

Desarrollo de Aplicaciones Web

Tema I: Introducción

Presentación

- Profesor Gregorio Villalobos Camacho
 - Bachillerato Sistemas Universidad Nacional
 - Maestría Universidad de Ottawa, Canadá
 - Horas de consulta: Martes 5pm y Jueves 6 pm
 - Correo electrónico:
 - gregorio.villalobos.camacho@una.cr
- Presentación De los Estudiantes

Introducción

- Revisión de la Carta al Estudiante
- Firma de recibido de parte de los estudiantes

Deberes y Responsabilidades

- Estudiante
 - 11 Horas Semanales
 - 2 Horas Práctica
 - 2 Horas Teoría
 - 7 Horas de Estudio Independiente
 - Leer, Estudiar, Investigar
 - Desarrollar nuevos proyectos
 - Consultar

Deberes y Responsabilidades (Cont)

- Profesor
 - Impartir Curso
 - Cubrir los Temas del Curso
 - Evaluar el Avance de los Estudiantes
 - Procurar y Promover el entendimiento de los temas
 - Proveer Tiempo extra Clase para Consultas

Trabajo en Equipo

- Definición de Equipos de trabajo
 - 2 Estudiantes
- Roles
- Responsabilidades
- Manejo de Conflictos
- Distribución Equitativa del Trabajo

PRIMERA PARTE DEL CURSO

Temas a Abarcar

- Metodologías ágiles de programación
- Aplicaciones empresariales
- Arquitectura en capas
- Arquitectura en tres capas
- Distribucion en niveles

METODOLOGÍAS ÁGILES DE PROGRAMACIÓN

Contexto M todolog a  gil

- Nandhakumar & Avison 1999
 - Metodolog as tradicionales de desarrollo de sistemas de informaci n “son tratadas principalmente como una ficci n necesaria para presentar una imagen de control o para proveer un estatus simb lico”.
- Truex et al. 2000
 - Es posible que los m todos tradicionales sean “meramente ideales inalcanzables y hip t icos *straw-men* que proveen una gu a normativa a situaciones de desarrollo ut picas.”
- McCauley 2001
 - La filosof a en la cual se basan los m todos orientados a procesos establece que los requerimientos de un proyecto de software quedan congelados antes de que el dise o y desarrollo del software comience.

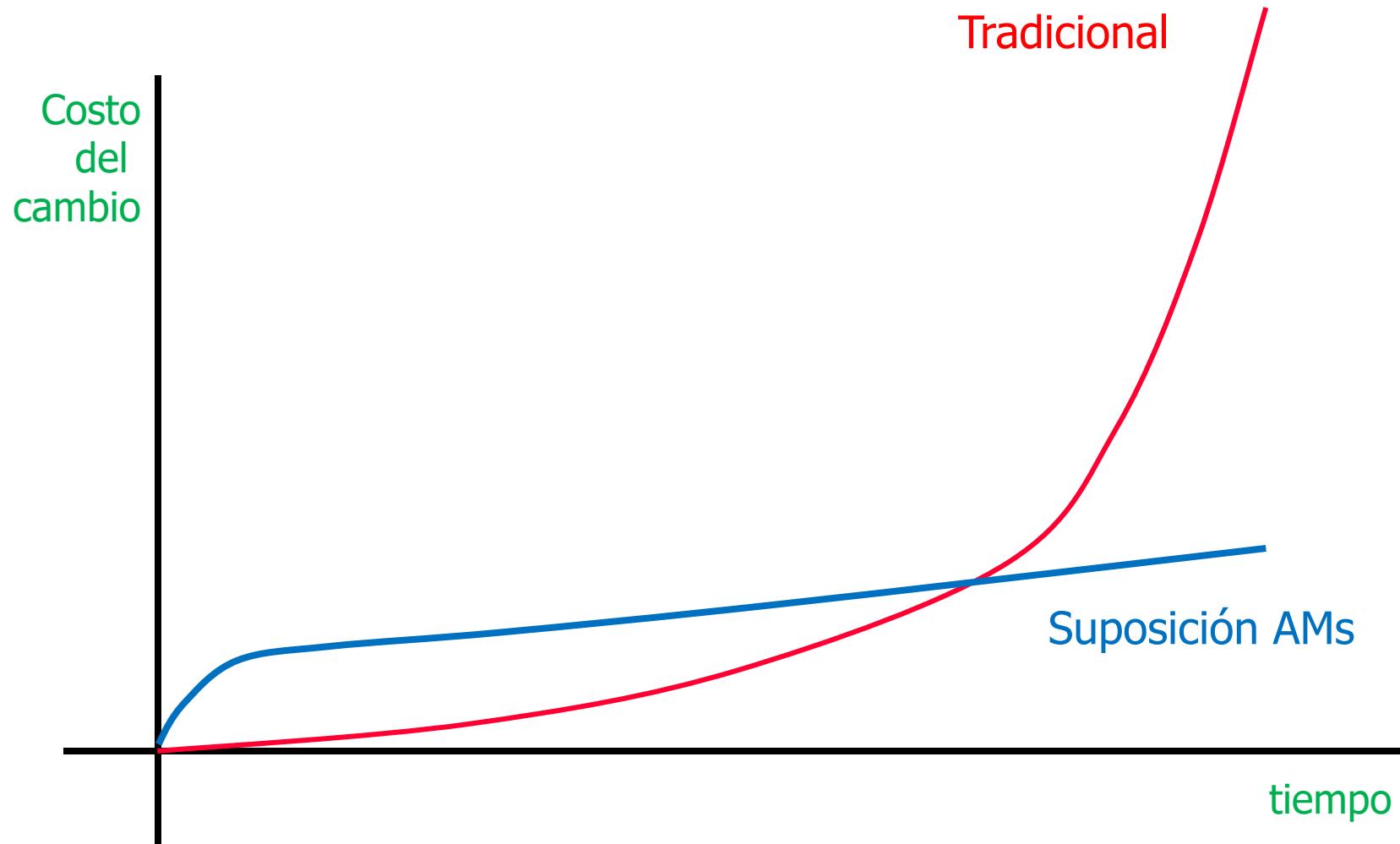
La metodología Ágil (AM) valora

- Más al individuo y las interacciones en el desarrollo
 - Menos que a las actividades y las herramientas
- Más el desarrollar software que funcione
 - Menos que conseguir una buena documentación
 - Busca minimalismo del modelado y documentación del sistema
- Más la colaboración con el cliente
 - Menos que la negociación de un contrato
- Responder a los cambios de forma ágil
 - Menos que seguir estrictamente una planificación

¿Por qué surgen las Metodologías Ágiles (AMs)?

