JAVA 2 ENTERPRISE EDITION

Jon Castro
Jonathan Escolano

Índice

- Arquitecturas características de las aplicaciones empresariales
- Tecnologías J2EE
- Alternativas a J2EE
- Tecnologías de integración de aplicaciones
- Patrones arquitectónicos Model-View-Controller y Layers

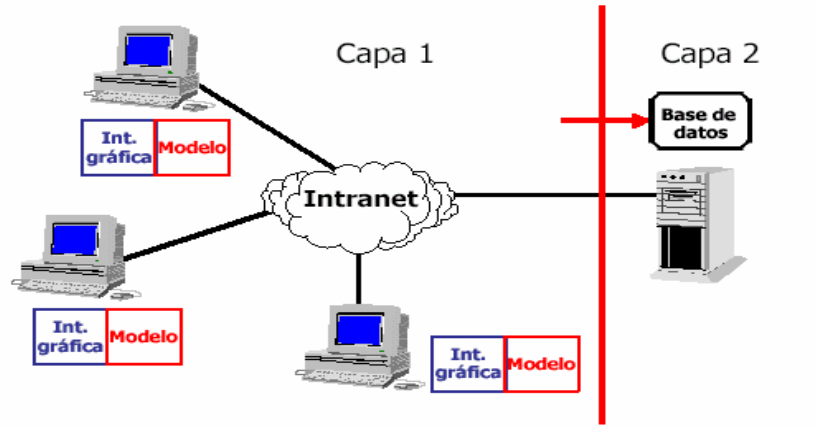
Características de las aplicaciones empresariales(1)

- Acceso a bases de datos (BD)
  - Normalmente con BD relacionales
- Transaccionales
  - Propiedades ACID (Atomicity-Consistency-Isolation-Durability)
- Escalables
  - Deberían poder soportar más carga de trabajo sin necesidad de modificar el software (sólo añadir más máquinas)
- Disponibilidad
  - Idealmente no deben dejar de prestar servicio
- Seguras
  - No todos los usuarios pueden acceder a la misma funcionalidad
- Integración
  - Es preciso integrar aplicaciones construidas con distintas tecnologías

Características de las aplicaciones empresariales (y 2)

- Tipo de interfaz
  - De entorno de ventanas (clientes standalone): normalmente sólo tiene sentido en intranets
  - Web: En Internet y en intranets
- Separación clara entre la interfaz gráfica y el modelo
  - Modelo: encapsula la lógica de negocio
  - Ejemplo => aplicación bancaria
    - Modelo: conjunto de clases que nos permiten: crear cuentas, destruirlas, encontrarlas por distintos criterios, hacer transferencias bancarias, etc.
  - El modelo debería ser reusable con distintas interfaces gráficas
    - En el ejemplo de la aplicación bancaria podría haber dos clientes: uno web y otro standalone
- Arquitecturas multi-capa

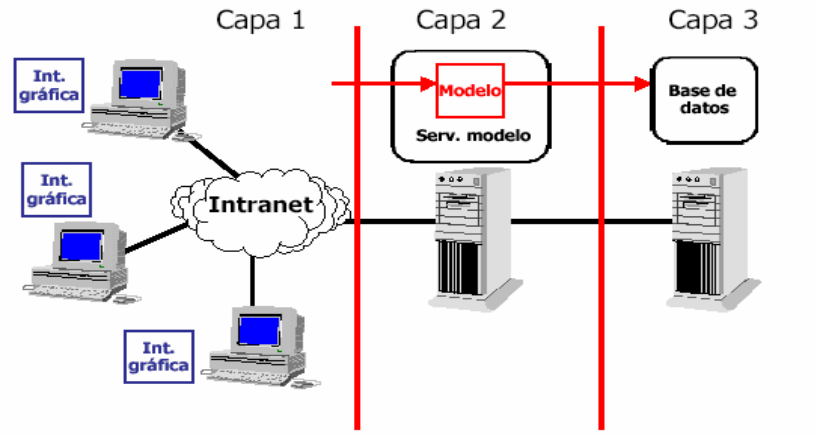
Una aplicación con clientes standalone
Arquitectura en dos capas (1)



