

# Ciclo de Ingeniería de Software

Desarrollo Iterativo de Software  
Aplicaciones Cliente Servidor  
Aplicaciones OO

Universidad FASTA  
2008



# Aplicaciones Cliente Servidor

- Contenido
  - Introducción
  - Conceptos
  - Diseño
  - Desarrollo
  - Modelos
  - Debate casos reales

# Aplicaciones Cliente Servidor

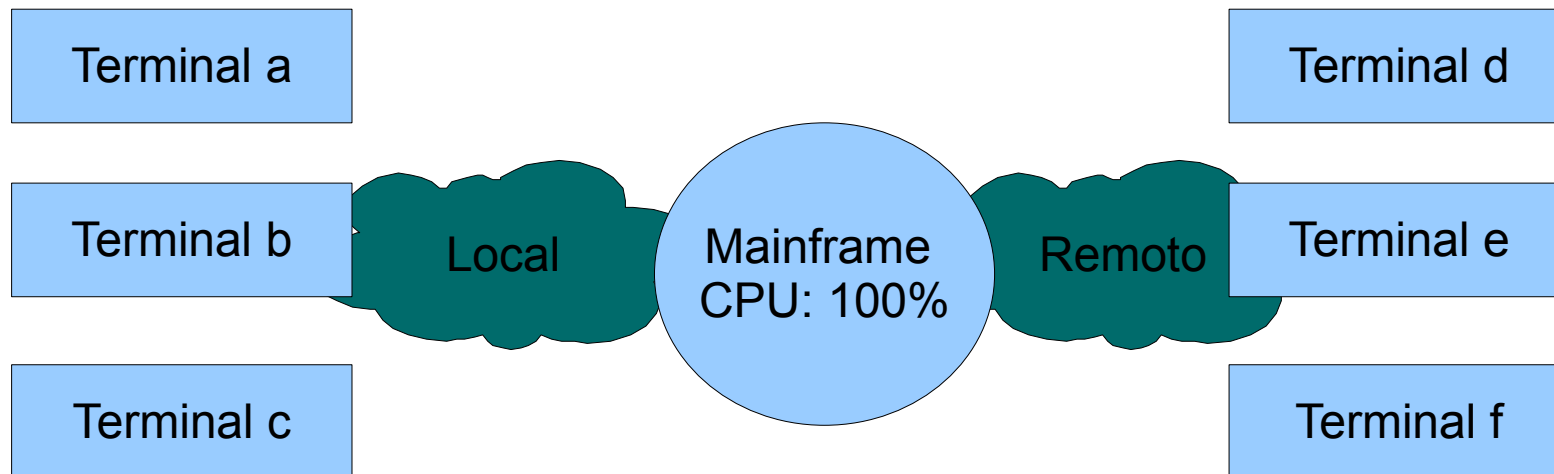
- Introducción

- Historia

- En los 60s terminales de caracteres.
    - En los 70s aplicaciones interactivas y transaccionales.
    - El término C/S se empezó a usar en la década de 1980 luego de la aparición de las Pcs y las primeras redes.
    - Gran esfuerzo en middleware
    - Se terminó de aceptar a fines de los 80s.

