



# Tema I - Introducción



Tecnologías de los Sistemas de Información en la Red



# Objetivos

---

- ▶ Entender por qué todo sistema que utilice una red de intercomunicación es un sistema distribuido.
- ▶ Identificar qué es un sistema distribuido, por qué son relevantes y cuáles son sus aplicaciones principales.
- ▶ Conocer algunos ejemplos de sistemas distribuidos.
- ▶ Estudiar la evolución de los sistemas distribuidos escalables e identificar la computación en la nube (“*cloud computing*”) como la etapa actual de esta evolución.



# Índice

---

1. Concepto de sistema distribuido
2. Relevancia
3. Áreas de aplicación
4. Computación en la nube
5. Paradigmas de programación
6. Conclusiones



# I. Concepto de sistema distribuido

---

- ▶ Conjunto de agentes autónomos
  - ▶ Cada agente es un proceso secuencial que avanza a su propio ritmo.
- ▶ Los agentes interactúan. Opciones:
  - ▶ Intercambio de mensajes
  - ▶ Memoria compartida
- ▶ Los agentes tienen su propio estado independiente
- ▶ Hay algún objetivo común en esta cooperación
  - ▶ Mediante el que se podrá evaluar el comportamiento global del “sistema”.
- ▶ En la práctica, un sistema distribuido es un sistema en red.



# Índice

---

1. Concepto de sistema distribuido
2. Relevancia
3. Áreas de aplicación
4. Computación en la nube
5. Paradigmas de programación
6. Conclusiones



## 2. Relevancia

---

- ▶ **Sistemas distribuidos**
  - ▶ Área en evolución desde sus orígenes
  - ▶ Rama de los sistemas concurrentes
    - ▶ Ampliamente estudiada por su utilidad en el diseño de sistemas de tiempo compartido.
    - ▶ CSD proporcionó la base para familiarizarnos con múltiples aspectos de los sistemas concurrentes.
  - ▶ Reforzada con la evolución de las redes de ordenadores.
    - ▶ ¿Cómo conseguir que todos esos ordenadores hagan algo globalmente útil?



## 2. Relevancia

### ▶ Aspectos relevantes (presentados en los 80)

#### a) Mejora del rendimiento

- ▶ Seleccionar una actividad (problema) compleja, dividirla en tareas (subproblemas), asignar cada tarea a un ordenador diferente.

#### b) Mayor disponibilidad. Idea básica:

- ▶ Si un ordenador se avería, todavía habrá otros ordenadores capaces de ejecutar las tareas del que ha fallado.

#### c) Compartición de recursos

- ▶ Un ordenador puede tener recursos (p.ej., impresoras, discos...) que otros ordenadores no tengan (y que no necesiten tener).
- ▶ Debe ser posible el acceso a recursos desde cualquier ordenador.



## 2. Relevancia

---

- ▶ Todas esas razones son todavía válidas porque el entorno de computación actual **ESTÁ** distribuido e interconectado
  - ▶ Infinidad de “ordenadores” conectados
  - ▶ Infinidad de servicios remotos
    - ▶ Accedidos como recursos compartidos
    - ▶ Todos conocemos y utilizamos la web
- ▶ Desafíos
  - ▶ Aprovechar la conectividad para obtener resultados útiles
  - ▶ Crear subsistemas capaces de proporcionar servicios robustos
    - ▶ ¿Cómo se las apaña Google para implantar su servicio de búsqueda?
    - ▶ ¿Cómo gestiona Dropbox el uso compartido de ficheros por parte de millones de usuarios?
    - ▶ ¿Cómo distribuir entre millones de voluntarios la simulación de nuevos fármacos contra el cáncer?





# Índice

---

1. Concepto de sistema distribuido
2. Relevancia
3. Áreas de aplicación
4. Computación en la nube
5. Paradigmas de programación
6. Conclusiones



### 3. Áreas de aplicación

---

- ▶ Las más destacables son:
  1. *World Wide Web*
  2. Redes de sensores
  3. *Internet of Things*
  4. Computación cooperativa
  5. *Clusters* altamente disponibles
- ▶ Las tratamos a continuación...