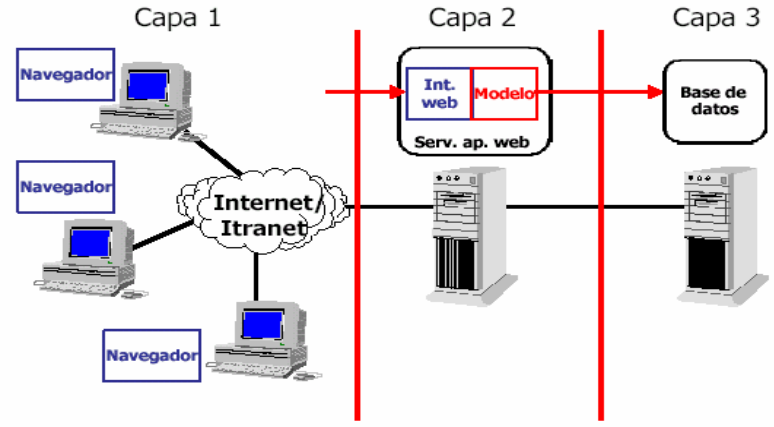
Una aplicación con clientes standalone
Arquitectura en dos capas (y 2)

- **Problema**
  - Cambios en la implementación de la capa modelo => recompilación de toda la aplicación y reinstalación en clientes
    - Cambios de drivers de acceso a la BD
    - Cambios en la lógica del modelo
    - Cambio de tipo de BD
- **Solución**
  - Modelo en servidor intermedio
    - Un cambio en la implementación del modelo sólo afecta al servidor
  - Clientes standalone
    - Sólo disponen de la interfaz gráfica
    - Acceden al servidor que implementa el modelo

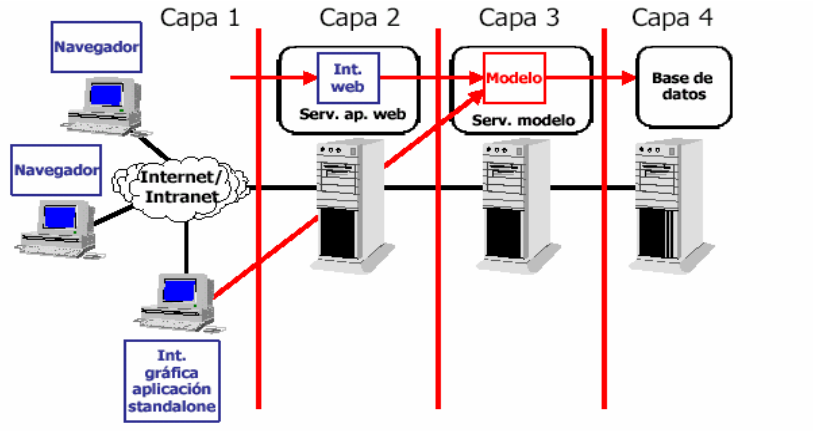
Una aplicación con clientes standalone
Arquitectura en tres capas



Una aplicación puramente web
Arquitectura en tres capas



Una aplicación con distintos tipos de clientes Arquitectura en 4 capas



Comentarios acerca de las anteriores arquitecturas

- ¿ Cómo conseguir escalabiliad y disponibilidad ?
  - Replicando los servidores de aplicaciones web y del model en máquinas distintas
- Un comentarios sobre las aplicaciones puramente web
  - La arquitectura en 3 capas es la más usada
  - La arquitectura en 4 capas puede ser más escalable

¿ Qué es J2EE ?

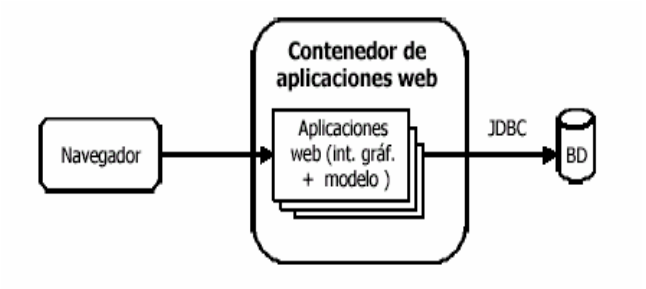
- J2EE es un conjunto de **especificaciones** de APIs Java para la construcción de aplicaciones empresariales
  - La mayor parte de las abstracciones de las APIs corresponden a interfaces y clases abstractas
  - Existen múltiples implementaciones de distintos fabricantes, incluso algunas OpenSource
  - Una aplicación construida con J2EE no depende de una implementación particular
  - Sitio central: http://java.sun.com/j2ee
- Es necesario distinguir entre
  - J2ME (Java 2 Platform, Micro Edition)
    - Para dispositivos (ej.: PDAs)
  - J2SE (Java 2 Platform, Standard Edition)
    - Para aplicaciones y applets
  - J2EE (Java 2 Platform, Enterprise Edition)
    - Se apoya en J2SE
    - Con el paso del tiempo, algunas APIs de J2EE se pasaron (y quizás se sigan pasando) a J2SE