# Aplicaciones Cliente Servidor

- Introducción
  - Arquitectura mainframe

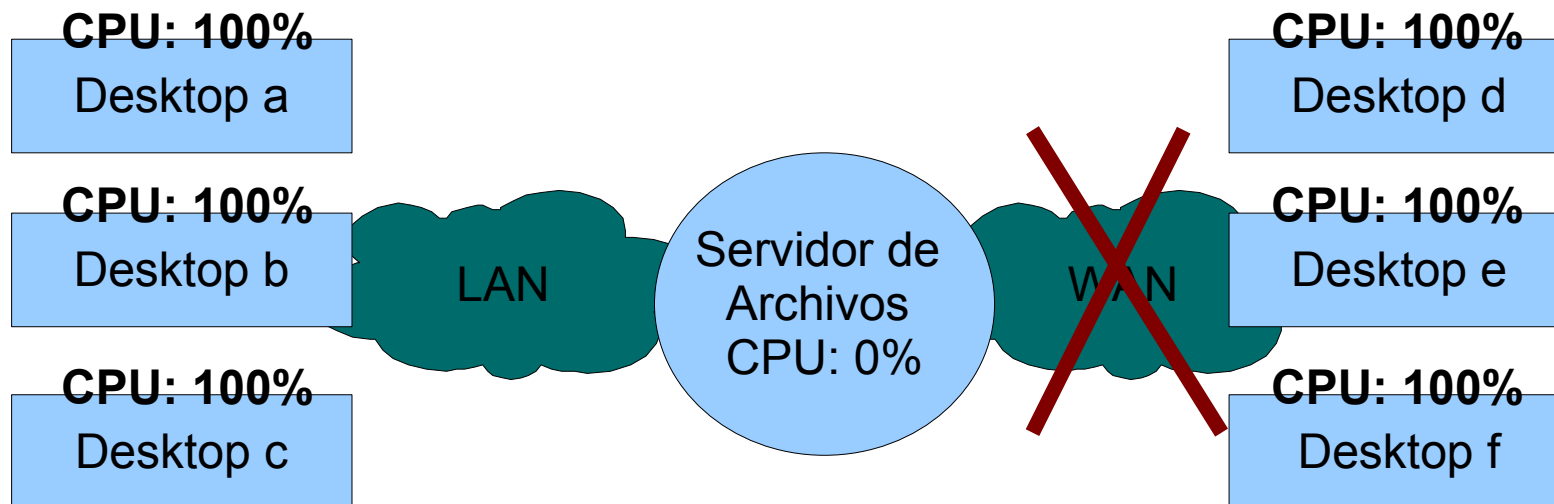


- Multiusuario
- Local/remoto

# Aplicaciones Cliente Servidor

- Introducción

- Arquitectura File Sharing



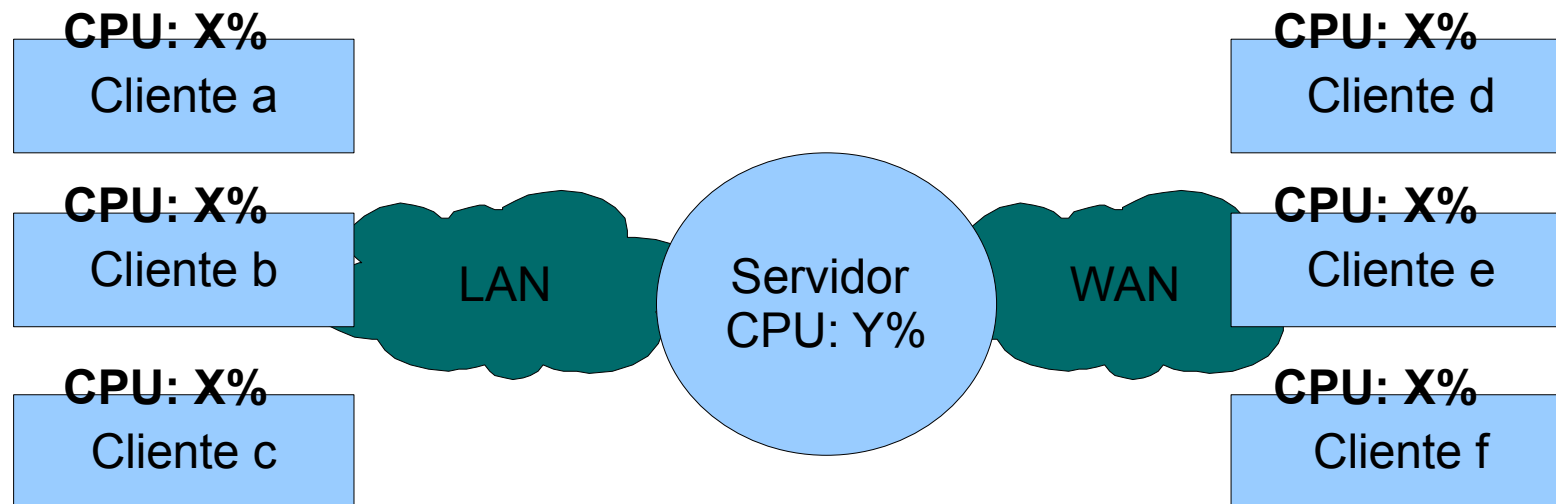
- Multiusuario (+/-)

- LAN

# Aplicaciones Cliente Servidor

- Introducción

- Arquitectura Cliente Servidor (genérica)



- Procesamiento distribuído.  $X+Y=100\%$

- Multiusuario

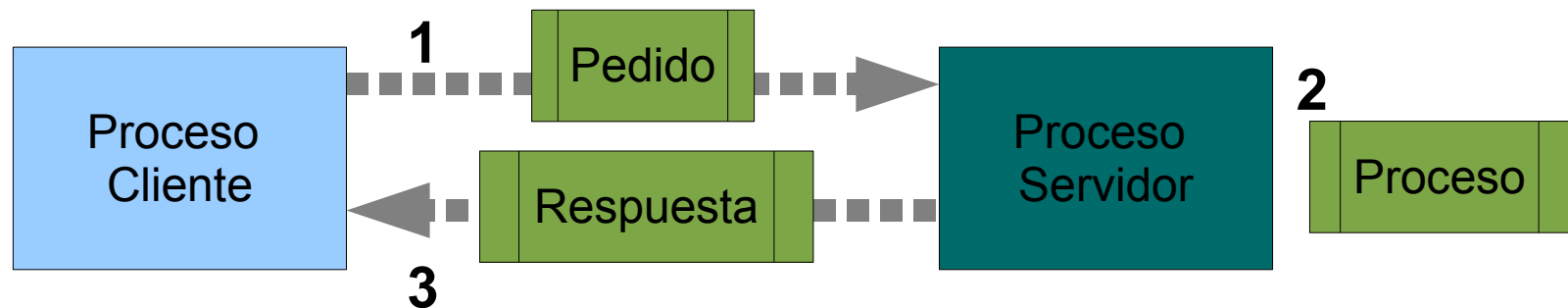
- LAN/WAN

# Aplicaciones Cliente Servidor

- Introducción

- Arquitectura Cliente Servidor

- La arquitectura cliente/servidor es una **infraestructura** versátil, modular y basada en mensajes (message-based) que mejora la **usabilidad, flexibilidad, interoperabilidad y escalabilidad** de los sistemas.
    - Funcionalmente se basa en el principio maestro-esclavo

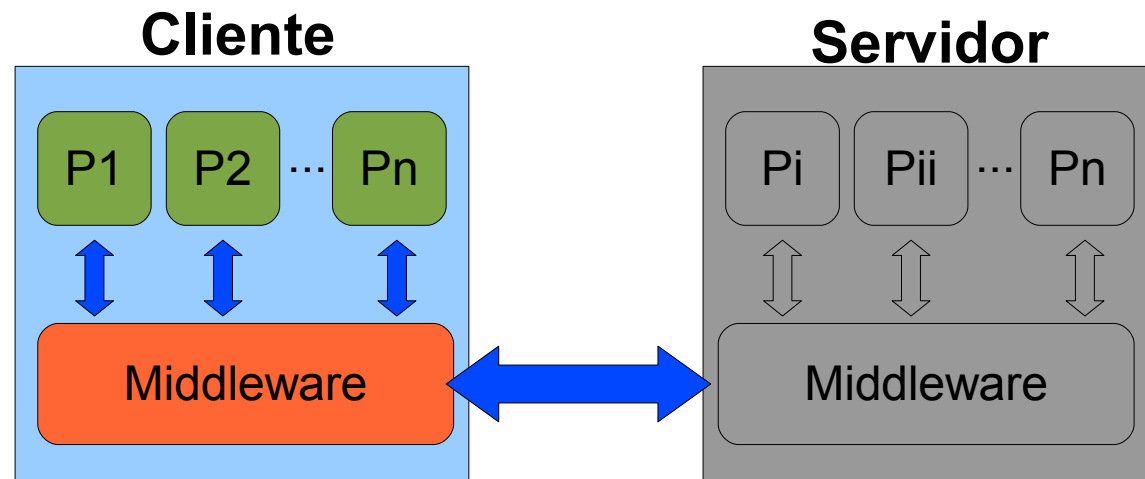


# Aplicaciones Cliente Servidor

- Conceptos

- Cliente

- Ejecuta los procesos que generalmente interactúan con el usuario, y en la arquitectura C/S son los que inician una petición (remitente o maestro) de un servicio a los servidores