## 3.1 Aplicación a WWW

---

- ▶ Basada en el modelo cliente/servidor.
- ▶ El servidor espera peticiones de documentos.
  - ▶ Las peticiones implican la lectura o modificación de un documento.
- ▶ Los clientes son los navegadores web, que envían y reciben documentos
  - ▶ Los navegadores analizan el documento buscando metadatos.
  - ▶ Los enlaces son un caso particular de metadatos que apunta a otros documentos.
    - ▶ Los documentos pueden estar en otro servidor.
- ▶ Paradigma simple y potente
  - ▶ Diseñado inicialmente para compartir documentos.
  - ▶ Extendido para permitir que las peticiones sobre documentos se convirtieran en peticiones de servicio
    - ▶ Los “documentos” retornados incluyen el resultado de la petición efectuada.



## 3.2 Aplicación a redes de sensores

---

- ▶ Han surgido gracias al coste descendente de los equipos.
- ▶ Mini-ordenadores de propósito específico
  - ▶ “Motes”
- ▶ Empotrados en dispositivos de uso cotidiano
  - ▶ Pej., en algunos electrodomésticos
- ▶ Contienen sensores
  - ▶ Humedad, temperatura, consumo eléctrico...
- ▶ Amplio rango de aplicaciones potenciales
  - ▶ Vigilancia
  - ▶ Detección de desastres (químicos, biológicos...)
  - ▶ Monitorización del consumo eléctrico
  - ▶ ...



## 3.3 Aplicación a la “*Internet of Things*”

---

- ▶ Motivación: facilitar la conectividad e interoperabilidad de todos los dispositivos
  - ▶ Generalización de las redes de sensores
    - ▶ Todos los dispositivos pueden interactuar entre sí
    - ▶ Los dispositivos pueden alterar su entorno físico
  - ▶ Se abren nuevos escenarios
    - ▶ Ciudades inteligentes
    - ▶ Automatización de múltiples procesos (construcción, fabricación...)
    - ▶ Cuidado médico informatizado
    - ▶ ...

## 3.3 Aplicación a la “Internet of Things”





## 3.4 Aplicación a la Computación cooperativa

---

- ▶ La mayor parte de los recursos computacionales se infrautilizan
  - ▶ Los ordenadores personales pasan muchas horas diarias sin hacer nada
- ▶ Muchos problemas científicos e ingenieriles pueden dividirse en piezas menores (tareas)
  - ▶ Cada tarea puede resolverse en un intervalo breve.
  - ▶ Los resultados de cada tarea pueden componerse para construir la solución del problema completo.
- ▶ Los servidores pueden obtener una instancia de esos problemas
  - ▶ El servidor crea un conjunto de tareas
- ▶ Los ordenadores con acceso a Internet pueden suscribirse para recibir tareas que resolver
  - ▶ Instalan un cliente especial: el “runtime” para ejecutar tareas
  - ▶ El cliente se registra en el servidor
- ▶ El servidor distribuye tareas entre los clientes registrados y recoge sus resultados



## 3.5 Aplicación a *clusters* altamente disponibles

- ▶ Hasta ahora hemos presentado áreas de aplicación dedicadas a la cooperación y la compartición de recursos.
- ▶ Hecho:
  - ▶ Los dispositivos fallan. Los ordenadores son dispositivos. Ellos fallan en algún momento con probabilidad 100%.
- ▶ Hecho:
  - ▶ No todos los dispositivos de un sistema fallan a la vez.
  - ▶ ¿Por qué podría pasar esto?
- ▶ Algunos entornos necesitan un alto nivel de disponibilidad
  - ▶ Bancario
  - ▶ Empresarial
  - ▶ Asistencia médica
  - ▶ ...
- ▶ Conviene tener más de un dispositivo para soportar las situaciones de fallo.





## 3.5 Aplicación a *clusters* altamente disponibles

---

- ▶ **Cluster altamente disponible:**
  - ▶ Conjunto de ordenadores con programas servidores de los que los clientes dependen en todo momento.
  - ▶ Típicamente mantienen un conjunto de datos crítico.
  - ▶ Diseñados con protocolos específicos para soportar fallos en los ordenadores.
  - ▶ Dos aspectos principales:
    - ▶ Mantener la integridad de la información gestionada
    - ▶ Mantener la disponibilidad de los servidores



## 3.5 Aplicación a *clusters* altamente disponibles

- ▶ Principal tendencia actual para construir y facilitar servicios
  - ▶ Hechos aceptados:
    - ▶ Se infrautiliza la potencia de cómputo con las arquitecturas tradicionales
      - ▶ Ya se ha discutido previamente
    - ▶ Resulta caro establecer centros de cómputo para empresas, con todas las aplicaciones que estas requieren:
      - ▶ Adquirir programas y equipos
      - ▶ Sueldos de los ingenieros que administren estas aplicaciones y equipos.
      - ▶ Coste de la energía eléctrica
        - Todavía más caro si consideramos la infrautilización
- ... esto conduce a la computación en la nube (“*cloud computing*”  
→ CC)...