Principales tecnologías proporcionadas por J2EE (1)

- JDBC (J2SE)
  - API para acceso a bases de datos relacionales
  - El programador puede lanzar queries (consulta,actualización, inserción y borrado), agrupar queries en transacciones, etc.

Principales tecnologías proporcionadas por J2EE (2)

- Tecnologías web (J2EE)
  - APIs: Servlets, páginas JSP y JSTL
  - Permiten implementar la interfaz gráfica de una aplicación web

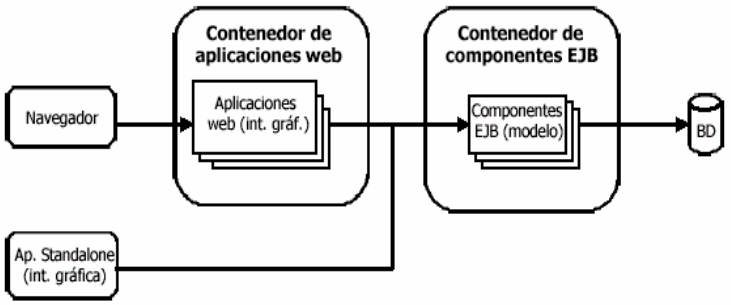


Principales tecnologías proporcionadas por J2EE (3)

- Componentes EJB (J2EE)
  - Entity Beans
    - Permiten implementar fácilmente los objetos persistentes del modelo
    - Representan una alternativa a JDBC (idealmente), permitiendo construir una capa modelo que no depende de un tipo particular de BD (relacional, objetual)
  - Session Beans
    - Permiten implementar fachadas del modelo
    - Se definen con interfaz remota si interesa separación física entre interfaz gráfica y modelo (solución más reusable)
    - Se definen con interfaz local en otro caso
  - Permiten especificar las operaciones que son transaccionales, así como las que requieren seguridad, de forma **declarativa**
    - Facilidad de desarrollo

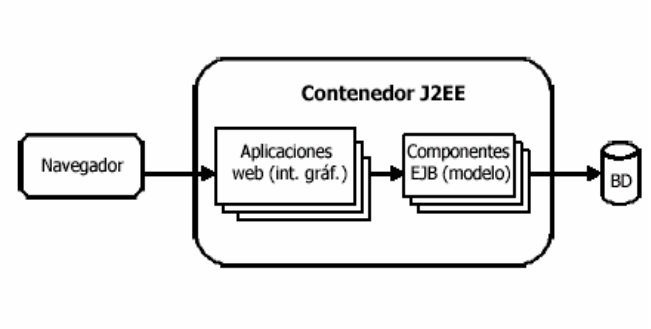
Principales tecnologías proporcionadas por J2EE (4)

- Componentes EJB (cont)



Principales tecnologías proporcionadas por J2EE (5)

- Componentes EJB (cont)



Principales tecnologías proporcionadas por J2EE (6)

APIs para XML

XML (http://www.w3c.org)

Lenguaje de tags (similar en sintaxis a HTML)

Es extensible (no dispone de tags predefinidos)

Permite expresar datos y no aspecto visual (a diferencia de HTML)

Ejemplo

```
<?xml version="1.0">
<forecasts>
<city name="COR">
<forecast type="sunny" day="1" month="10" year="2001"/>
<forecast type="foggy" day="2" month="10" year="2001"/>
</city>
<city name="LUG">
<forecast type="rainy" day="1" month="10" year="2001"/>
<forecast type="rainy" day="2" month="10" year="2001"/>
</city>
...
</forecasts>
```

Principales tecnologías proporcionadas por J2EE (y 7)

