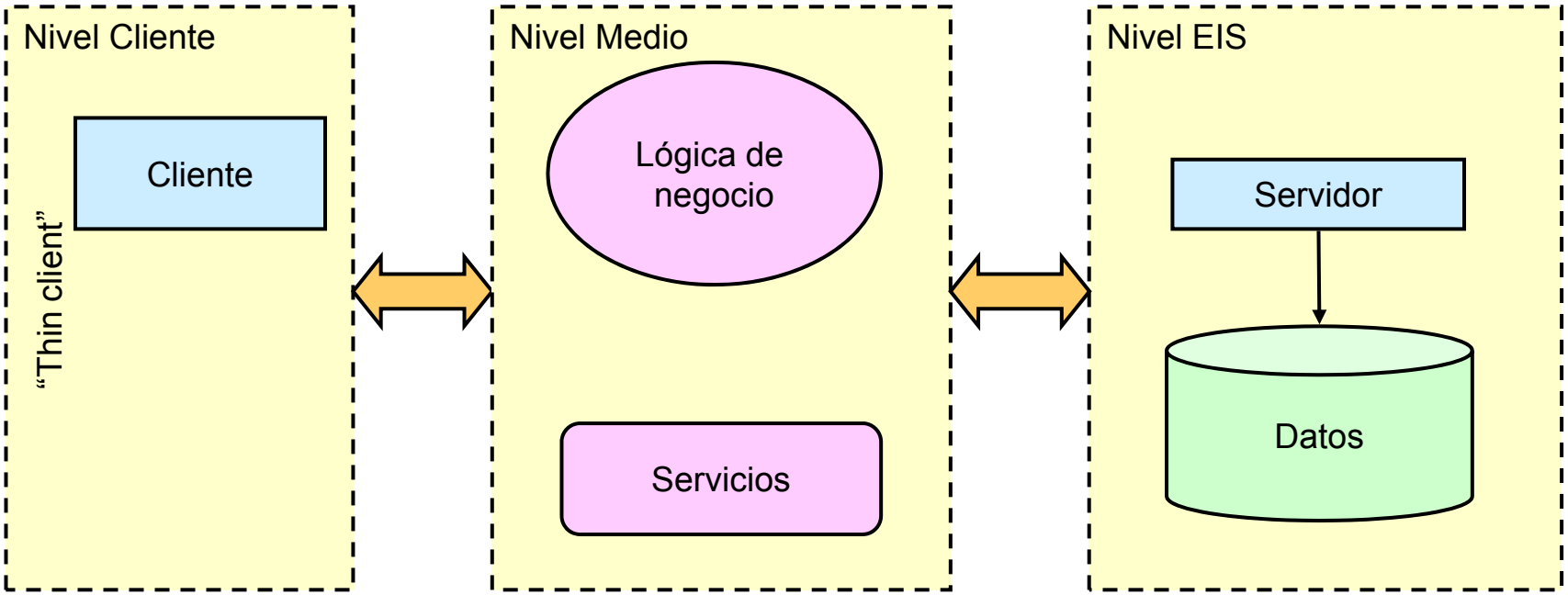
Arquitecturas multi-tier (multi-nivel)

- En las arquitecturas multi-tier, se añaden niveles (tiers) software adicionales que se encargan de realizar ciertas tareas críticas
  - Los “Fat client” se convierten en “Thin client”
- Los niveles intermedios extienden la responsabilidad del lado del servidor
  - Aunque pueden estar situados en ordenadores o sistemas independientes
- Cada nivel comunica sólo con los niveles contiguos a través de interfaces claramente definidas

Ejemplo de distribución básica multi-nivel

- Nivel cliente (*Client tier*):
  - Interfaz de usuario e interacción con el usuario (aplicaciones o applets Java, o páginas Web)
- Nivel medio (*Middle tier*):
  - Servidor de aplicaciones: parte principal de la aplicación (lógica de la aplicación y de negocio, preparación de la información para el usuario)
    - Procesado de datos basado en transacciones
    - Acceso seguro
  - Middleware: software especializado para la realización de determinadas tareas:
    - Monitores, sistemas de nombres, sistemas de colas de mensajes, etc.
  - Procesado de la presentación, por ejemplo Web-servers
- Nivel de datos (*Back-end tier*):
  - Bases de datos o Enterprise Information Systems
  - Responsable de la administración, acceso rápido y persistencia de los datos

