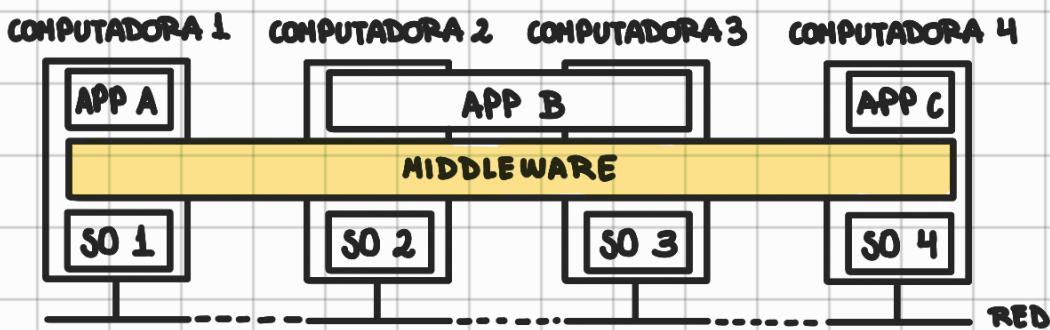


# Distribuidos - PEP 2

## Sistema Distribuido

- Colección de computadoras independientes que colaboran para mostrar al usuario un sistema único, que les permite interactuar con él sin importar dónde y cuándo.
- Capa alto nivel: Usuario y aplicaciones.
- Capa inferior: Sistemas Operativos y recursos básicos de comunicación.



**MIDDLEWARE** → Software que actúa como capa intermedia entre el sistema operativo y las aplicaciones, para facilitar la comunicación entre componentes distribuidos.

→ Funciones principales:

- Facilitar comunicación y transferencia de datos y mensajes.
- Gestión de la transacción para la consistencia de datos.
- Autenticación y autorización de componentes distribuidos.
- Gestión de recursos compartidos.
- Facilita la escalabilidad.
- Gestión y recuperación de fallas.
- Interoperabilidad, integración de tecnologías.

## Objetivos:

### ① Accesibilidad de recursos

- Los usuarios y aplicaciones deben acceder a los recursos remotos de manera fácil, compartida, controlada y eficiente.
- Se comparten recursos para ahorrar recursos.

### ② Transparencia

- Ocultar al usuario el hecho de que los procesos y recursos están físicamente distribuidos en varias computadoras.

Tipo	Oulta...
Acceso	Diferencias en representación y acceso a datos.
Ubicación	Localización de un recurso.
Migración	Recurso cambia de ubicación.
Reubicación	Recurso cambia de ubicación mientras se usa.
Concurrencia	Cuando un recurso es compartido.
Falla	Falla y recuperación de un recurso.

- Sin embargo, no siempre se puede asegurar transparencia, hay latencia y fallas que no se pueden ocultar. Además, afecta al rendimiento y mantener múltiples réplicas lleva tiempo.
- A veces, puede ser beneficioso utilizar servicios basados en ubicación para diferentes zonas horarias y dar retroalimentación de fallas.

### ③ Grado de Apertura

- Capacidad de ofrecer servicios de acuerdo con las reglas estándar que describen la sintaxis y la semántica.
- Las reglas se formalizan con protocolos.
- Los servicios se especifican a través de interfaces, definidos por IDL que capturan la sintaxis de los servicios.
- Debe cumplir con → Interfaces bien definidas.
  - Interoperabilidad. Dos sistemas distintos pueden coexistir y colaborar.
  - Portabilidad. Un sistema A se puede ejecutar en un sistema B.
  - Extensibilidad. Facilidad de agregar o reemplazar componentes sin afectar a otros.

### ④ Escalabilidad

- Los problemas de escalamiento aparecen como bajas en rendimiento ocasionados por la limitación de capacidad de servidores y redes.
- Escalable respecto a su tamaño (agregar usuarios y recursos).
  - Problema: Hay que rutear una enorme cantidad de mensajes, los cuales se deben recopilar y transportar, sobrecargando la red.