# Índice

---

1. Concepto de sistema distribuido
2. Relevancia
3. Áreas de aplicación
4. Computación en la nube
5. Paradigmas de programación
6. Conclusiones



## 4. Computación en la nube (CC: *cloud computing*)

---

► Hablaremos de:

1. Programas y servicios
2. Roles en el ciclo de vida de un servicio
3. Evolución de los servicios software
  - a) Mainframes
  - b) Ordenadores personales
  - c) Centros de cómputo empresariales
  - d) SaaS
  - e) IaaS
  - f) SaaS sobre IaaS
  - g) PaaS
4. Resumen



## 4.1 CC: Programas y servicios

---

- ▶ **Objetivo general del CC:**
  - ▶ Convertir la creación y explotación de los servicios “software” en algo más sencillo y más eficiente.
- ▶ **Un hecho aceptado y obvio:**
  - ▶ Los programas siempre se han desarrollado para ofrecer algún tipo de servicio.
  - ▶ Con la ayuda de los ordenadores, por supuesto.
- ▶ **La evolución de la industria informática ha ocultado parcialmente este hecho:**
  - ▶ La industria de los ordenadores personales ha impuesto un modo particular de interacción de los usuarios con sus ordenadores.



## 4.2 Roles en el ciclo de vida de un servicio

---

Consideramos estos 4 roles:

- a) **Desarrollador**
  - ▶ **Implanta los componentes de las aplicaciones**
- b) **Proveedor de servicios**
  - ▶ **Decide las características del servicio, los componentes que lo constituyen y cómo debe ser configurado y administrado**
- c) **Administrador del sistema**
  - ▶ **Se encarga de que cada pieza de *software* y *hardware* esté en su lugar apropiado y adecuadamente configurado.**
- d) **Usuario**
  - ▶ **Accede al servicio**



## 4.3 Evolución de los servicios *software*

### a) *Mainframes*

---

- ▶ La *administración del sistema* está realizada por especialistas
- ▶ Muy pocos focos de contención
  - ▶ Sistemas con una reducida base de usuarios
- ▶ Uso eficiente de los equipos
  - ▶ Compartidos por múltiples usuarios
  - ▶ Coste bajo para cada usuario
    - ▶ Coste de adquisición soportado por el propietario del equipo: institución
- ▶ Los usuarios siguen roles mixtos
  - ▶ Muchos fueron desarrolladores
  - ▶ Muchos fueron también sus propios proveedores de servicios
    - ▶ Con los programas que ellos desarrollaron
    - ▶ Con los programas desarrollados por otros
- ▶ Los usuarios estaban implicados en demasiados detalles de la gestión de los servicios que ellos mismos debían utilizar



## 4.3 Evolución de los servicios *software*

### b) Ordenadores personales

---

- ▶ Los ordenadores personales fueron el resultado de la tendencia a una mayor potencia de cómputo en cada equipo
  - ▶ Los usuarios ya no necesitaban acceder a un “*mainframe*” en un centro de cálculo.
- ▶ Se elimina la contención
  - ▶ Uno de los principales argumentos de venta de este paradigma
- ▶ Uso deficiente de los recursos: el ordenador se infrautiliza
- ▶ Inversión a realizar: coste de la compra
- ▶ Se racionaliza el rol de desarrollador
  - ▶ Empresas especializadas construyen y comercializan los programas
- ▶ Pero todavía se exige que el usuario desempeñe varios roles:
  - ▶ Proveedor de servicios
    - ▶ Debe seleccionar qué programas necesitará para realizar sus tareas
  - ▶ Administrador de su ordenador personal
- ▶ Entorno demasiado complejo para la mayoría de los usuarios