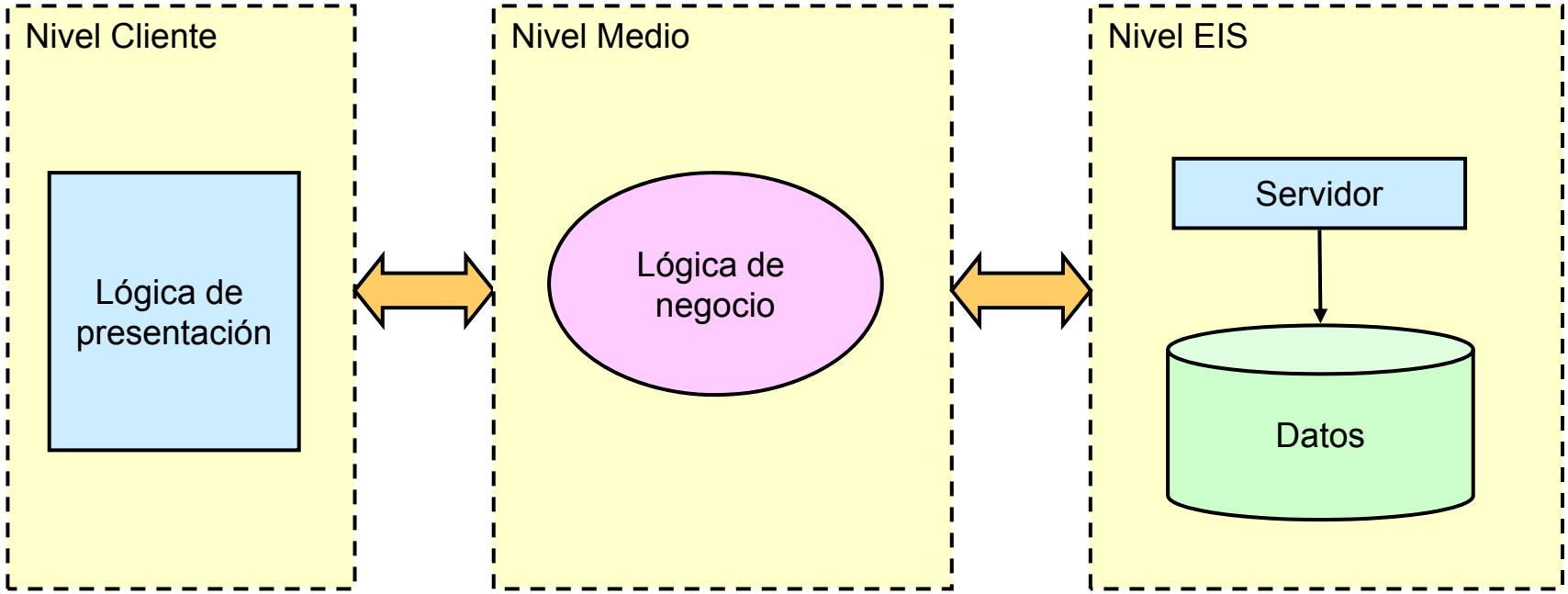
Arquitecturas multi-nivel (multi-tier)



Ventajas de las arquitecturas multi-nivel

- Las partes críticas de la aplicación se encuentran en el nivel medio, más cercanos a nivel de datos → acceso más eficiente
  - Sólo los datos necesarios son transferidos al cliente → menor carga de red
  - Problema: al aumentar el número de niveles aumenta el número de comunicaciones, aumentando el tiempo de respuesta
- Mayor flexibilidad y escalabilidad. Además:
  - Menores costes de instalación
  - Facilidad en el cambio de la base de datos
  - Aislamiento frente a cambios
  - Seguridad
  - Administración central de recursos
  - Localización de fallos

Arquitectura tradicional en tres niveles



Arquitectura tradicional en tres niveles

- El nivel medio aumenta la escalabilidad reutilizando recursos → mejora rendimiento
- Mejora la seguridad y gestión de aplicación
- Problemas:
  - Complejidad (distribución, multithreading, seguridad)
  - Falta de portabilidad (distintas APIs)
  - Soporte limitado (distintos protocolos, herramientas, vendedores)
  - Incompatibilidad con la Web