- Dificultad para implantar metodologías tradicionales. Sofisticadas herramientas CASE y notaciones (UML)
- Una solución a medida para un segmento importante de proyectos de desarrollo de software
- Pugna entre comunidades/gurús
- “Aceptar el cambio” ...

Costo de los Cambios en SW



Manifiesto de las AMs

Principios

1. La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software que le reporte un valor
2. Dar la bienvenida a los cambios. Los AMs capturan los cambios para que el cliente tenga una ventaja competitiva
3. Entregar frecuentemente software que funcione, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente

Manifiesto de las Ams (cont)

4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto
5. Construir proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir el trabajo
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo
7. El software que funciona es la medida principal de progreso

Manifiesto de las Ams (cont)

8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad
10. La simplicidad es esencial
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos
12. En intervalos regulares, el equipo reflexiona respecto de cómo llegar a ser más efectivo, y según esto ajusta su comportamiento

Comparación Ágil - \neg Ágil

Metodología Ágil	Metodología No Ágil
Pocos Artefactos	Más Artefactos
Pocos Roles	Más Roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
Cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

Metodologías Ágiles

Metodología	Acrónimo	Creación	Tipo de modelo	Característica
Adaptive Software Development	ASD	Highsmith 2000	Prácticas + Ciclo de vida	Inspirado en sistemas adaptativos complejos
Agile Modeling	AM	Ambler 2002	“Metodología basada en la práctica”	Suministra modelado ágil a otros métodos
Crystal Methods	CM	Cockburn 1998	“Familia de metodologías”	MA con énfasis en modelo de ciclos
Agile RUP	dX	Booch, Martin, Newkirk 1998	Framework / Disciplina	XP dado vuelta con artefactos RUP
Dynamic Solutions Delivery Model	DSDM	Stapleton 1997	Framework / Modelo de ciclo de vida	Creado por 16 expertos en RAD
Evolutionary Project Management	Evo	Gilb 1976	Framework adaptativo	Primer método ágil existente
Extreme Programming	XP	Beck 1999	“Disciplina en prácticas de ingeniería”	Método ágil radical
Feature-driven development	FDD	De Luca & Coad 1998 Palmer & Felsing 2002	“Metodología”	Método ágil de diseño y construcción
Lean Development	LD	Charette 2001, Mary y Tom Poppendieck	“Forma de pensar” – Modelo logístico	Metodología basada en procesos productivos
Microsoft Solutions Framework	MSF	Microsoft 1994	Lineamientos, Disciplinas, Prácticas	Framework de desarrollo de soluciones
Rapid Development	RAD	McConnell 1996	Survey de técnicas y modelos	Selección de <i>best practices</i> , no método
Rational Unified Process	RUP	Kruchten 1996	Proceso unificado	Método (¿ágil?) con modelado
Scrum	Scrum	Sutherland 1994 - Schwaber 1995	“Proceso” (framework de management)	Complemento de otros métodos, ágiles o no

Tarea: Investigar sobre

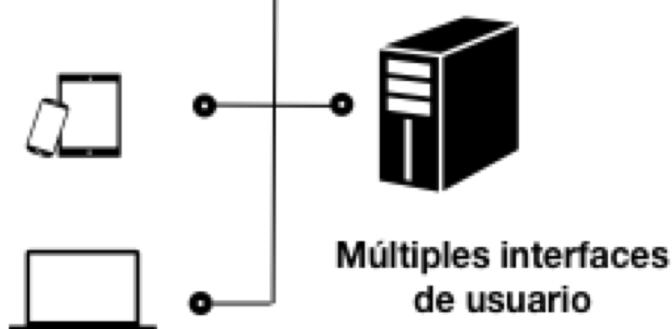
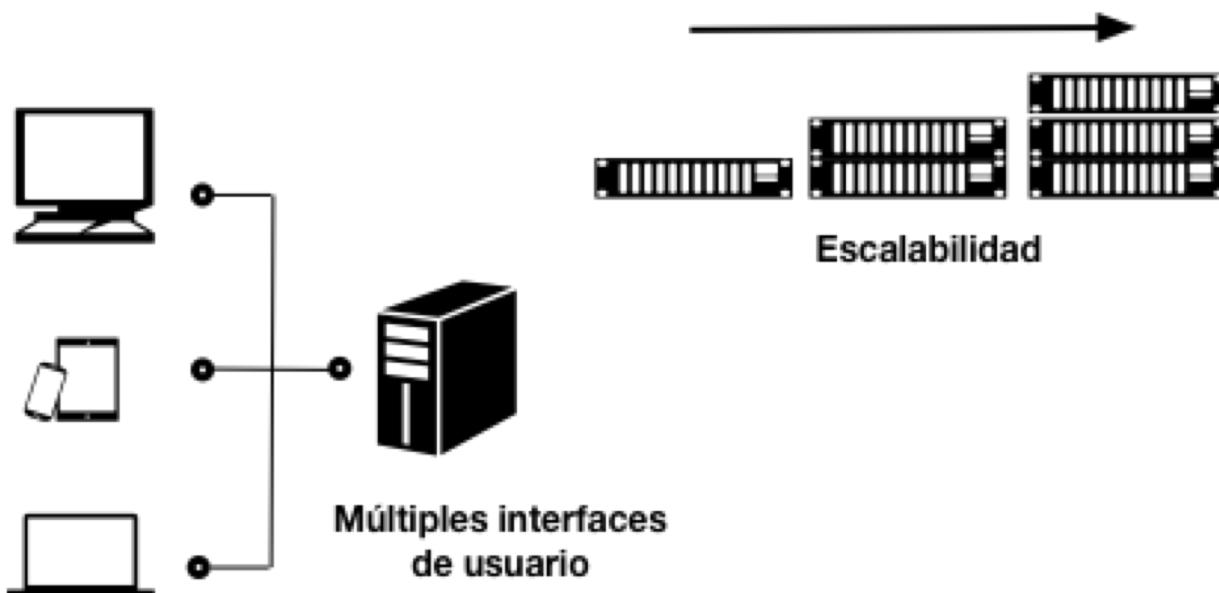
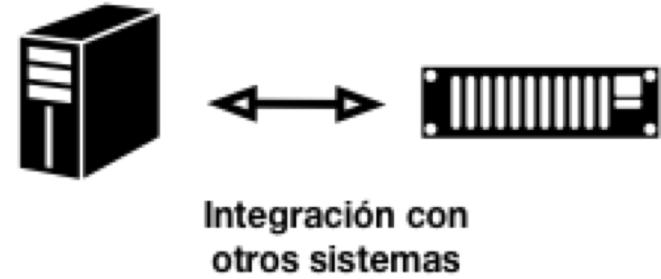
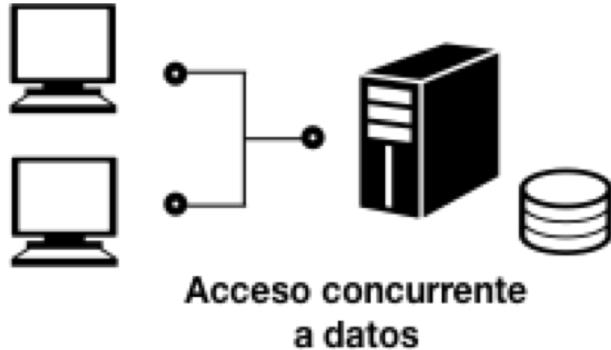
1. eXtreme Programming (Incluir sus 12 prácticas)
2. SCRUM (principios y el ciclo de SCRUM)
3. FDD (Feature Driven Development)
4. Lean Development
5. DSDM (Dynamic systems development method)
6. ASD (Adaptive software development)
7. EVO (Evolutionary Software Development)
8. Métodos ágiles en MSF
9. Rapid Development
10. RUP
11. Agile Modeling (Incluir mejores prácticas)
12. Crystal Methods
13. AUP