## 4.3 Evolución de los servicios *software*

### c) Centros de cómputo empresariales

---

- ▶ Implantados mediante *clusters* altamente disponibles
- ▶ Características similares al entorno de ordenadores personales
  - ▶ El usuario es ahora la empresa
  - ▶ Personal especializado que sigue compartiendo los roles de administrador de sistema y proveedor de servicios: Coste muy alto
  - ▶ En ocasiones se añade el rol de desarrollador de programas internos
- ▶ Variante basada en mantener estos programas en centros de datos externos
  - ▶ Evita el coste de adquisición de los equipos
  - ▶ Reduce y externaliza el coste de administración y mantenimiento de los equipos
  - ▶ Evita el coste fijo de consumo eléctrico
  - ▶ La gestión de los costes informáticos resulta más sencilla



## 4.3 Evolución de los servicios *software*

### d) *Software as a Service* (SaaS)

---

- ▶ Se accede a los servicios a través de la red
  - ▶ Mediante un navegador web
- ▶ Separación clara del rol de usuario
  - ▶ El servicio está definido por una tercera parte: el proveedor de servicios
- ▶ No queda tan clara la separación de los demás roles
  - ▶ Los programas suelen ser desarrollados inicialmente por el proveedor
  - ▶ Todas las tareas de administración suelen recaer en el proveedor
    - ▶ Incluyendo la administración de equipos en los centros de datos
    - ▶ Incluyendo la administración de los programas instalados en estos equipos
- ▶ Inicialmente, surgen algunas ineficiencias:
  - ▶ Falta de flexibilidad en la distribución de los equipos
    - ▶ Conlleva que el proveedor se ciña a cierto uso de los recursos
    - ▶ Limita la compartición de recursos
  - ▶ Contención limitada: se reservan recursos para la demanda esperada



## 4.3 Evolución de los servicios *software*

### d) *Software as a Service* (SaaS)

---

- ▶ Factores que condujeron hacia los sistemas SaaS:
  - ▶ Mejora de las tecnologías de red
    - ▶ Mayor ancho de banda
    - ▶ Menor retardo
  - ▶ Capacidad de los centros de datos existentes
    - ▶ Posibilitó la oferta de servicios a usuarios externos
  - ▶ Mejoras en la tecnología de los navegadores web
    - ▶ Tipificadas en el término “Web 2.0”
    - ▶ Navegadores capaces de ejecutar localmente interacciones complejas
      - Permiten interfaces de usuario más atractivas
      - Escalabilidad mejorada: Menor carga en el servidor



## 4.3 Evolución de los servicios *software*

### e) *Infrastructure as a Service* (IaaS)

---

- ▶ Facilita la capacidad para asignar o redistribuir los recursos de cómputo y de red bajo petición.
  - ▶ Peticiones vía API a un servicio (el servicio IaaS)
  - ▶ Posibilidad de cargar imágenes de SO sobre esos ordenadores
  - ▶ Posibilidad de solicitar capacidades concretas para los ordenadores y los recursos de red
- ▶ Posible gracias a la tecnología de virtualización de equipos
  - ▶ La asignación de recursos de cómputo (virtuales) es fácil y rápida
  - ▶ Fácil configuración de la capacidad de los recursos de cómputo
  - ▶ Resulta fácil la instalación de una imagen de sistema sobre una máquina virtual



## 4.3 Evolución de los servicios *software*

### f) SaaS sobre IaaS

- ▶ IaaS introduce un modelo de “pago por uso”
  - ▶ Una característica central de la computación en la nube
- ▶ Facilita la creación de SaaS que se adaptan a la carga generada por sus usuarios
  - ▶ Cuanto mayor sea esa carga, se solicitará un mayor número de recursos a la infraestructura
    - ▶ *Elasticidad*: otra característica central de los sistemas Cloud
  - ▶ Traslada el modelo de “pago por uso” a los sistemas SaaS
    - ▶ Los usuarios de un sistema SaaS también pagan según su utilización del servicio
- ▶ Obliga a un uso eficiente de los recursos por parte del proveedor SaaS
  - ▶ La mayoría de los costes son variables
  - ▶ No hay costes directos por reservar cierta capacidad (compra o compromiso de pago)
  - ▶ Lo que se ahorre beneficiará al usuario SaaS: mercado competitivo de servicios