El problema de la web …

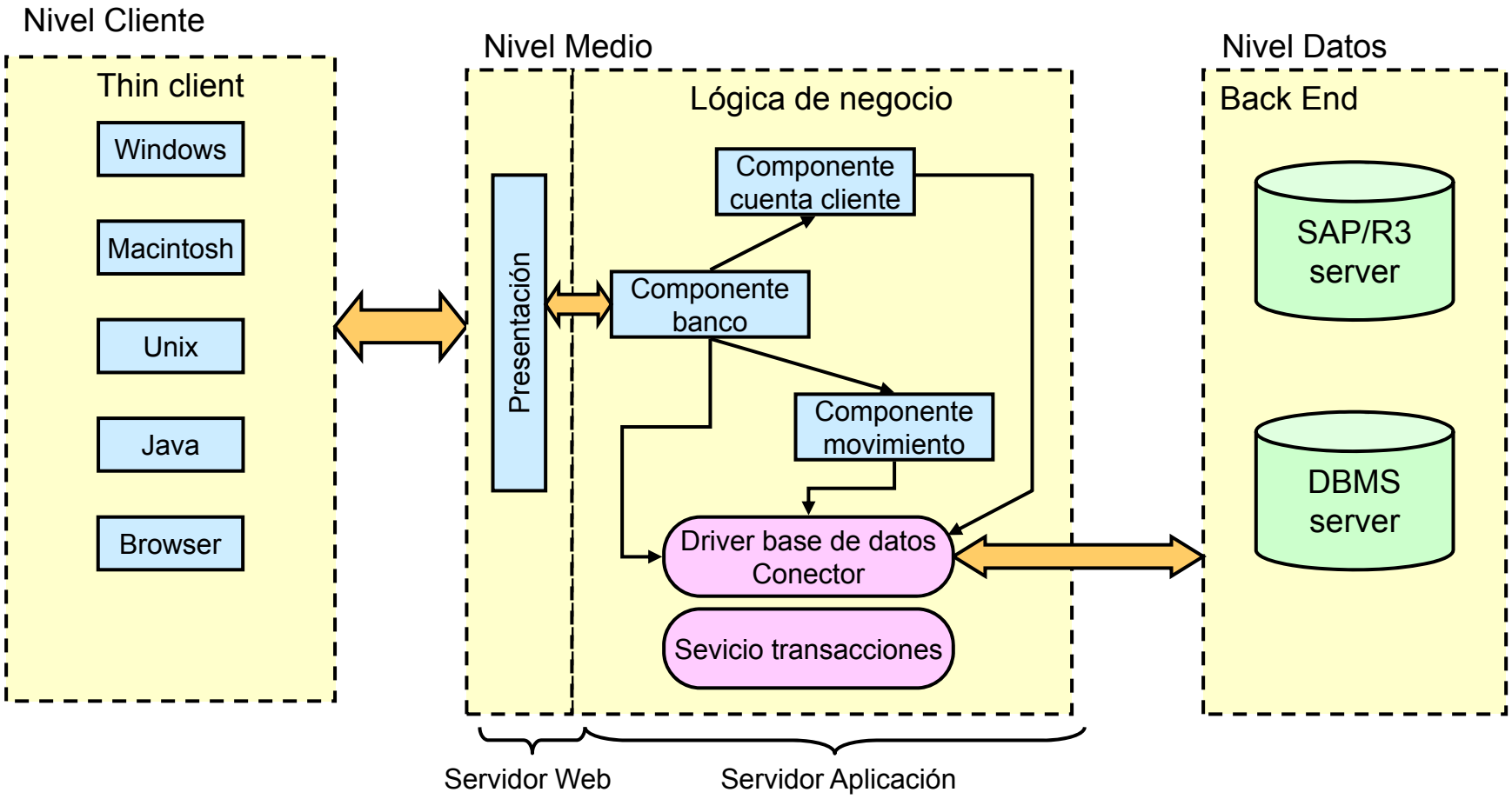
- Muchas arquitecturas han definido protocolos propietarios que son difíciles de compaginar con los requisitos de la Web.

- Aunque ha habido soluciones particulares a este problema, mediante nuevos módulos, aún persisten el resto de problemas descritos.