## • Escalable geográficamente (componentes lejanos entre si).

↳ Problema: Es imposible obtener la sincronización de todos los relojes, mientras mas grande sea el sistema, mayor será la incertidumbre.

↳ Problema: No se puede escalar si se basa en comunicación síncrona, porque bloquea un servicio hasta obtener una respuesta.

↳ Problema: Comunicación no fiable y virtualmente de punto a punto.

## • Escalable administrativamente (organizaciones diferentes).

↳ Problema: Políticas conflictivas con respecto al uso de recursos, administración y seguridad.

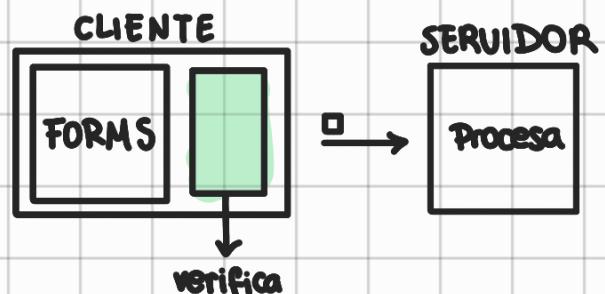
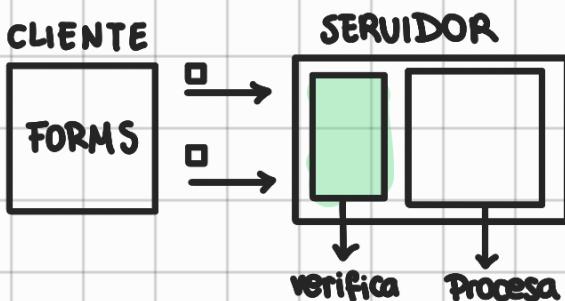
## • Técnicas de escalamiento

### 1) Ocultar latencias de comunicación para lograr escalabilidad geográfica.

#### COMUNICACIÓN ASÍNCRÓNICA

Evitar la espera de respuestas de peticiones remotas de servicios y realizar otras tareas útiles por parte de la máquina que hace la petición. Cuando llega la respuesta se invoca un handler.

En aplicaciones interactivas, no se puede utilizar comunicación asíncrona, entonces se reduce el volumen de la comunicación al llevar parte del cómputo a la máquina que hace la petición.



### 2) Distribución. Divide y dispersa un componente en partes más pequeñas. ej: DNS. Se evita que un servidor tenga que lidiar con todas las peticiones de dominio.

### 3) Replicar los componentes a lo largo del sistema.

Incrementa la disponibilidad, balancea la carga y mejora el rendimiento.

Hay problemas de consistencia por caché y replicas.

## ⑤ Dependencia

- Un componente puede ser dependiente de otro si lo necesita para prestar servicios.
- Requerimiento → Disponibilidad  
→ Fiabilidad  
→ Seguridad  
→ Mantenibilidad

## ⑥ Seguridad

- Un sistema distribuido que no es seguro, no es fiable.
- Se quiere confidencialidad, integridad, autenticación, autorización y confianza.
- Mecanismos de seguridad → Cifrar y decifrar con llaves de seguridad.
  - **CRPTOSISTEMA SIMÉTRICO**  
Las llaves de cifrado y descifrado son las mismas y deben mantenerse en secreto.
  - **CRPTOSISTEMA ASIMÉTRICO**  
Existe una llave pública y privada.

### Tipos de sistemas distribuidos

#### ① Cómputo

- **Clúster**: Programación paralela de alto rendimiento. Cada nodo tiene el mismo sistema operativo.
- **Grid**: Alto grado de heterogeneidad, no hace suposiciones de sistemas operativos ni tecnologías.