## 4.3 Evolución de los servicios *software*

### f) SaaS sobre IaaS

- ▶ Los proveedores IaaS toman los riesgos de la inversión directa (compra de los recursos físicos)
  - ▶ Esperan una gran población de proveedores SaaS
    - ▶ A su vez, facilitando servicios a un alto número de usuarios SaaS
    - ▶ Gran demanda de recursos virtualizados
- ▶ El proveedor SaaS todavía desempeña varios roles
  - ▶ Proveedor de servicios *software* (su rol natural)
  - ▶ Debe gestionar la asignación de recursos *hardware*
  - ▶ Debe gestionar las imágenes de sistema a instalar, sus actualizaciones y la base de programas a utilizar sobre esos sistemas
  - ▶ Debe implantar su propia estrategia de gestión de servicios
    - ▶ Mecanismos de monitorización
    - ▶ Mecanismos de actualización



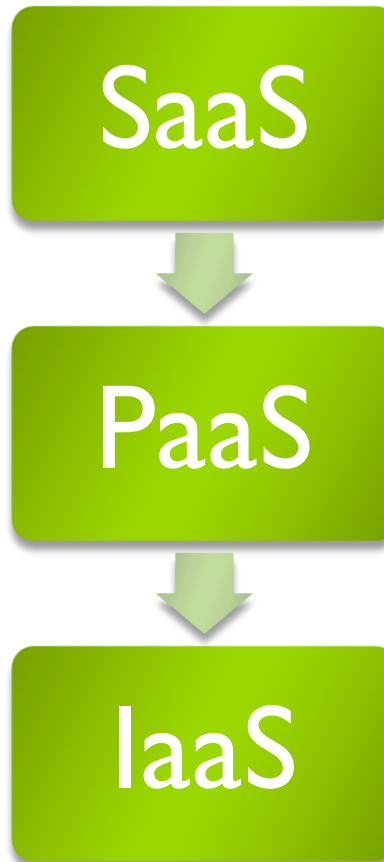
## 4.3 Evolución de los servicios *software*

### g) *Platform as a Service* (PaaS)

---

- ▶ Debería eliminar cualquier tarea extraña para los proveedores SaaS
  - ▶ Todavía en sus inicios, desafortunadamente.
- ▶ Será equivalente a un sistema operativo
  - ▶ Especifica un modelo de servicios sobre el que basar la especificación de los SaaS y el desarrollo de sus componentes *software*
  - ▶ Incluye los aspectos siguientes
    - ▶ Modelos de configuración y de gestión del ciclo de vida (incluyendo las relaciones de dependencia entre componentes)
      - Mecanismos de composición, configuración, despliegue y actualización
    - ▶ Modelo de rendimiento
      - Monitorización automática de parámetros relevantes
      - Expresión de puntos de elasticidad
      - Reconfiguración automatizada en función de la carga

### Estructura ideal (en niveles)







## 4.4 Resumen

- ▶ La computación en la nube (CC) se centra en la eficiencia y la facilidad de uso:
  - ▶ **Compartición eficiente de los recursos**
    - ▶ Consumir solo lo que se necesite
    - ▶ Pagar solo por lo que se ha utilizado
  - ▶ **Adaptación sencilla a una cantidad de usuarios variable**
  - ▶ **Facilitar formas sencillas para desarrollar y proveer un servicio**
- ▶ Tres niveles de servicios en la nube han sido identificados:
  - ▶ ***Software as a Service (SaaS)***
    - ▶ Su objetivo es facilitar aplicaciones como servicio a un gran número de usuarios
  - ▶ ***Platform as a Service (PaaS)***
    - ▶ Aconsejable para automatizar la gestión de recursos para los SaaS y la fácil creación y despliegue de estos servicios
  - ▶ ***Infrastructure as a Service (IaaS)***
    - ▶ Proporciona elasticidad para los sistemas SaaS
- ▶ Desde la perspectiva de los usuarios, el CC es similar a un regreso a la era de los “*mainframes*”.