# Aplicaciones Cliente Servidor

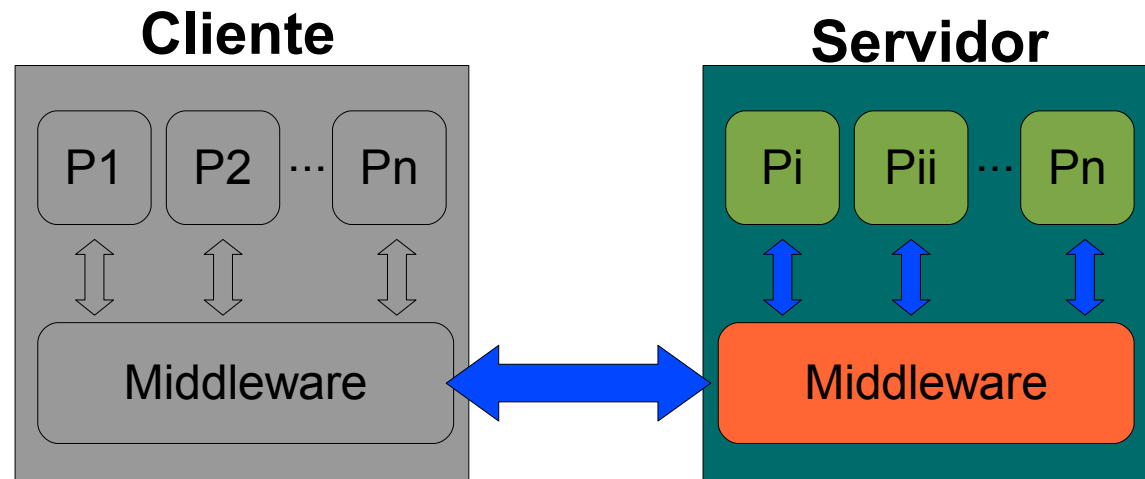
- Conceptos
  - Cliente
    - Generalmente permite interactuar al usuario
    - No siempre conoce el servidor
    - Plataforma independiente del servidor
    - Hay una gran variedad de dispositivos

# Aplicaciones Cliente Servidor

- Conceptos

- Servidor

- Ejecuta los procesos que atienden las solicitudes requeridas por los clientes. (esclavos)



# Aplicaciones Cliente Servidor

- Conceptos
  - Servidor
    - Atiende múltiples solicitudes
    - Maneja prioridades de atención
    - Maximizar disponibilidad
    - Debe ser seguro
    - Puede necesitar de otros servidores
    - Plataforma independiente del cliente

# Aplicaciones Cliente Servidor

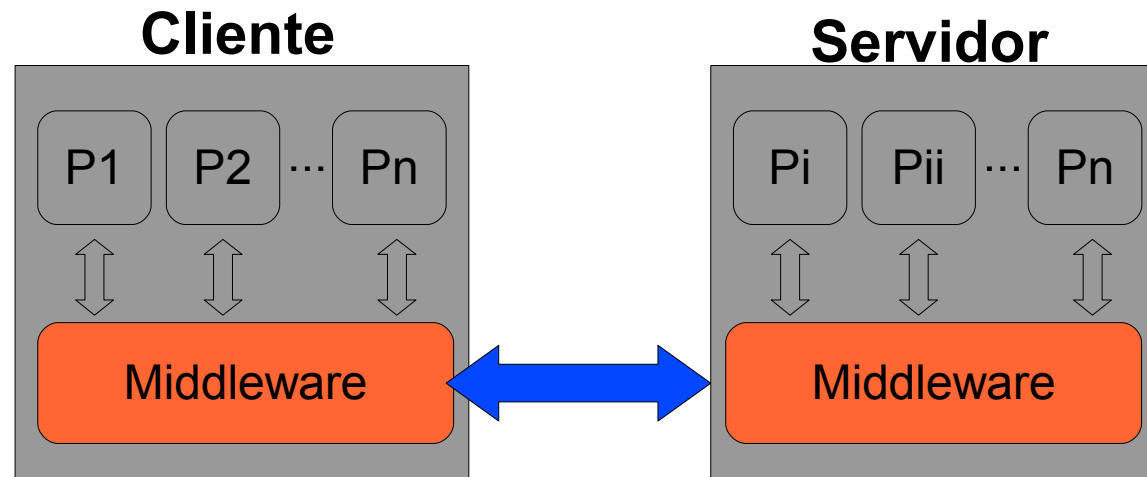
- Conceptos
  - Servidores Típicos
    - RDBMS
    - Mail
    - DNS
    - Web
    - ...

# Aplicaciones Cliente Servidor

- Conceptos

- Middleware

- Es la infraestructura que se ejecuta en ambos extremos y soporta el paso de los mensajes.