#### ORGANIZACIÓN VIRTUAL

- **Fabricación**: Interfaces para recursos locales.
- **Conectividad**: Protocolo de comunicaciones.
- **Recursos**: Administrador de un solo recurso.
- **Colectiva**: Manipular el acceso a múltiples recursos.
- **Aplicación**: App que operan dentro de una organización virtual



#### ② Información

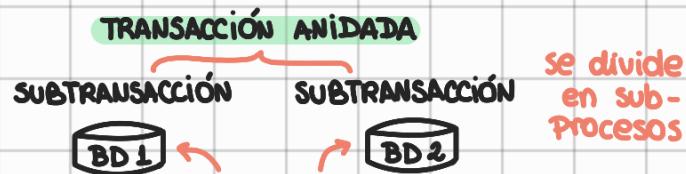
- Una app en la red se ejecuta en un servidor y pone a disposición servicios a clientes remotos.
- Permite la comunicación directa entre aplicaciones.

## ♥ Sistema de Procesamiento de transacciones

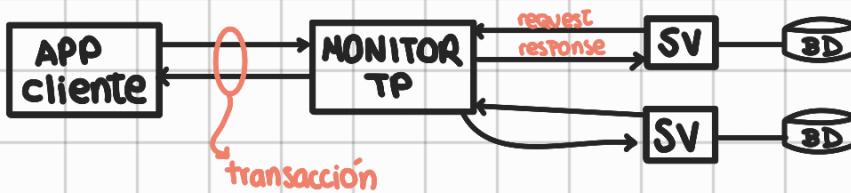
Las operaciones se llevan a cabo en forma de transacciones.

Las llamadas a procedimientos remotos (RPC) se encapsulan en una transacción

- atómicas
- consistentes
- aisladas
- durables



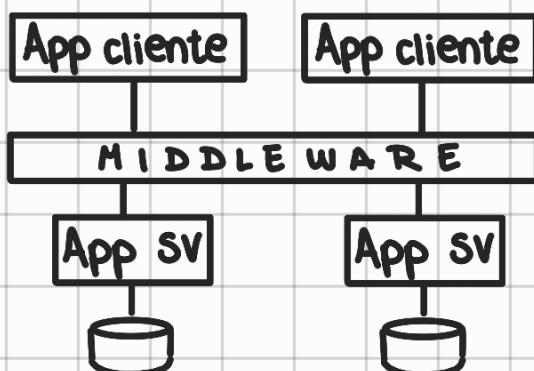
## Monitor de Procesamiento de transacciones



Permite que una aplicación acceda a múltiples SV y BD.

## ♥ Integración de aplicaciones empresariales

Componentes con aplicaciones independientes de BD necesitan comunicarse de manera directa, sin request - response.



### Llamadas a Procedimientos Remotos

- **RPC**: Un componente puede enviar una petición que se empaca como msje, se envía al otro componente y retorna una respuesta.

### Invocaciones a Procedimientos Remotos

- **RMI**: RPC orientado a objetos, en lugar de aplicaciones.

## Middleware orientado a mensajes

- **MOM**: No necesita saber dónde está el otro componente, las aplicaciones envían mensajes a puntos de contacto lógicos.

Sistemas de publicación - suscripción.

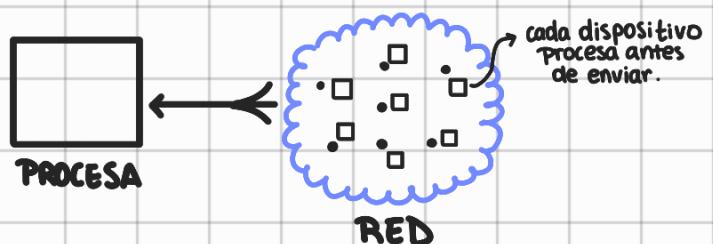
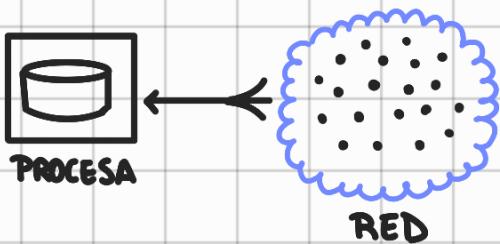
## ③ Omnipresentes

- El sistema se integra de forma natural en el entorno del usuario.

- Elementos básicos → Dispositivos distribuidos en la red y accesibles de forma transparente.