# Índice

---

1. Concepto de sistema distribuido
2. Relevancia
3. Áreas de aplicación
4. Computación en la nube
5. Paradigmas de programación
6. Conclusiones



## 5. Paradigmas de programación

---

- ▶ La forma más común de organizar un sistema distribuido se basa en convertir cada proceso en un “servidor”
  - ▶ Recibe peticiones, las procesa y retorna respuestas
- ▶ Los servidores, a su vez, solicitan servicio a otros servidores
  - ▶ Pueden necesitar esos servicios para completar una petición recibida
- ▶ Para ser escalable, un servidor no debe suspenderse mientras gestione una petición
  - ▶ Debe ser capaz de aceptar otras peticiones



## 5.1. Servidores concurrentes (estado compartido)

- ▶ Programas concurrentes (con múltiples hilos)
  - ▶ Cada petición es servida por su propio hilo
  - ▶ Todos los hilos comparten un estado global
  - ▶ Se utilizan mecanismos de control de concurrencia para garantizar atomicidad
- ▶ Ventajas
  - ▶ Los hilos pueden suspenderse esperando peticiones, sin suspender a todo el proceso servidor
- ▶ Inconvenientes
  - ▶ La programación multi-hilo tiene sus propias sobrecargas
    - ▶ Necesita mecanismos de control de concurrencia. Implica suspensión.
  - ▶ La programación concurrente con memoria compartida resulta...
    - ▶ difícil de implantar sin errores
    - ▶ difícil de razonar sobre cómo se comporta (y justificar su corrección)
- ▶ Entornos predominantes:
  - ▶ Java
  - ▶ .NET



## 5.2. Servidores asincrónicos

- ▶ La programación asincrónica (o programación dirigida por eventos)...
  - ▶ corresponde fielmente al modelo de programación guarda/acción.
  - ▶ genera programas con múltiples actividades, pero...
    - ▶ el estado compartido nunca podrá ser accedido concurrentemente por esas actividades
- ▶ Los “eventos” son las “guardas”
- ▶ Las acciones se establecen como “*callbacks*” de los eventos
  - ▶ Para facilitar la programación se relacionan dinámicamente las acciones/guardas.
  - ▶ Al implantar las acciones, ciertos mecanismos del lenguaje de programación permiten establecer qué estado se verá afectado por ellas.
    - ▶ Se reduce la complejidad para “preparar” el estado que relacionará acciones internas.
- ▶ Las acciones preparadas para ejecución se “encolan”
  - ▶ Se ejecutarán siguiendo el orden FIFO de la cola



## 5.2. Servidores asincrónicos

### ▶ Ventajas

- ▶ La complejidad en la gestión de estado compartido desaparece
  - ▶ Pero debe considerarse el orden de activación (es decir, de “encolado”) para evitar sorpresas
- ▶ Menor sobrecarga, pues no se necesita un soporte multi-hilo
  - ▶ Mayor escalabilidad
- ▶ Modelo más próximo a la forma real de trabajo en un sistema distribuido: dirigido por eventos
  - ▶ Resulta más fácil razonar sobre qué está ocurriendo en cada momento

### ▶ Inconvenientes

- ▶ Se necesita una gestión adecuada del estado al implantar las acciones
- ▶ Se necesita que todo el entorno sea asincrónico, no sólo la comunicación entre procesos
  - ▶ Los servicios del sistema operativo deben ser asincrónicos, para evitar suspensiones

### ▶ Entornos predominantes con soporte nativo en el lenguaje:

- ▶ NodeJS
- ▶ Async .NET



# Índice

---

1. Concepto de sistema distribuido
2. Relevancia
3. Áreas de aplicación
4. Computación en la nube
5. Paradigmas de programación
6. Conclusiones



## 6. Conclusiones

---

- ▶ Los sistemas en red son sistemas distribuidos
  - ▶ La mayoría de la computación se desarrolla hoy en día sobre la red
    - ▶ Por tanto, es distribuida
  - ▶ Un diseño y desarrollo adecuados requieren un profundo conocimiento de la programación concurrente y de las arquitecturas utilizadas
- ▶ Amplio conjunto de áreas de aplicación ya en explotación
- ▶ La computación en la nube como última etapa importante en la evolución de la computación
  - ▶ Caracterizada por la eficiencia en el uso de los recursos
  - ▶ Con un modelo de acceso de “pago por uso”
  - ▶ Elasticidad y escalabilidad como objetivos principales
- ▶ Dos paradigmas de programación para desarrollar servicios distribuidos:
  - ▶ Servidores concurrentes (multi-hilo)
    - ▶ Deben gestionar condiciones de carrera. Puede haber bloqueos.
  - ▶ Servidores asincrónicos
    - ▶ Orientación a eventos. Fácilmente escalables.