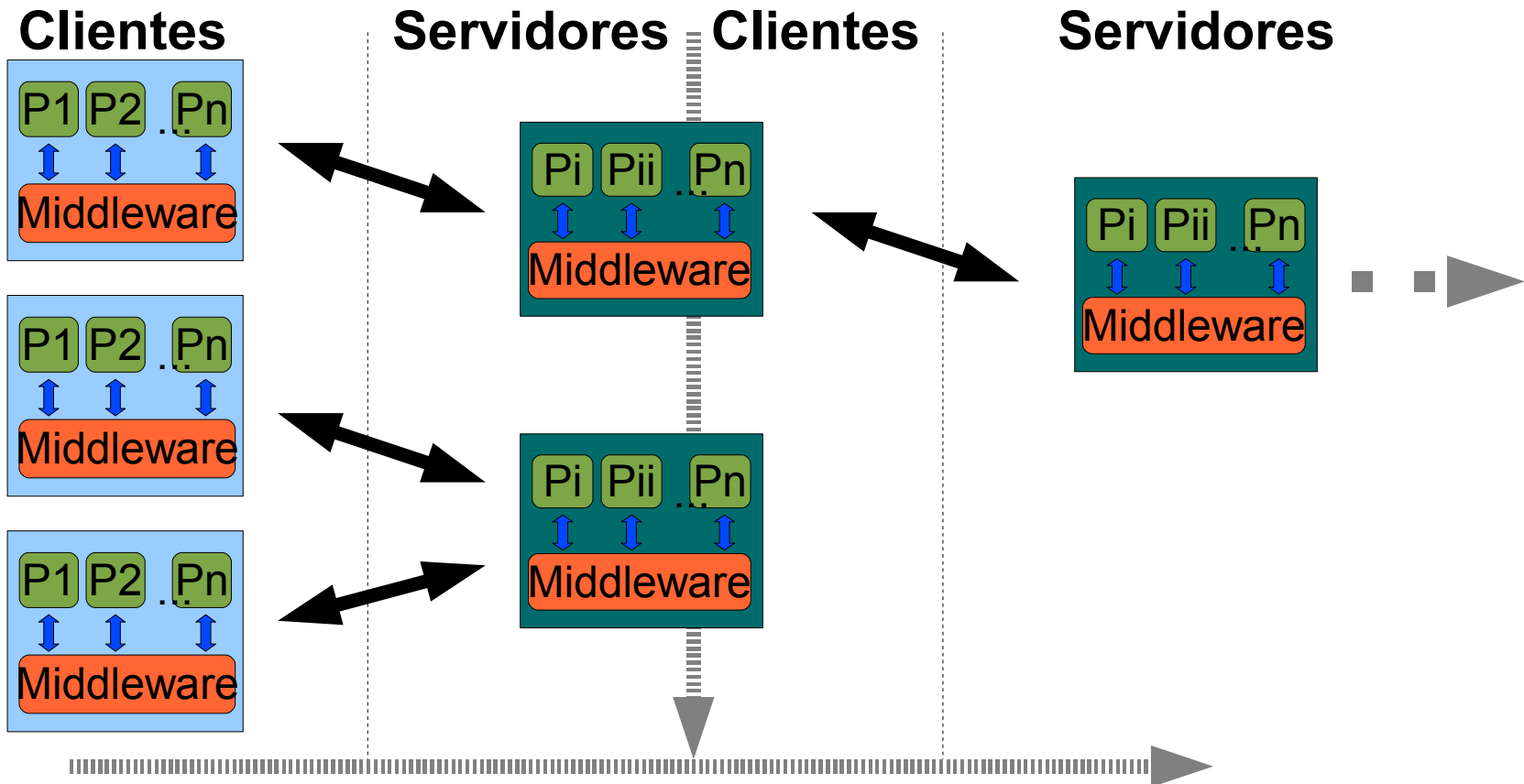


# Aplicaciones Cliente Servidor

- Conceptos
  - Middleware General
    - TCP/IP
    - Named Pipes
    - NetBios
  - Middleware Específico
    - ODBC, JDBC
    - MAPI, SMTP, POP
    - CORBA, RMI, DCOM
    - HTTP, SSL
    - SOAP

# Aplicaciones Cliente Servidor

- Conceptos
  - Escalabilidad horizontal y vertical



# Aplicaciones Cliente Servidor

- Conceptos
  - Ventajas Cliente Servidor
    - Usabilidad
    - Flexibilidad
    - Interoperabilidad
    - Escalabilidad

# Aplicaciones Cliente Servidor

- Conceptos
  - Desventajas de Cliente Servidor
    - Integrar componentes cliente, middleware y servidor.
    - El diseño de las aplicaciones es más complejo.
    - Debug no siempre integrado.
    - Deploy con mayores dificultades.
    - Administración de cambios más complejos.

# Aplicaciones Cliente Servidor

- Conceptos

- Layers y Tiers. Capas y Niveles.

- Layers se usa para referir a capas lógicas en las que se separa una aplicación de software.
    - Tiers se usa para referir a la separación física de los componentes o capas de una aplicación entre varios servidores.
    - Sin embargo, por la mala aplicación del lenguaje se aplican indistintamente los términos “Capa” y “Nivel”. Lo que a veces ocasiona algunas malinterpretaciones.