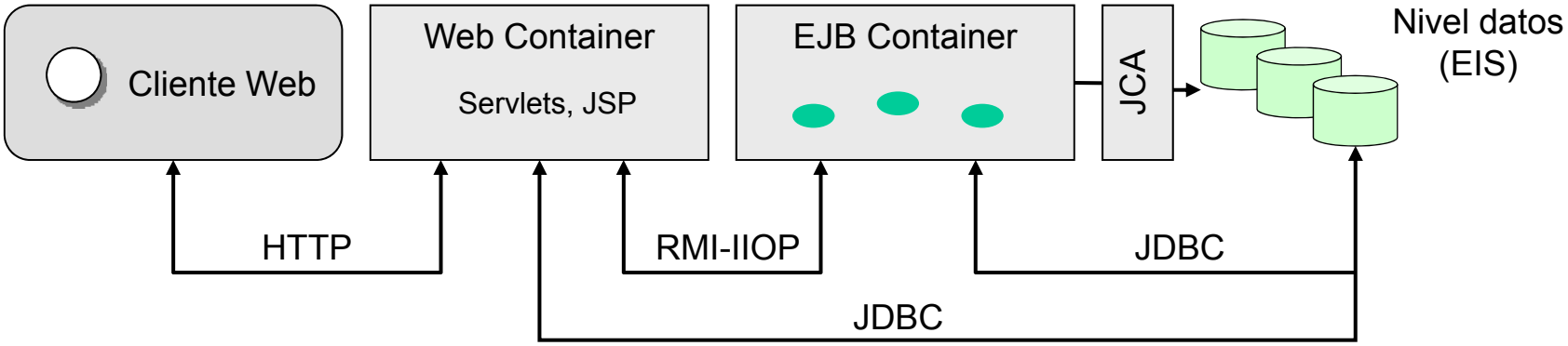
Primeras arquitecturas de aplicación Web (CGI)

- Extensiones (plug-ins) a los servidores Web que invocan aplicaciones de servidor
- Scripts CGI-BIN:
  - No proporcionan encapsulación estructurada de los procesos y entidades de negocio
  - Difíciles de desarrollar, mantener y gestionar
  - Mezclan la lógica de negocio con la lógica de presentación
  - Difícil mantenimiento de las reglas de negocio (distribuidas por varios cgi-bins en varios servidores)