APLICACIONES EMPRESARIALES

Importancia de las Aplicaciones Empresariales (AE)

- Su ejecución y puesta en práctica
 - Clave para el éxito de la economía de la información
- Son altamente complejas (ERP)
- Tienen altas expectativas
- Debe tener alta disponibilidad
- Debe ser escalable, maleable y predecible
- Debe responder fácilmente al negocio
- Debe moverse rápido de prototipos a producción

AE: Características



AE: Características (cont)

- Acceso concurrente a los datos
- Múltiples interfaces de usuario
 - Los usuarios pueden ser ocasionales o regulares
 - Con algo o nada de experiencia en TI
 - Se debe presentar múltiples interfaces por tipo de datos
 - Muchos sistemas tiene procesamientos por lotes
 - Por un diseño enfocado en casos de uso

AE: Características (cont)

- Integración con otros sistemas
 - Tienen contacto con otras apps en la empresa
 - Desarrollos en diferentes momentos
 - Desarrollos con diferentes tecnologías
 - Cada uno con su tipo de comunicación
 - La coordinación se vuelve complicada
 - Puede incluir integración con socios de negocio

AE: Características (cont)

- Escalabilidad
 - Maneja modificaciones e incrementos
 - Mantiene niveles de rendimiento aceptable
 - Tanto vertical como horizontal
 - Permite añadir equipos si aumenta la carga
 - Se debe identificar el incremento
 - Su naturaleza
 - Entender su impacto en todo el sistema

AE: Características (cont)

- Disponibilidad
 - Tiempo que el sistema está en ejecución
 - Depende de diseños cuidadosos
 - Depende de una operación disciplinada
 - Con control de cambios
 - Pruebas rigurosas
 - Mecanismos de actualización rápidos
 - Recuperación eficiente de errores

Arquitectura en Capas

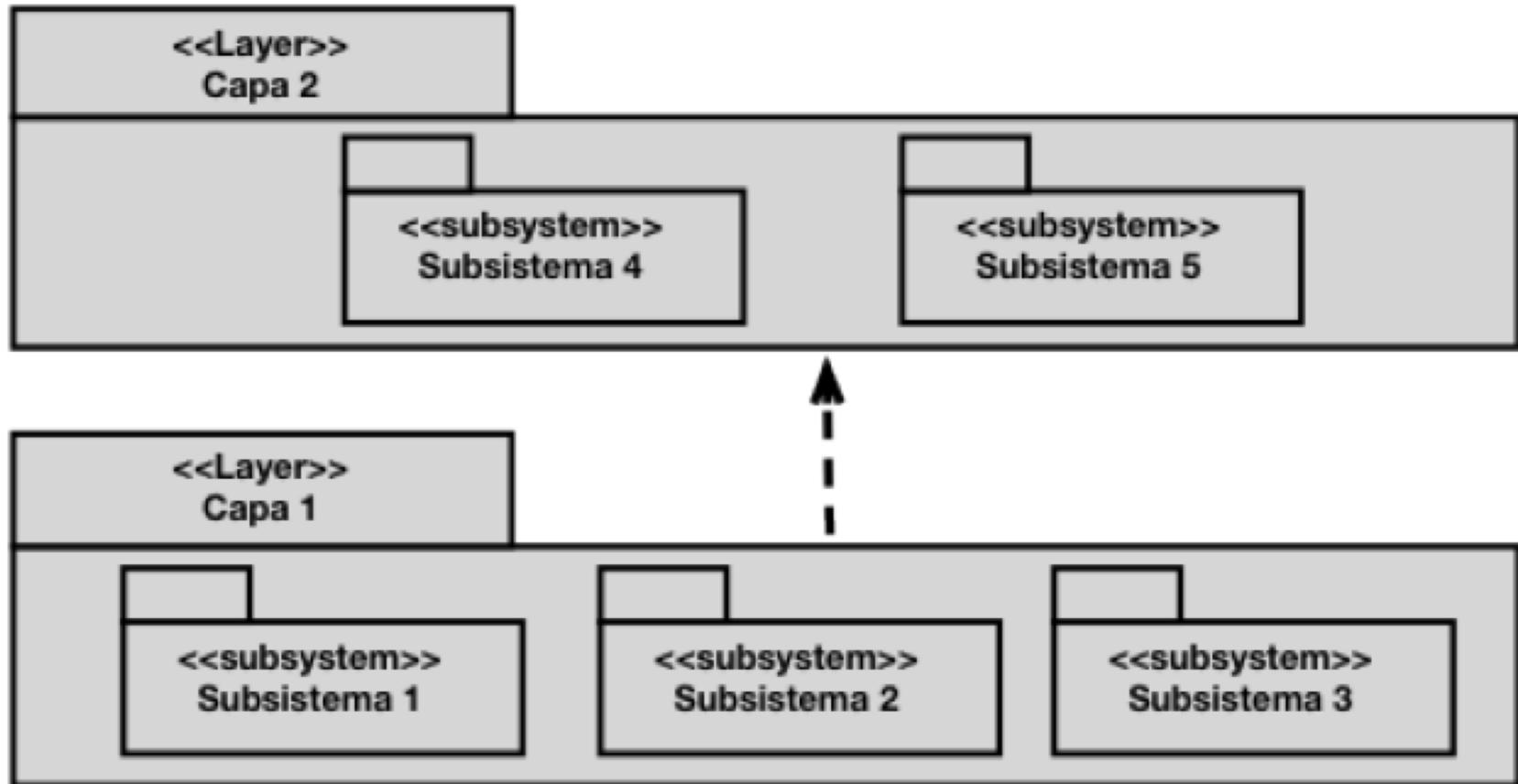
Arquitectura en Capas

- Las mejores soluciones Empresariales
 - Son simples para resolver el proceso necesario
 - Son simples de mantener
 - Son simples de agregar nuevas funcionalidades
- Cuando el sistema es grande y complejo
 - Es necesario utilizar mecanismos simples
 - Múltiples partes forman un todo