- Interacción discreta entre usuarios y dispositivos.
- El sistema conoce el contexto del usuario.
- Dispositivos funcionan de forma autónoma.
- El sistema puede gestionar una amplia gama de acciones.

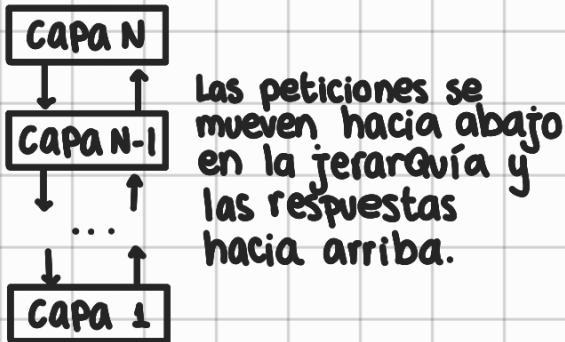
- **Ubicuos**: Interacción continua entre el sistema y el usuario.
- **Móviles**: Intrínsecamente móviles.
- **Redes de sensores**: Detección y colaboración con el entorno.



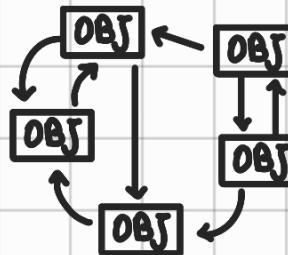
## Arquitectura

- Cómo se organizan los componentes, recordando que el objetivo es separar las aplicaciones de las plataformas subyacentes mediante una capa de middleware.
- **Conector**: Mecanismo para mediar la comunicación, coordinación o cooperación entre componentes.
- **Componentes**: Unidad modular con las interfaces requeridas bien definidas. Es reemplazable dentro de su ambiente.

### En Capas

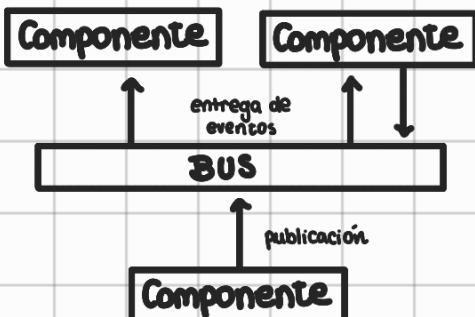


### Basada en Objetos



Los objetos o componentes se conectan a través de llamadas a procedimientos remotos. cliente - servidor.

### Basado en Eventos

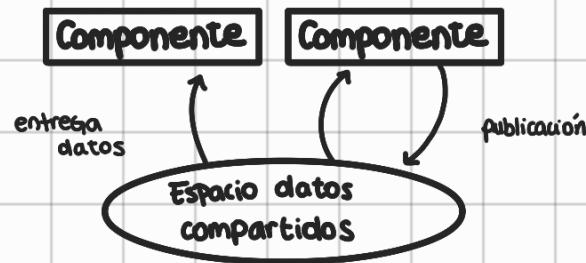


Los procesos se comunican a través de la propagación de eventos.

El middleware se encarga de que los procesos suscritos reciban publicaciones.

No se refencian explícitamente.

### Espacio de Datos Compartidos



No es necesario que ambos estén activos cuando se lleva a cabo la comunicación.

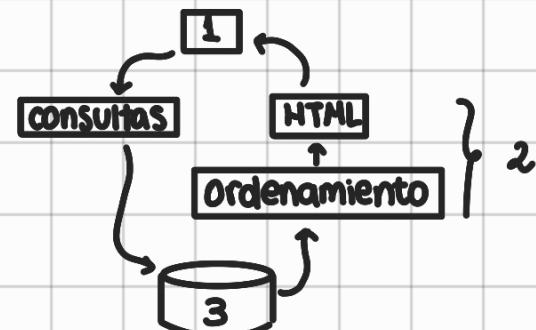
## Arquitecturas Centralizadas

- Comportamiento solicitud - respuesta.
- Servidor: Implementa un servicio.
- Cliente: Sigue un servicio de un servidor.