Arquitectura multi-nivel con modulo Web



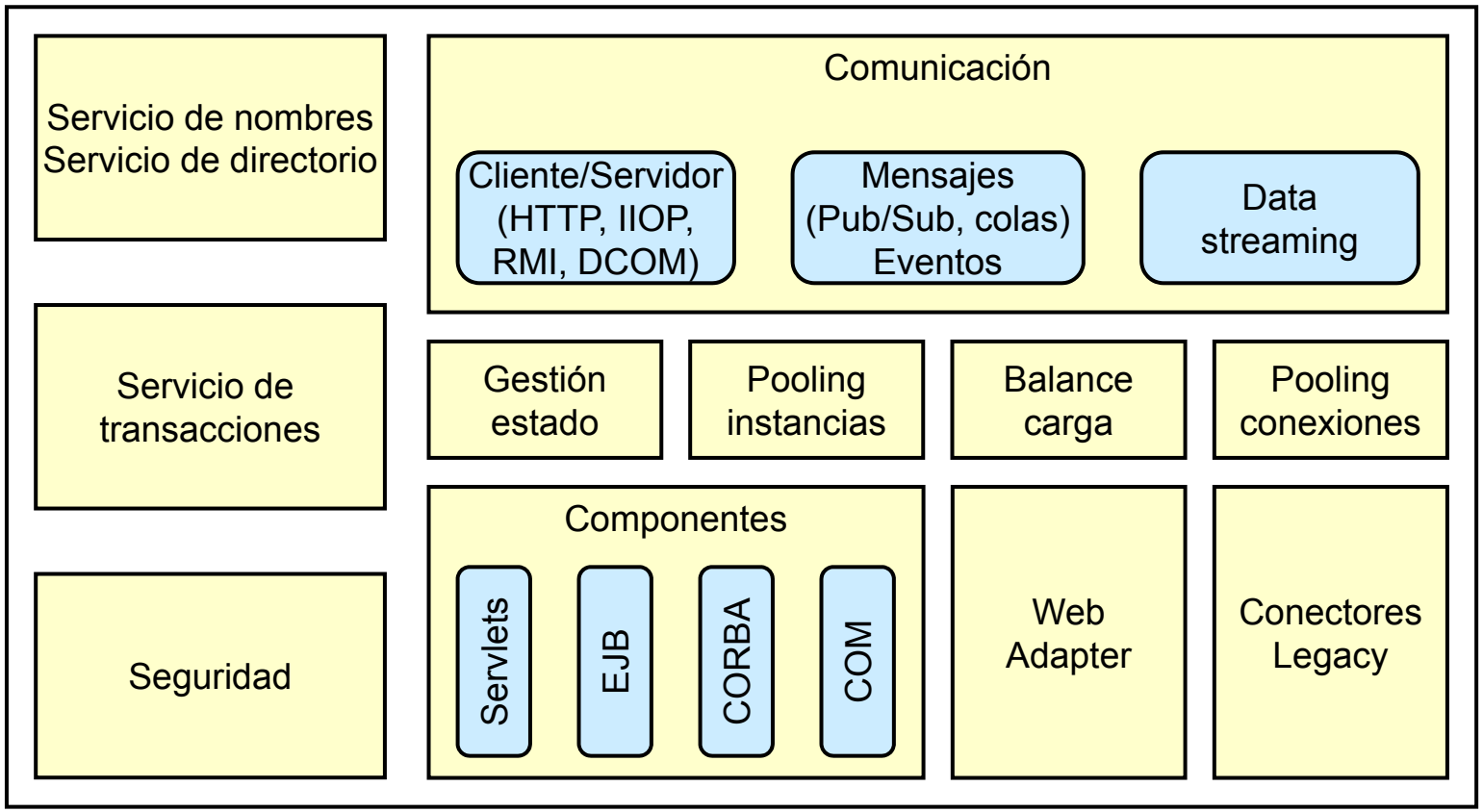
Arquitectura de aplicación Web de J2EE



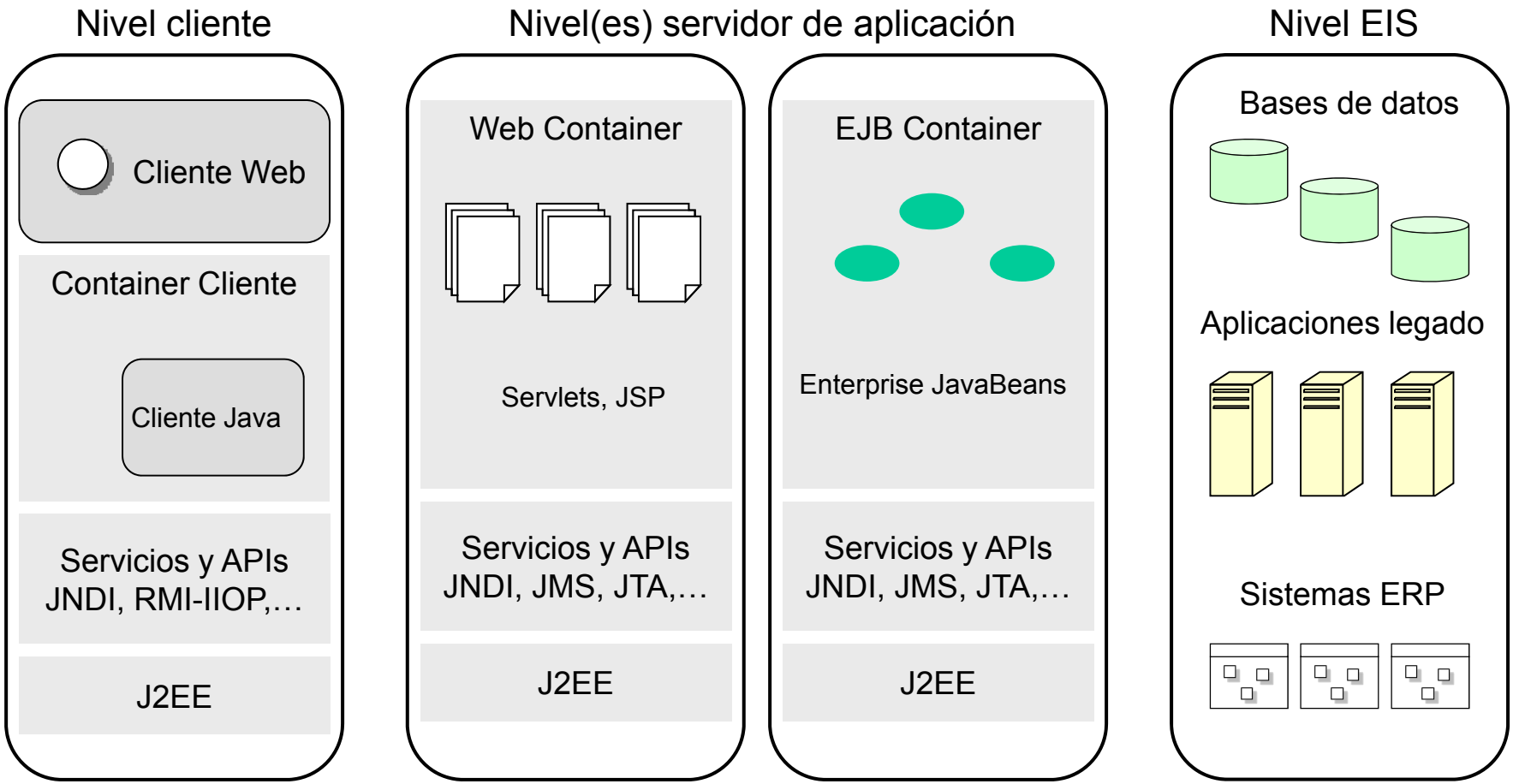
Servidor de aplicaciones: claves

- Es un sistema de soporte para componentes de servidor
  - Proporciona un entorno de desarrollo para los componentes, que a su vez proporcionan la lógica de negocio
  - Los componentes de servidor utilizan los servicios del servidor de aplicaciones
- Los elementos constitutivos del servidor de aplicaciones se denominan también componentes y pueden instalarse y administrase de forma independiente
- Tareas de infraestructura:
  - Instanciación de componentes
  - Comunicación
  - Sincronización de acceso concurrentes
  - Preparación de un entorno seguro
  - Disponibilidad
  - Seguridad de transacciones