Arquitectura en Capas (Cont)

- La Arquitectura en capas
 - Permite abstracción
 - Maneja seguridad por niveles
 - Divide grandes áreas en módulos
 - Cada capa está débilmente acoplada a la inferior
 - Los componentes de una capa pueden interactuar
 - Con elementos del mismo nivel
 - Con elementos de niveles inferiores
 - La idea es que en la misma capa
 - Los componentes sean cohesivos entre sí

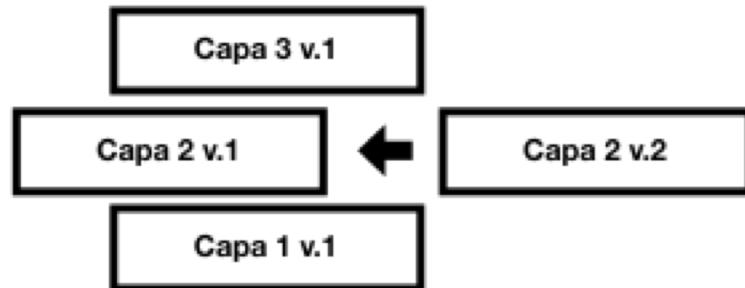
Arquitectura en Capas (Cont)



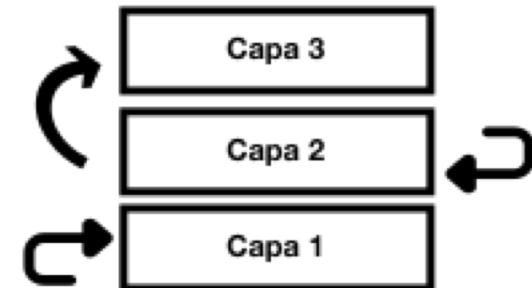
Arquitectura en Capas (Cont)

- Beneficios
 - Cada capa se entiende por sí misma
 - 1 capa no tiene que conocer las otras capas
 - Los cambios se reducen a 1 capa
 - Reduce el tiempo de depuración y corrección de errores
 - Permite reutilización de código
 - Se puede rediseñar 1 capa sin afectar las demás
 - Permite dividir desarrollo en equipos de trabajo
 - La separación puede ser además física
 - Incrementando la tolerancia a fallos

Arquitectura en Capas (Cont)