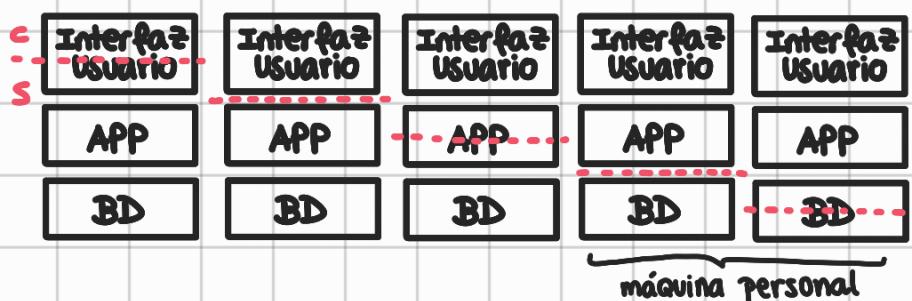
## Aplicación de Capas

- Nivel de interfaz de usuario. ①
- Nivel de Procesamiento. ②
- Nivel de datos. ③



## Arquitectura Multinivel

- Distribuye físicamente una aplicación cliente - servidor a través de varias máquinas.
- Máquina cliente que implementa el nivel de usuario.
- Máquina servidor que realiza el resto.



## Arquitectura Descentralizada

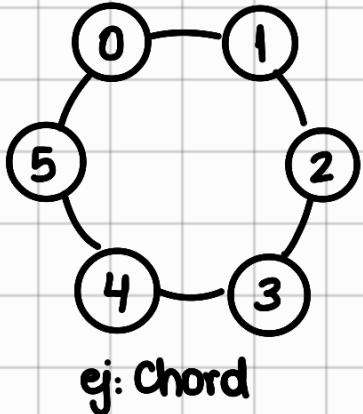
- Utiliza distribución horizontal, es decir, los procesos se dividen físicamente en partes lógicas equivalentes, pero cada parte opera en su propio espacio del conjunto de datos.

## Peer to Peer

- Cada proceso actúa como cliente o servidor.
- Un proceso no puede comunicarse directamente con otro proceso cualquiera, necesita enviar mensajes a través de los canales de comunicación disponibles.

## P2P Estructurados

- Organiza los procesos con una tabla hash distribuida DHC, que asigna un ID aleatorio.
- Para buscar un elemento, se debe enrutar una petición hacia el nodo responsable.



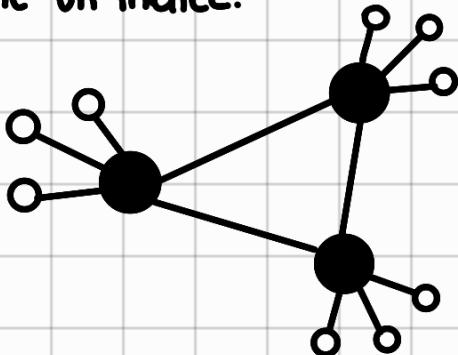
Organiza los nodos en un anillo, de tal forma que un elemento con llave  $k$  se mapea hacia el nodo con el ID más pequeño  $ID \geq k$ . El nodo se conoce como sucesor de la llave  $k$  y se devuelve su dirección de red.

## P2P no estructurados

- Cada nodo mantiene una lista de vecinos que se construye de manera aleatoria, al igual que la posición de los datos en los nodos.
- Cuando un nodo necesita localizar un elemento, inunda la red en la búsqueda.
- El punto crucial es la construcción de una nueva vista parcial con c entradas, luego hay que descartar en lo posible la mayor cantidad de entradas viejas.
- Mientras más alto sea un grado de entrada de un nodo  $P$ , más alta será la probabilidad que algún otro nodo decida contactar a  $P$ , lo que produce desequilibrio en la carga de trabajo.

## Superpeer

- Localizar elementos puede ser complejo cuando la red crece, por la acumulación de solicitudes, por lo que utilizan nodos que mantienen un índice.



Toda comunicación desde y hacia un punto regular procede a través del superpunto asociado al punto.

El superpunto sabrá dónde encontrar a sus puntos.