Elementos de un servidor de aplicaciones



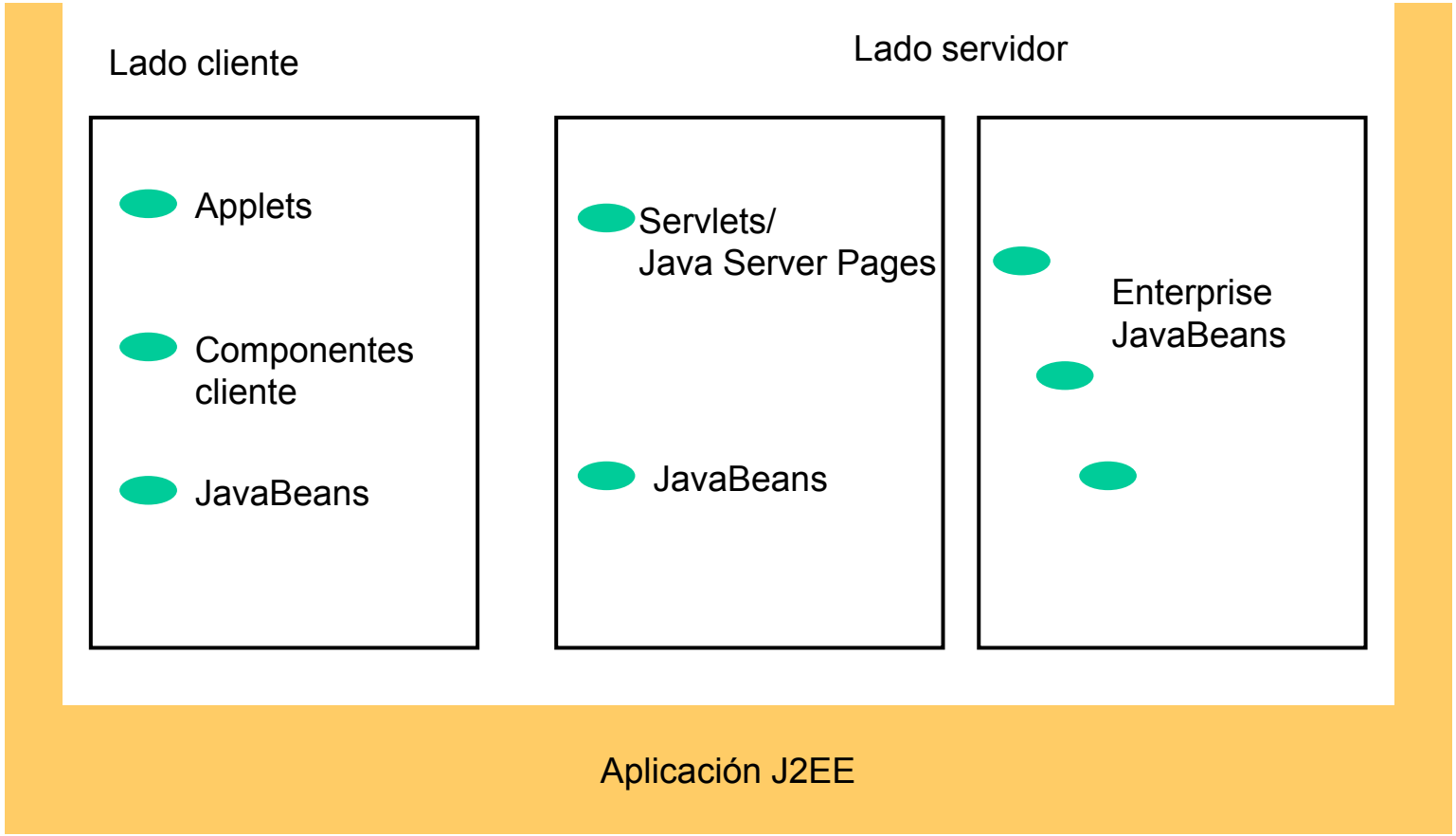
Arquitectura de aplicación J2EE



Enterprise JavaBeans

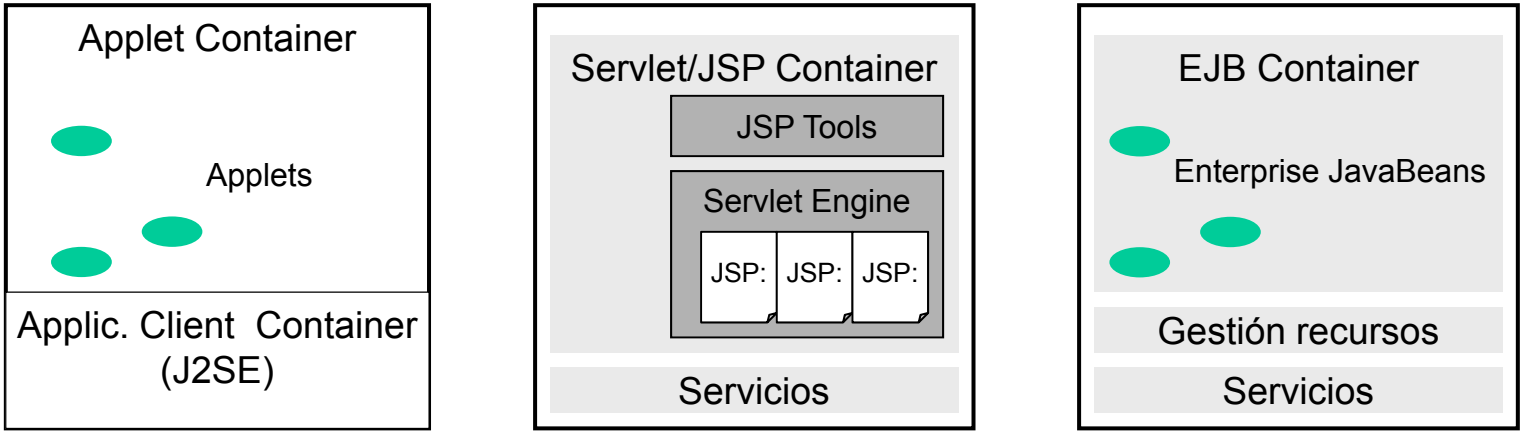
- Enterprise JavaBeans (EJB) es una completa especificación de arquitectura para componentes de servicio
- Permite el desarrollo en Java de aplicaciones multi-nivel, basadas en componentes y orientadas a transacciones, que se apoyan en servidores de aplicación y otros productos middleware
- Objetivos de la arquitectura de componentes EJB:
  - Facilitar el desarrollo de aplicaciones, concentrándose en la lógica de negocio: desarrollo, aplicación y aspectos de tiempo de ejecución
  - Independencia del proveedor de componentes mediante la especificación de interfaces
  - Independencia de la plataforma gracias al principio: “Write Once Run Anywhere” (WORA) y a su realización en Java
  - Compatibilidad con Java-APIs existentes, con sistemas de servidor de terceros y con protocolos CORBA

Tipos de componentes en J2EE



Containers (contenedores)

- Ofrecen el entorno de ejecución para todos los componentes de aplicación
- Proporcionan una vista uniforme de los servicios solicitados en la especificación
- Herramientas adicionales (Deployment Tools) para la instalación y configuración de componentes (también en tiempo de ejecución)
- Las tareas principales de los componentes del lado del servidor son la gestión de recursos y del ciclo de vida



Servicios J2EE

- **Servicio de nombres**: acceso a componentes y recursos mediante nombres lógicos
  - portabilidad y mantenibilidad
  - *Java Naming and Directory Interface (JNDI)*
- **Servicio de transacciones**: ejecución de una serie de pasos de forma atómica y aislada
  - concepto declarativo de límite de transacción mediante descriptores
  - posibilidad de control de transacción programada mediante un interfaz de programación
  - *Java Transaction Service (JTS)*
- **Servicio de seguridad**: directivas de seguridad para recursos protegidos
  - control de acceso en J2EE en dos pasos: autentificación y autorización
  - realización declarativa o programada
  - *Java Athentication & Authorization Service (JAAS)*