# Aplicaciones Cliente Servidor

- Diseño

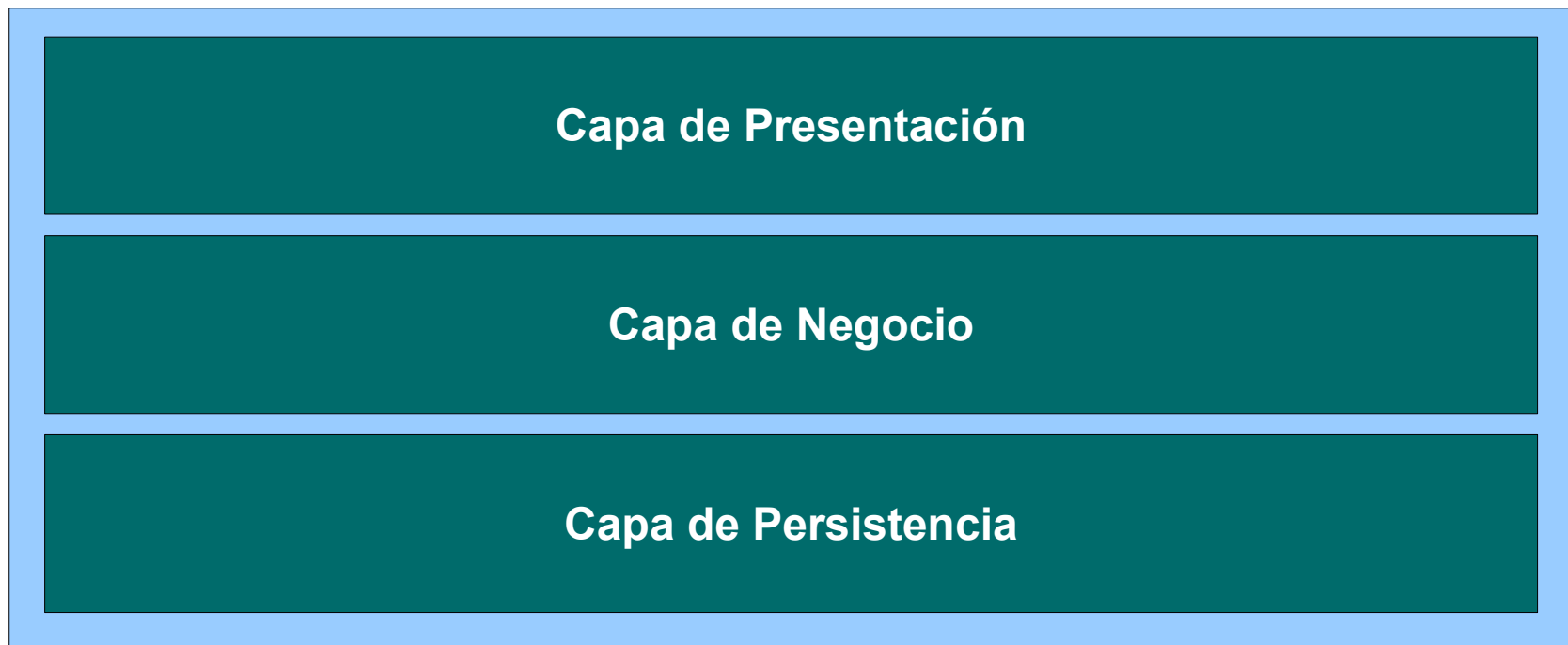
- Arquitectura en Capas

- Tiene como objetivo separar la lógica de una aplicación en capas con el principio de que la comunicación se da únicamente entre capas adyacentes.
    - El criterio más difundido es separar la lógica de diseño, de la lógica de negocios.
    - La ventaja de este diseño es que separando la aplicación en capas y aprovechando la arquitectura cliente servidor podemos crear aplicaciones de varios niveles.

# Aplicaciones Cliente Servidor

- Diseño
  - Arquitectura en Capas

**Ej: Aplicación de 3 capas**

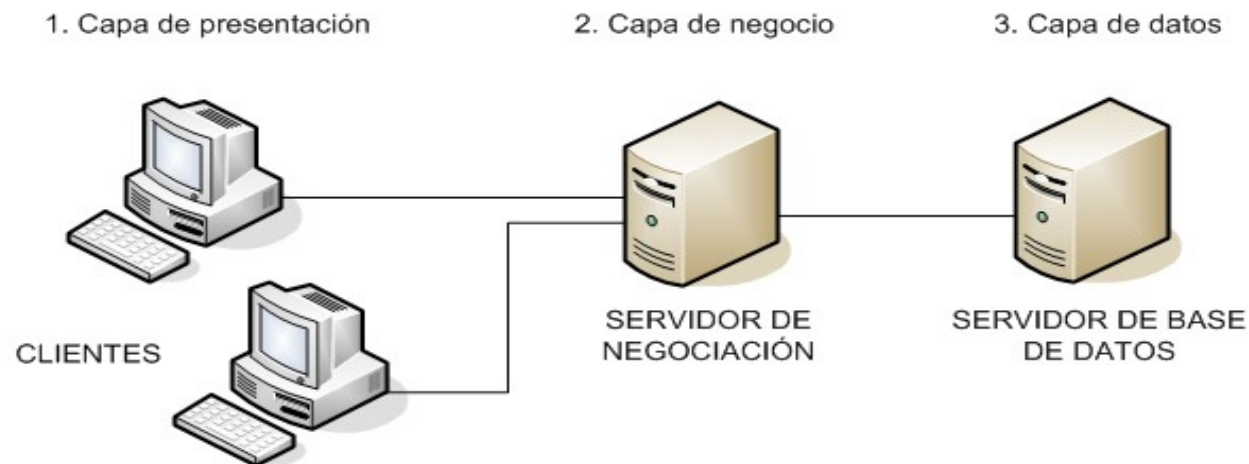


# Aplicaciones Cliente Servidor

- Diseño

- Arquitectura de Niveles

- Los niveles son independientes.
    - Se puede organizar el equipo de desarrollo por niveles y skill.
    - Problemas: robustez, tasa de fallos, debug.