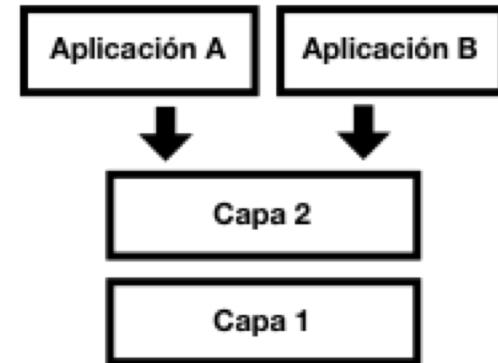
Se puede sustituir la implementación



Un cambio afecta pocas capas



Múltiples equipos pueden trabajar con independencia

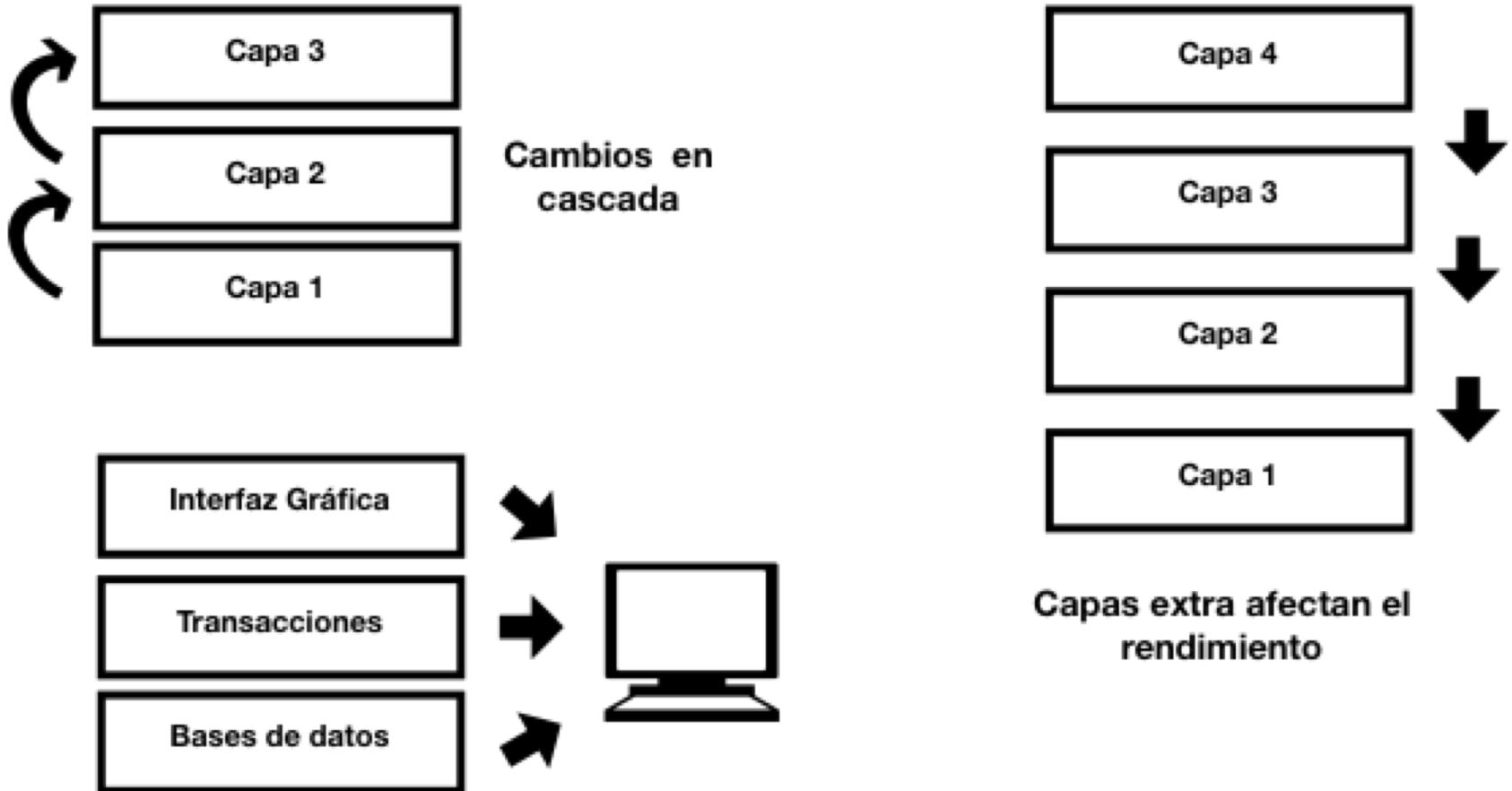


La misma capa puede ser usada por varias aplicaciones

Arquitectura en Capas (Cont)

- Desventajas
 - No siempre se puede aplicar
 - Puede ser más complejo el mantenimiento
 - Cambios pueden generar efectos en cascada
 - Hacia las capas superiores
 - Puede generar problemas en el rendimiento
 - Algunas apps son simples y no necesitan capas

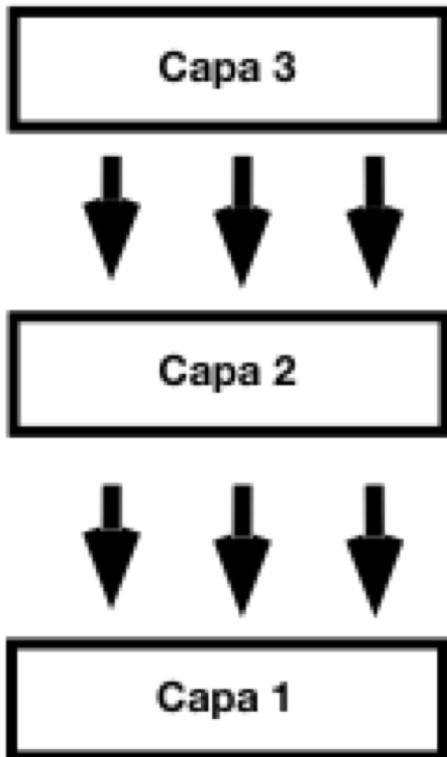
Arquitectura en Capas (Cont)



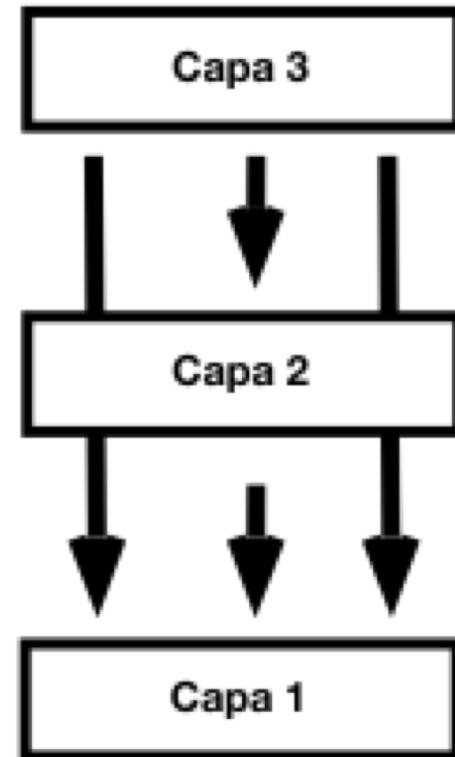
Agregan complejidad a
aplicaciones sencillas

Arquitectura en Capas (Cont)

Tipos de Enfoques



Capas estrictas



Capas relajadas

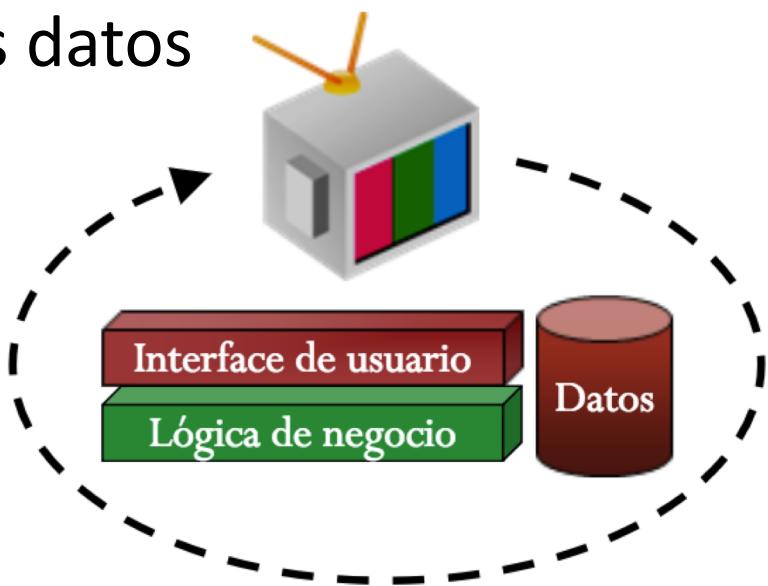
Arquitectura en 3 Capas

1 Capa

- Se conoce como aplicación monocapa
- Todo el código se encuentra mezclado
- No hay una separación
 - Difícil de mantener y modificar
- En la interfaz del usuario se contiene
 - Las operaciones normales de sistema
 - Las consultas a la base de datos

2 Capa

- Se conoce como aplicación bicapa
- Se inicia con el proceso de separación
- Genera la separación entre
 - La lógica del negocio y la interfaz
 - Con relación al acceso de los datos



3 Capas

- Aplicaciones Tricapa
 - Todo está debidamente separado
 - Se maneja por pequeñas funcionalidades

3 Capas (Cont)

- Capa I: Capa de Presentación
 - Lo que ve el usuario
 - Presenta el sistema al usuario
 - Interactúa con el usuario
 - Muestra información
 - Solicita y captura información
 - Acá está la GUI (amigable y fácil)
 - Solo se comunica con la Capa II