- APIs para XML (cont)
  - Campos de aplicación
    - Intercambio de datos entre aplicaciones heterogéneas
    - Configuración de aplicaciones
    - Generación de aspecto visual (ej.: HTML) a partir de los datos
    - Bases de datos
    - ... y muchos otros …
  - JAXP (J2SE)
    - API Java para procesamiento de documentos XML
- APIs para integración de aplicaciones heterogéneas
  - CORBA (API básica en J2SE) y Servicios Web (J2EE)

Implementaciones de J2EE (1)

- Existen un gran número de fabricantes que venden **servidores de aplicaciones certificados J2EE**
  - Lista completa en http://java.sun.com/j2ee/compatibility.html
  - Algunos ejemplos
    - BEA WebLogic Server: http://www.bea.com
    - Inprise/Borland AppServer: http://www.inprise.com
    - IBM WebSphere ApplicationServer: http://www.ibm.com
    - IONA iPortal Application Server: http://www.iona.com
    - Sun ONE Application Server: http://www.sun.com
    - Macromedia JRun Server: http://www.macromedia.com
    - Oracle Application Server: http://www.oracle.com
    - Sun Java 2 SDK Enterprise Edition: http://java.sun.com/j2ee/download.html
      - ¡ Es la implementación de referencia y no es eficiente !
      - Es especialmente útil para los fabricantes de servidores J2EE

Implementaciones de J2EE (y 2)

- Implementaciones OpenSource
  - Tomcat (subproyecto de Jakarta): http://jakarta.apache.org/tomcat
    - Contenedor de aplicaciones web
  - JBoss: http://www.jboss.org
  - Evidan JOnAS: http://www.evidian.com/jonas
  - OpenEJB: http://openejb.sourceforge.net
- Portabilidad
  - Si una aplicación sólo usa las APIs estándares => es posible instalarla sobre cualquier servidor de aplicaciones conforme a J2EE
  - ¡ No se depende de un fabricante !

Alternativas a J2EE (1)

- .NET
  - http://www.microsoft.com/net
  - Define un Common Language Runtime (CLR) y un IL (Intermediate Language) al que todos los lenguajes conformes a .NET compilan
    - Idea similar a la máquina virtual de Java y a los bytecodes generados por el compilador de Java, respectivamente
  - Lenguajes
    - Visual Basic .NET, Visual C++ .NET, Visual C# .NET, Visual J# .NET, etc.
  - Tecnologías
    - ADO.NET, ASP.NET, COM+: similares en concepto a JDBC, JSP y EJB, respectivamente
    - Son una mejora de sus versiones anteriores (ADO, ASP, COM, etc.)
    - APIs para XML
  - ¡ Un solo fabricante !
    - Pero es Microsoft ...
  - Menos maduro que J2EE

Alternativas a J2EE (y 2)

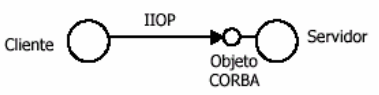
- LAMP
  - http://www.onlamp.com
  - Linux + Apache + MySQL +Perl/PHP/Python
  - Perl/PHP/Python
    - Lenguajes tipo Script
    - Acceso a base de datos
    - Tecnologías web
    - Soporte para XML
  - Requiere menos conocimientos técnicos que J2EE o .NET
  - ¿ Y la calidad del software ?

Tecnologías de integración de aplicaciones (1)

- ¿ Cómo podemos interconectar dos aplicaciones construidas con distintas tecnologías ?
  - Una aplicación Java que quiere acceder a un servidor C++
  - Una aplicación .NET que quiere acceder a una aplicación Java
- Tecnologías de integración de aplicaciones
  - CORBA
  - Servicios Web

Tecnologías de integración de aplicaciones (2)

- CORBA
  - Tecnología de objetos distribuidos que permite la invocación de métodos de objetos remotos (como si fuesen objetos locales) sin que importe la tecnología que usen cliente y servidor
    - En realidad, el cliente usa un Proxy del objeto remoto
  - Protocolo de comunicación: IIOP
    - Binario



  - Estandarizado por OMG (http://www.omg.org)
    - Éxito comercial en 1995
  - El OMG ha estandarizado numerosos servicios CORBA
    - Nombres, Seguridad, Transacciones, Eventos, etc.
  - Existen múltiples implementaciones comerciales y OpenSource, disponibles para los lenguajes y sistemas operativos más usuales