# Aplicaciones Cliente Servidor

- Diseño

- Nuevamente Capas y Niveles

- Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y un nivel.
    - Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.
    - Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

# Aplicaciones Cliente Servidor

- Diseño

- Arquitectura MVC

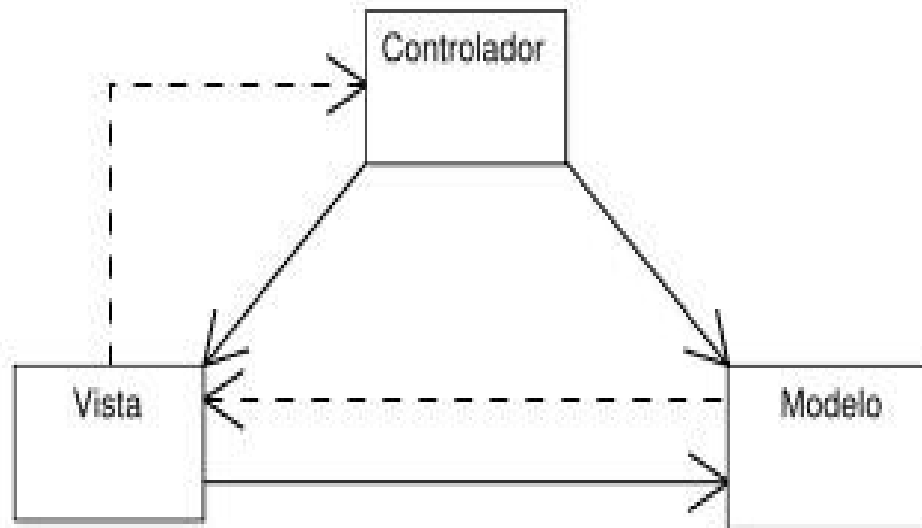
- Tiene como objetivo organizar la lógica de una aplicación separando los datos, la interfaz de usuario y la lógica de control en tres componentes diferentes.
    - La **Vista** es la lógica que permite interactuar al usuario con la aplicación.
    - El **Controlador** es la lógica que procesa las solicitudes de usuario y controla la ejecución de la aplicación.
    - El **Modelo** se compone de la lógica de negocios.

# Aplicaciones Cliente Servidor

- Diseño

- Arquitectura MVC

- Lo que sería capa de presentación, se separa en vista y controlador.



# Aplicaciones Cliente Servidor

- Diseño
  - Arquitectura MVC
    - Aumenta el desacoplamiento entre la vista y el modelo.
    - Reduce la complejidad en el diseño de la arquitectura.
    - Incrementa flexibilidad.
    - Permite la reutilización.
  - Aclaración: MVC también como patrón de diseño.

# Aplicaciones Cliente Servidor

- Diseño

- Arquitectura de Servicios. SOA.

- Es una manera de organizar los sistemas y su integración, basándose en los procesos de negocio, implementados como servicios interoperables.
    - Fuertemente orientada a la integración de sistemas y a la creación y cambios de los procesos de negocio.
    - Se basa en:
      - Principio de diseño “service-orientation”.
      - Cada servicio se expone via interfaces que lo describen para que otros (apps y servicios) lean y comprendan como puede ser utilizado.

# Aplicaciones Cliente Servidor

- Diseño
  - Arquitectura de Servicios. SOA.
    - Reusabilidad
    - Granularidad
    - Modularidad
    - Composición
    - Portabilidad
    - Interoperabilidad
    - Basada en estándares: HTTP, SOAP, XML, BPEL, etc.

# Aplicaciones Cliente Servidor

- Diseño

- Arquitectura de Servicios. SOA.

- SOA 1.0

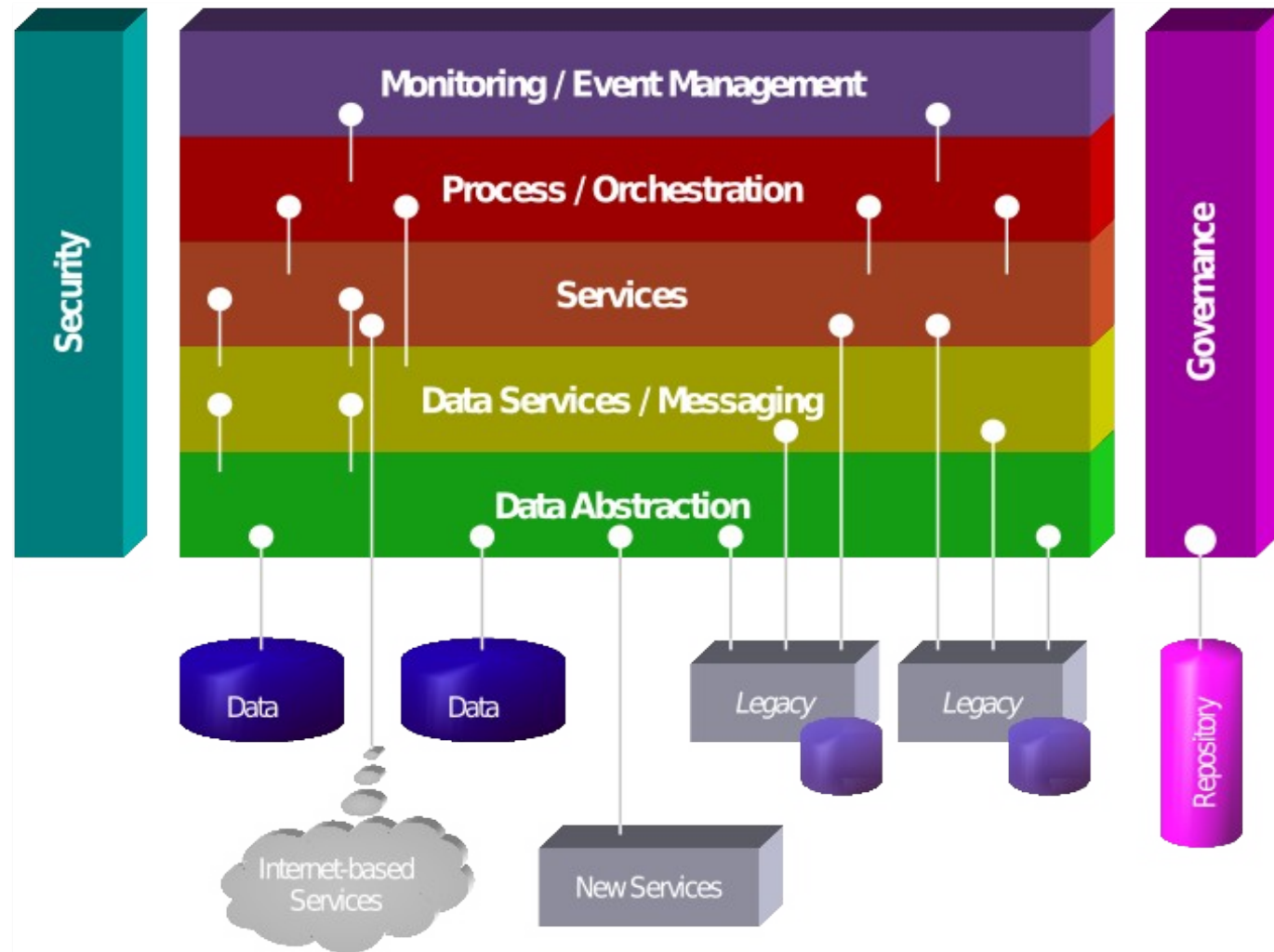
- Basada en arquitectura cliente servidor
      - Implementada mayormente con Web Services
      - Implica elementos tecnológicos, de la IS y de negocios.
      - Enfocada a procesos de negocio. Capa de negocio.