3 Capas (Cont)

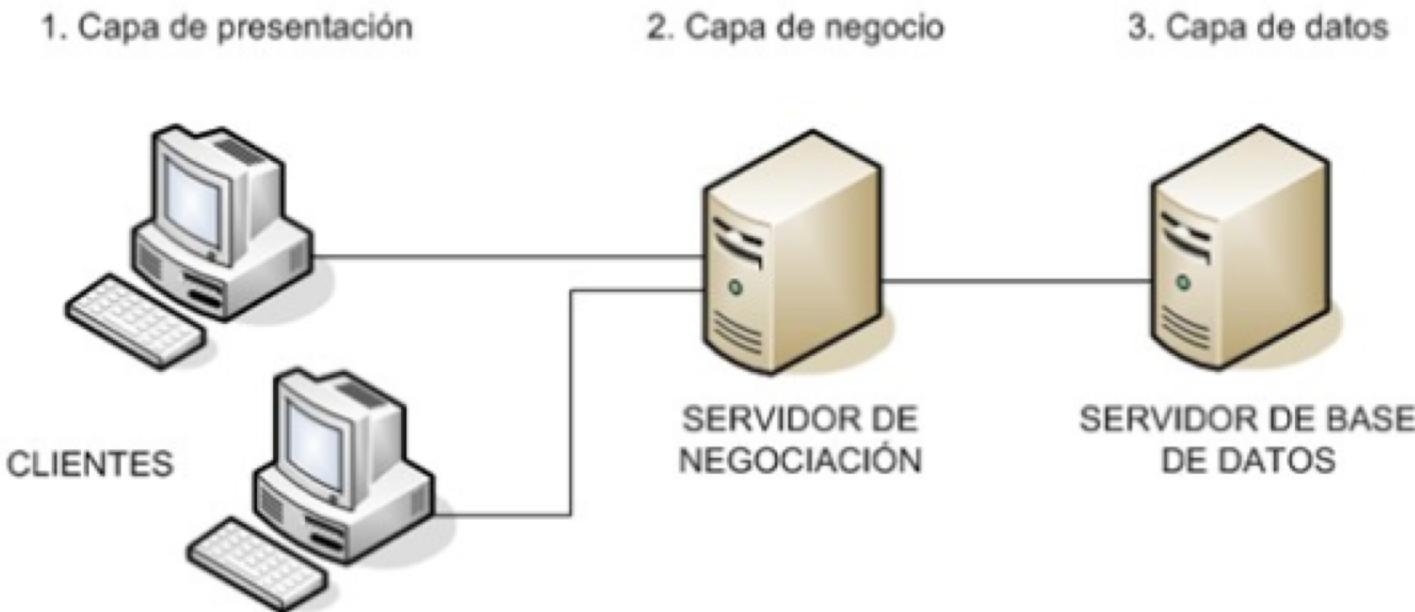
- Capa II: Capa de Negocio
 - Lugar donde se reciben las peticiones del usuario
 - Aquí se realiza el procesamiento
 - Se aplican reglas predefinidas
 - Si es necesario se trabaja con los datos (Capa III)
 - Se retornan los resultados
 - La Capa de Presentación recibe y muestra

3 Capas (Cont)

- Capa III: Capa de Datos
 - Si la Capa II necesita acceder a datos los solicita a la Capa III
 - Formada por los gestores de Bases de Datos
 - Recibe las solicitudes de
 - Almacenar datos
 - Modificar datos
 - Recuperar datos para su consulta

3 Capas (Cont)

- Las capas pueden residir en
 - Un solo ordenador (Desarrollo)
 - Múltiples servidores (Producción)



Ejercicio 3 Capas (NetBeans)

- Cree 3 Packages: Capa1, Capa2 y Capa3
- Se debe crear 4 clases Dentro de la Capa3
 - Profesor
 - Nombre, Apellido, FechaNacimiento, Dpto, Usuario, Clave
 - Materia
 - Nombre, CodigoNRC, periodo, Aula, Profesor
 - ListaMateria
 - ListaMateria[], posición
 - Agregue un método que imprima la lista
 - ListaProfesor
 - ListaProfesor[], posición
 - Agregue un método que imprima la lista
 - Recuerde agregar constructores SET y GET

Ejercicio 3 Capas

- Cree 3 Packages: Capa1, Capa2 y Capa3
- Se debe crear 4 clases Dentro de la Capa3
 - Profesor
 - Nombre, Apellido, FechaNacimiento, Dpto, Usuario, Clave
 - Materia
 - Nombre, CódigoNRC, periodo, Aula, Profesor
 - ListaMateria
 - ListaMateria[], posición, Método AgregarLista
 - Método imprimeLista (recorre lista, imprime las cartas)
 - ListaProfesor
 - ListaProfesor[], posición, Método AgregarLista
 - Método imprima la listaLista (recorre lista, imprime las cartas)
 - Recuerde agregar SET y GET
 - Las clases simples Constructores completos, las listas Constructores vacíos

Ejercicio 3 Capas: En la Capa2

- Cree 4 clases nuevas
 - LogicaMateria
 - miMateria (Constructor con mismos parámetros Materia)
 - LogicaProfesor
 - miProfesor (Constructor con mismos parámetros Profesor)
 - LogicalistaMateria
 - miListaMateria (Const Vacío, Agrega, imprime)
 - LogicalistaProfesor
 - miListaProfesor (Const Vacío, Agrega, imprime)
 - Agregue todos los SET y GET
 - Agregue e imprime invocan los métodos en Capa3

Ejercicio 3 Capas: En la Capa1

- Solamente puede trabajar con Capa2
 - Cree 2 materias
 - Cree 2 profesores
 - Agréguelos a las listas correspondientes
 - Invoque los métodos de impresión de las listas

Distribución de Niveles (Tiers)

Distribución de Niveles

- Se agrupan por niveles físicos
 - Computadores
 - Servidores
 - Clientes
- Se agrupan porque comparten una o más
 - Perfil de consumo de recursos (RAM, I/O, Sockets)
 - Requerimientos operacionales (Disponibilidad, rendimiento)
 - Restricción de diseño (Políticas de seguridad)