Servicios J2EE

- **Persistencia**: almacenamiento persistente de objetos y estados de objetos, normalmente realizado en bases de datos relacionales
  - JDBC
- **Comunicación**: distintas técnicas de comunicación, proporcionadas por el proveedor de servicio de aplicación o de containers
  - Comunicaciones Web: TCP/IP, UDP/IP, HTTP 1.0 y HTTPS (con SSL adicionalmente)
  - Procesado de objetos distribuidos: RMI (Remote Method Invocation), basado en Java Remote Method Protocol (JRMP). Extensión a RMI que soporta además el protocolo CORBA-IIOP para interoperabilidad entre J2EE y sistemas CORBA.
- **Servicios de configuración y administración**: empaquetamiento, instalación y configuración flexible de componentes y la administración de aplicaciones
  - descripción mediante esquemas XML de las características de servidores, containers, aplicaciones, componentes y servicios.

Historia de Java 2 Platform, Enterprise Edition

Colección de especificaciones y directivas de programación para facilitar el desarrollo de aplicaciones de servidor distribuidas multi-nivel, alineada fuertemente con Internet

Un poco de historia…

- **1996**: Java Development Kit (JDK) 1.02: colección ordenada de bibliotecas de clases y paquetes
- **1999**: JDK 1.2 → Java 2 Platform: adicional al JDK, paquetes opcionales para mensajes, generación dinámica de páginas Web o programas de email en Java. Dividida en 3 ediciones:
  - Java 2 Platform, Standard Edition (J2SE): contiene el JDK actual y las APIs estándar. Desarrollo de aplicaciones de Desktop y applets
  - Java 2 Platform, Enterprise Edition (J2EE): basada en J2SE, extiende el lado del servidor. Version 1.3, 3er Trimestre 2001.
  - Java 2 Platform, Micro Edition (J2ME): especial para móviles, pagers, palmtops (*embedded systems*)
- **2006**: Aparece la especificación Java EE 5.0, desarrollada bajo el JSR 244. Cambia la denominación de la tecnología: de J2EE a JEE
- **2009**: Aparece Java EE 6.0

Elementos de la especificación J2EE

- **J2EE Platform**: estándar representado por un conjunto de APIs y directivas, soportadas por un servidor de aplicación (`java.sun.com/j2ee/download.html`)
- **J2EE Blueprints**: consejos para el desarrollo de aplicaciones J2EE, patrones de diseño y un ejemplo de aplicación (`java.sun.com/blueprints/`)
- **J2EE Server**: implementación de referencia de un servidor de aplicaciones para J2EE, incluido en J2EE SDK (`java.sun.com/j2ee/download.html`)
- **J2EE Testsuite**: J2EE Compatibility Testsuite (CTS), tests de compatibilidad (`java.sun.com/j2ee/compatibility.html`)

Cuestiones para reflexionar – Bloque I

- Las aplicaciones empresariales están más cerca de nosotros de lo que podemos pensar, ¿podría citar un ejemplo de tres aplicaciones empresariales que maneje a diario?
- De los ocho requisitos típicos de una aplicación empresarial, seleccione desde su punto de vista, aquel más característico.
- ¿Quién posee mayor grado de granularidad, un componente o una clase?
- ¿Qué ventaja aporta la programación basada en componentes al desarrollo de aplicaciones empresariales?

Cuestiones para reflexionar – Bloque II

- ¿Cuál es la principal ventaja que aportan las aplicaciones basadas en múltiples niveles lógicos a una aplicación empresarial?
- En la arquitectura MVC, indique de entre sus cinco tipos de comunicaciones, aquellas (tres en total) que son realizadas mediante invocaciones a métodos.
- En la arquitectura cliente/servidor, ¿dónde se coloca la lógica de negocio, en el cliente o en el servidor?
- En una arquitectura multinivel, ¿dónde se coloca la lógica de negocio?
- Cite dos ventajas que presenta la arquitectura multinivel frente a la cliente servidor.

Cuestiones para reflexionar – Bloque III

- Defina con sus propias palabras que es un servidor de aplicaciones y cuales son las tareas que tiene encomendadas
- De los 11 elementos constitutivos de un servidor de aplicaciones, cite tres servicios casi altamente relevantes
- ¿Cual es el significado de las siglas EJB? ¿De qué tipo de lógica se ocupan?
- De los 6 diferentes tipos de componentes presentes en J2EE, cite los tres que se alojan en la parte del servidor
- Enumere los 4 tipos de contenedores que existen en J2EE