- SOA 2.0

- Se incorpora la arquitectura dirigida por eventos.
      - Ofrece mayor potencial ya que la SOA respondería a cambios de estado y no solamente a solicitudes.

# Aplicaciones Cliente Servidor

- Diseño  
– SOA



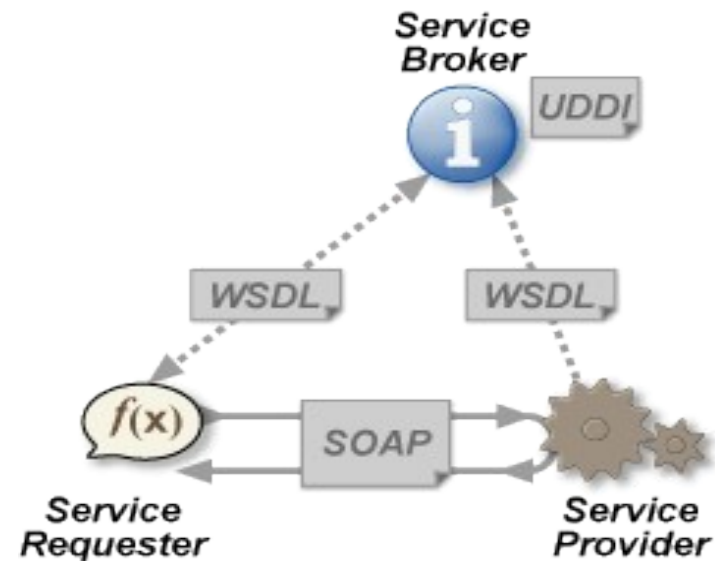
# Aplicaciones Cliente Servidor

- Diseño

- Servicios web

- “Sistema de software diseñado para soportar interoperabilidad (máquina a máquina) sobre una red” (W3C).

- HTTP
      - XML
      - XML-RPC
      - SOAP
      - WSDL
      - UDDI
      - BPEL
      - WS\*



# Aplicaciones Cliente Servidor

- Diseño
  - Invocaciones
    - **RPC**
    - **XML-RPC**
    - **SOAP**
    - **GWT RPC**
    - **REST**
    - RMI
    - DCOM
    - CORBA

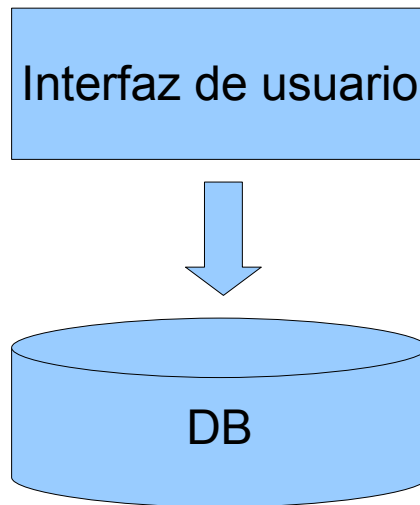
# Aplicaciones Cliente Servidor

- Desarrollo
  - Arquitectura de software como disciplina.
  - La arquitectura puede orientar el proceso.
  - Contemplar skills del equipo.
  - Manejar la complejidad del entorno de desarrollo.
  - Usar herramientas adecuadas.
  - Selección adecuada de componentes, frameworks y middleware.
  - Licencias.

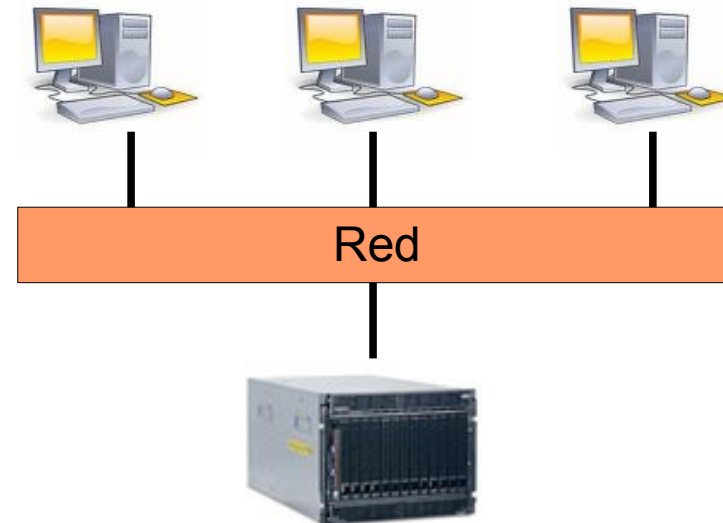
# Aplicaciones Cliente Servidor

- Modelos
  - 2 Capas

## Modelo lógico



## Implementación



# Aplicaciones Cliente Servidor

- Modelos

- 2 Capas

- Ideal para soluciones de negocios bajo redes LAN.
    - Amplio apoyo de la industria.
    - RAD.
    - Escala pero en forma limitada.
    - Conexiones “keep-alive”.
    - Dependencia de proveedor.
    - Integración.
    - Portabilidad.