Tecnologías de integración de aplicaciones (3)

- CORBA (cont)
  - CORBA ha sido y continúa siendo una buena tecnología para abordar integraciones complejas en intranets
  - Sin embargo, no ha tenido éxito para integración de aplicaciones en Internet
    - Existen firewalls que no reconocen IIOP
      - Hay fabricantes que venden proxies de IIOP, pero no se puede esperar que todas las empresas que han adoptado las tecnologías de Microsoft los compren
    - Microsoft no fabrica implementaciones de CORBA
      - Hay terceros que sí lo hacen (ej.: Iona, Inprise, etc.), pero no se puede esperar que todas las empresas que han adoptado las tecnologías de Microsoft usen CORBA
  - Para abordar integraciones de aplicaciones en Internet es preciso usar una tecnología que cuente con el apoyo de todos los fabricantes de tecnología (Sun, Oracle, IBM, Microsoft, etc.)

Tecnologías de integración de aplicaciones (y 4)

- Servicios Web
  - Conjunto de tecnologías que usan XML para intercambio de información en un entorno distribuido
    - Éxito comercial en 2000/2001
  - Protocolo de comunicación: SOAP
    - Estandarizado por W3C (http://www.w3c.org)
    - Protocolo basado en XML para el intercambio de información
    - Conceptualmente permite enviar peticiones/respuestas en XML (normalmente sobre HTTP)
  - Existen APIs, para los lenguajes más usuales
    - Disponible para J2EE, .NET y LAMP
  - Buena solución para integración de aplicaciones en Internet
    - Todos los firewalls reconocen HTTP
    - Todos los fabricantes de tecnología proporcionan soporte para Servicios Web
  - Las integraciones complejas en intranets suelen requerir funcionalidad que todavía no soportan los Servicios Web
    - Ej.: Transacciones

Patrones arquitectónicos Model-View-Controller y Layers (1)

- ¿ Cómo se debe diseñar una aplicación empresarial para que sea **mantenible** y contenga **partes reusables** ?
  - Debería estar diseñada siguiendo la arquitectura que fijan los **patrones arquitectónicos** Model-View-Controller (MVC) y Layers
    - F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Software Architecture: A System Of Patterns, John Wiley and Sons, 1996.
  - Un patrón arquitectónico es un patrón de alto nivel que fija la arquitectura global de una aplicación
  - Posteriormente, el diseño hará uso de patrones de diseño para resolver problemas específicos

Patrones arquitectónicos Model-View-Controller y Layers (2)

- Patrón arquitectónico MVC
  - Separación clara entre el modelo (lógica de negocio) y la vista (interfaz gráfica), gracias a un controlador que los mantiene desacoplados
  - Ventajas:
    - El modelo es reusable con distintas vistas (ej.: una vista web y una con interfaz de ventanas)
    - División clara de trabajo entre los miembros de un equipo, que estará formado por personas con distintos niveles de especialización
- Patrón arquitectónico Layers
  - El software está estructurado en capas
  - Permite ocultar las tecnologías que usa nuestro software
    - Cuando hay un cambio de versión en una de ellas (o incluso se reemplaza por otra distinta), no tiene impacto sobre las capas superiores
    - División clara de trabajo entre los miembros de un equipo
  - Dará soporte a la arquitectura MVC
    - Ej.: tanto la vista como el controlador nunca conocerán las tecnologías que usa la implementación del modelo

Patrones arquitectónicos Model-View-Controller y Layers (y 3)

- En esta asignatura nos concentraremos en J2EE y aprenderemos a diseñar aplicaciones empresariales con
  - Arquitectura {MVC + Layers} + múltiples patrones de diseño
    - Fuentes
      - Core J2EE Patterns
      - EJB Design Patterns
      - Java BluePrints (http://java.sun.com/blueprints)
    - La descripción de los anteriores patrones usa EJB, sin embargo gran parte de ellos son aplicables cuando se usa JDBC
  - Dos grandes ejemplos: MiniBank y MiniPortal,
    - Iremos viendo el diseño del modelo, vista y controlador a medida que avancemos
    - Proporcionan varias versiones de la implementación de la capa modelo (con JDBC y con EJB), sin que ello afecte a la vista y al controlador