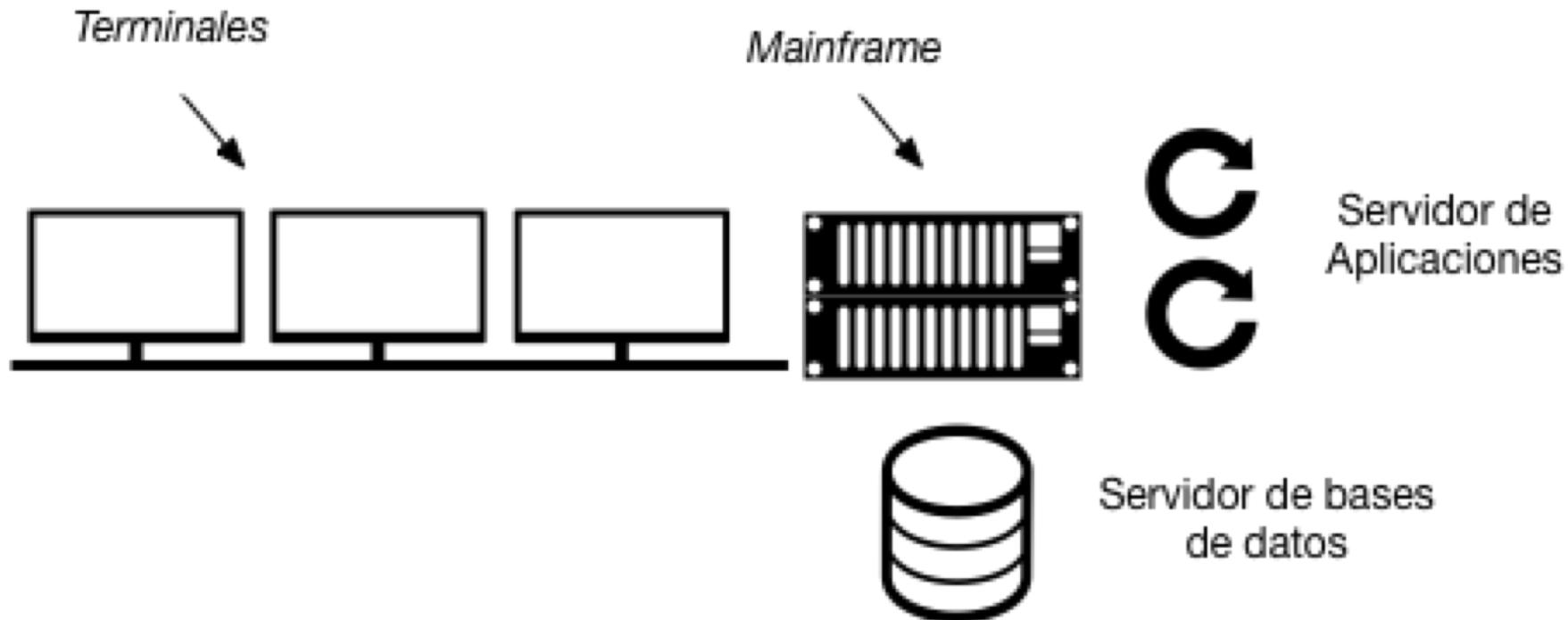
Distribución de Niveles (Cont)

- Cada nivel debe distribuir para
 - Balancear la carga entre los servidores
- La distribución se hace por arquitectura
 - Del sistema (Servidores, número y relación)
 - De la App (Componentes, empaquetado, ejecución)
 - La administración de la App
 - Se refiere a módulos ejecutables
 - Instalación y ejecución de módulos en los servidores correctos

Solución en un Nivel

- Un único PC hace todo el trabajo (Mainframe)
- Las terminales se conectan al MF (Usuarios)
 - Solo leen entradas y despliegan información
- La puesta en ejecución es fácil (Solo 1 MF)
- Más usuarios exige más recursos en el MF
 - Es poco escalable, pues agregar otro MF es caro
- Si todo trabaja en una intranet
 - La seguridad del modelo es muy alta