# Aplicaciones Cliente Servidor

- Modelos

- 3 Capas

## Modelo lógico

Interfaz de usuario



Lógica de negocios



DB

## Implementación



Red/Internet



Red/Internet



# Aplicaciones Cliente Servidor

- Modelos

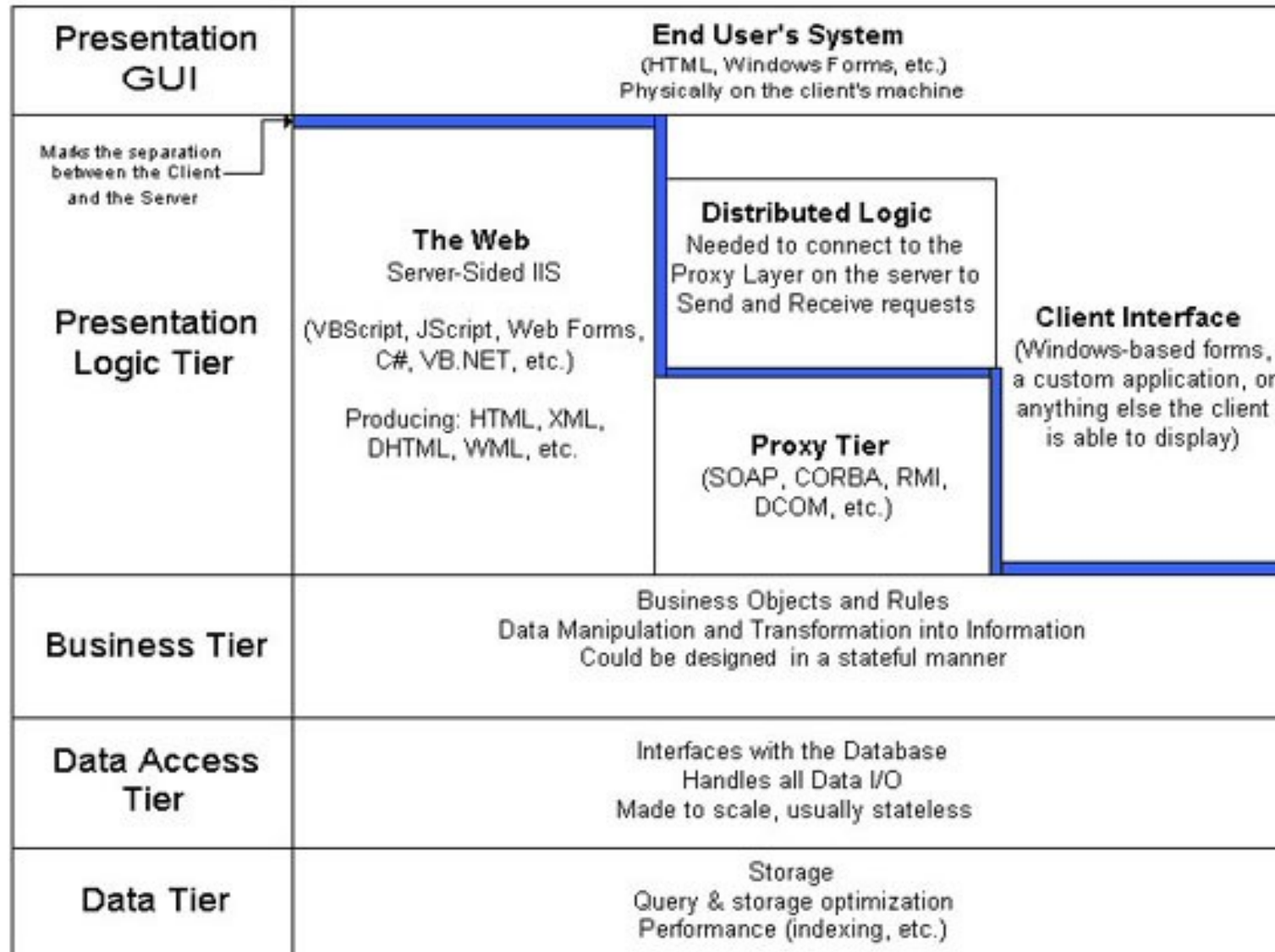
- 3 Capas

- Extiende las limitaciones de las 2 capas.
    - La capa intermedia mejora: performance, flexibilidad, mantenibilidad y reusabilidad.
    - Agrega un nivel para lógica de negocios y middleware más complejo.
    - Integración de plataformas y lenguajes.
    - Herramientas de diseño y deploy incompletas.

# Aplicaciones Cliente Servidor

- Modelos
  - n Capas
    - Se extiende alguna de las capas.
    - Se distribuye la lógica.
    - Se minimiza el acoplamiento.
    - Se adoptan frameworks.

# Aplicaciones Cliente Servidor



# Aplicaciones Cliente Servidor

- Casos Reales

- Rich Internet Applications (RIA)

- Basadas en la arquitectura de la web
    - Mejora la experiencia del usuario
    - Suma funciones naturales de interfaces gráficas y la flexibilidad de desacoplamiento de la web
    - Arquitectura cliente servidor con paso de mensajes asíncronos.

# Aplicaciones Cliente Servidor

- Casos Reales

- Net o Java

- Net Framework
  - Windows Communication Foundation
  - Net Remoting
  - Microsoft Message Queuing
  - ADO.NET
  - IIS
- JVM
  - J2EE containers
    - Jboss
    - Apache Gerónimo
    - Glassfish
  - Servlet containers
    - Apache Tomcat
  - Frameworks: Struts, Hibernate, etc.

# Aplicaciones Cliente Servidor

- Casos Reales
  - Web 2.0

# Ciclo de Ingeniería de Software

Desarrollo Iterativo de Software  
Aplicaciones Cliente Servidor  
Aplicaciones OO

Universidad FASTA  
2008