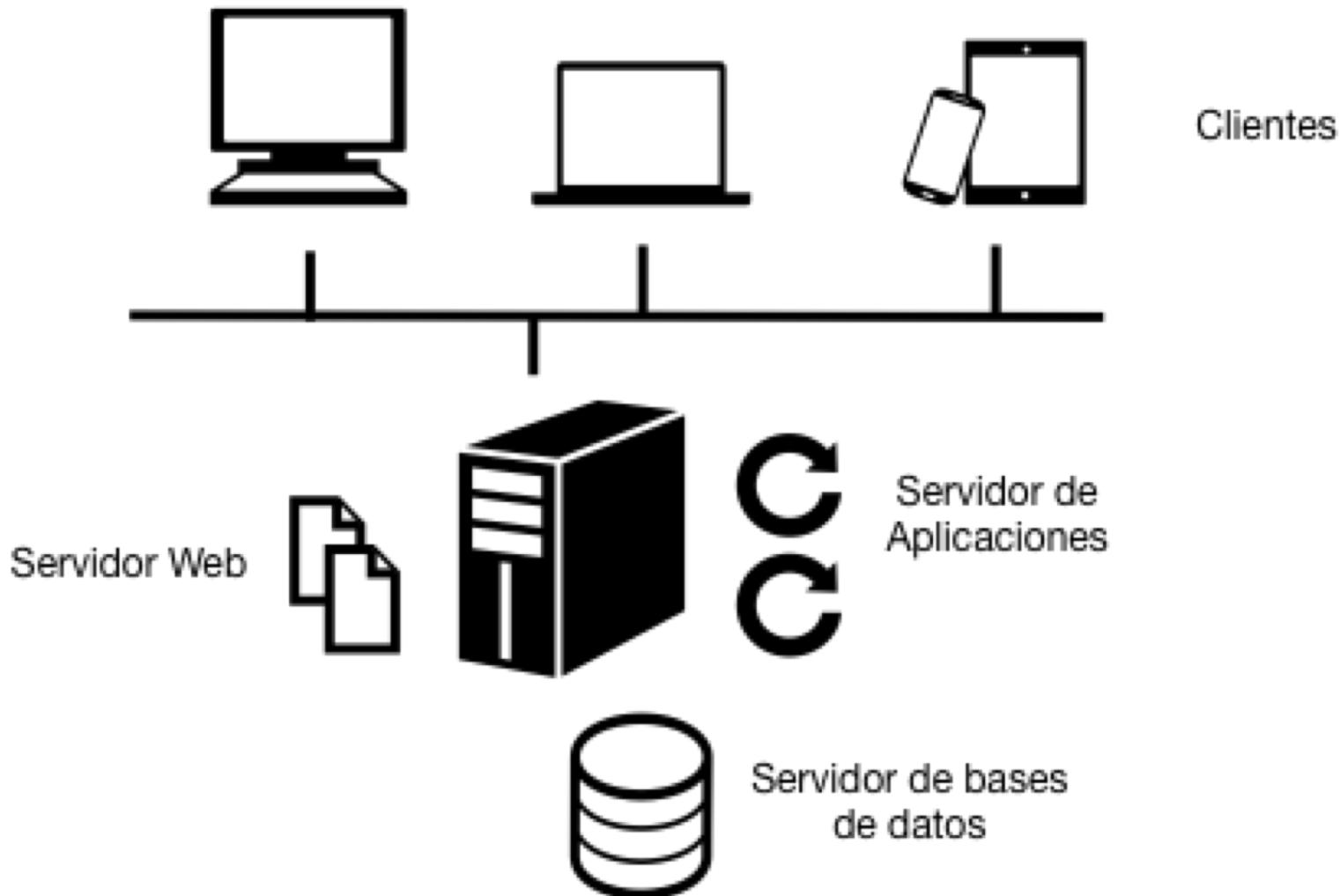
Solución en un Nivel



Solución en dos Niveles

- Cada usuario tiene un PC asignado
- Cada PC procesa y la info se almancena en BD
- Es más complicado de arrancar y mantener
- El app debe distribuirse en todas las PCs
- La escalabilidad mejora significativamente
 - Limitado por concurrencia en la BD
- El servidor de BD es más barato
 - La escalabilidad se abarata
- La seguridad es más compleja (Más vulnerable)

Solución en dos Niveles



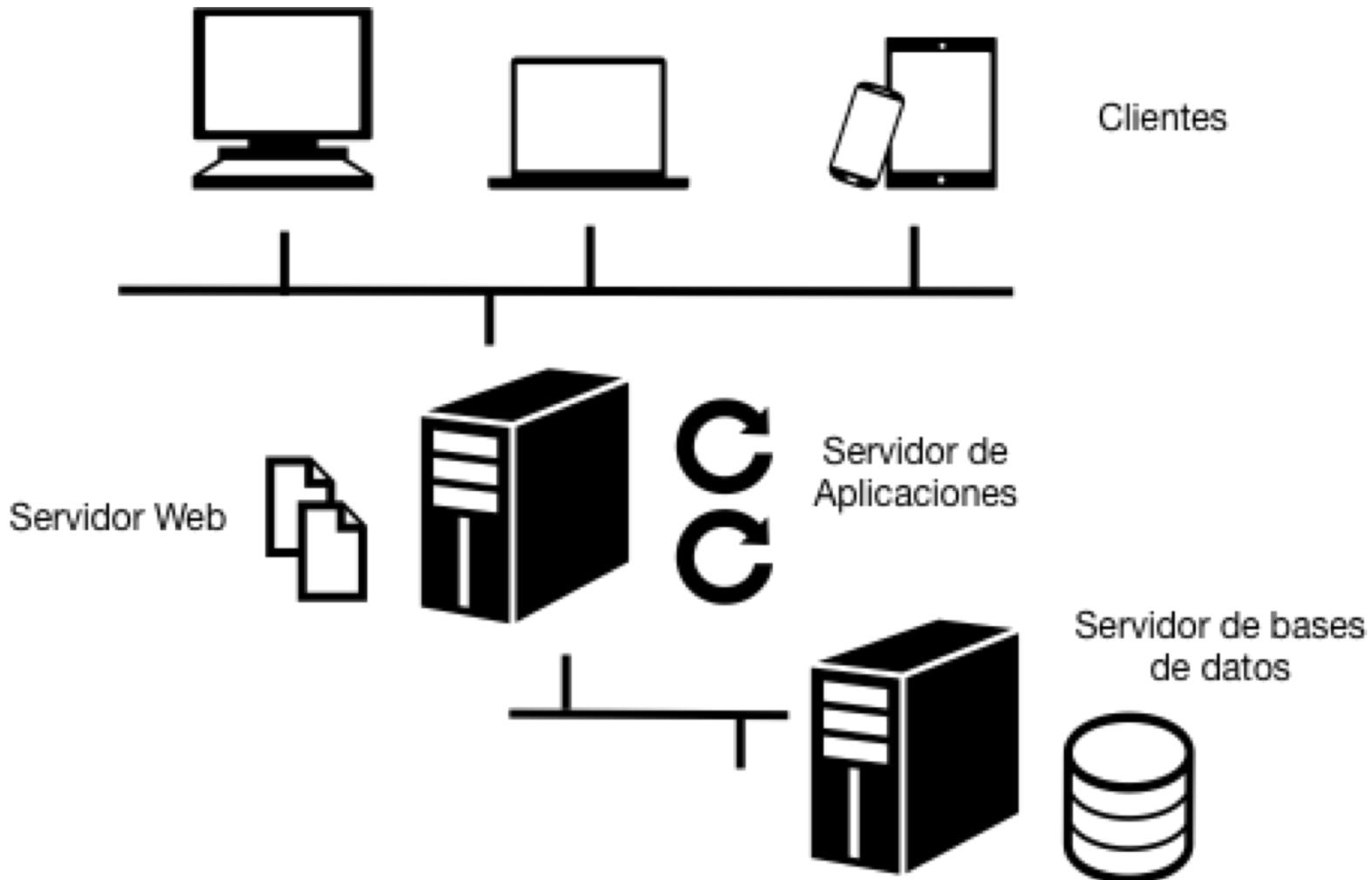
Solución en tres Niveles

- Se divide en
 - Cliente
 - Aplicación
 - Bases de Datos (BD)
- La lógica tiene su propio nivel
 - Esta invoca los datos de la BD cuando los requiere
-

Solución en tres Niveles (Cont)

- La puesta en marcha es más simple
 - La lógica reside en un servidor centralizado
 - Los cambios son más focalizados
 - Puede no necesitar modificaciones de la GUI
 - Esto es muy visto interfaces basadas en WEB
- El nivel de App es el más complejo
- La seguridad es igual de compleja (2 Tiers)
 - Puede requerir firewalls para usuarios externos
 - Con esto se separa el App del nivel de BD

Solución en tres Niveles (Cont)



Solución en cuatro Niveles

- Muy similar a la solución de 3 niveles
- Difiere en
 - Configuración
 - Seguridad
 - Escalabilidad
- Se separa en servidores aparte
 - Servidor Web
 - Servidor de Aplicaciones

Solución en cuatro Niveles (Cont)

- El servidor web se puede optimizar para
 - Manejar gran cantidad de conectores de red
 - Manejar una alta transferencia de I/O
- El servidor de aplicaciones se optimiza para
 - Procesar transacciones de negocio
 - Esto maximiza
 - La utilización del CPU
 - El uso de múltiples hilos de ejecución
 - La conexión con la base de datos

Solución en cuatro Niveles (Cont)

- Ventajas
 - Se puede colocar el servidor Web en red periférica
 - Se coloca el servidor de Apps en la intranet
 - Esto aumenta la seguridad
 - La información sensible no queda expuesta

Solución en cuatro Niveles (Cont)

