# Características de un sistema distribuido

## Heterogeneidad

* **Interoperabilidad**

Que un SD sea compatible o que este compuesto por diferentes lenguajes, SO, plataformas

## Extensibilidad

* **Ampliar a otros campos**

La agencia puede aplicar su modelo de negocio como reservar restaurantes

* **Añadir nuevo hardware**

Nuevos computadores/nodos al sistema

## Escalabilidad

## Seguridad

## Concurrencia y sincronización

Esta es la parte de Transacciones, se expone los problemas y se propone soluciones (sin hablar de algoritmos)

* **Concurrencia**
  + Que dos elementos accedan a la vez al mismo recurso
  + Prioridades
  + Garantizar el acceso concurrente
* **Sincronización**

Lo de la parte de la api-transacciones

* + Que mientras se realiza una operación puede suceder un error
* **Servidores de tiempo**

Se puede mencionar para que así todo el SD tenga la misma hora y se sincronice mejor

## Tolerancia a fallos

Se habla el tema de las caídas de sistemas

* **Uso de SAIs**
* **Clúster**

Para replicar los servidores para que siempre haya un servidor en funcionamiento

* **Duplicación en la línea de comunicación**
* **¿Qué pasa si se cae un sistema?**

## Transparencia

* **Movilidad**
* **Acceso**
* **Tolerancia a fallos**

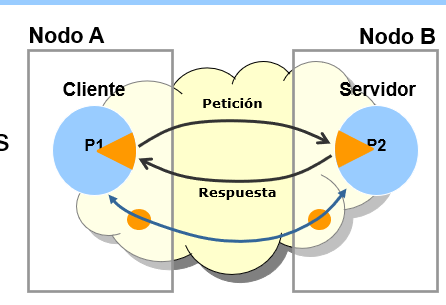
El usuario no es consciente de cuando ha habido un error de SW/HW

* **Localización**
* **Escalabilidad**

# Modelos arquitectónicos

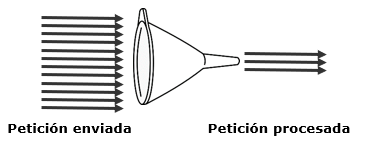
## Cliente/Servidor

* **Gestión centralizada**

El servidor se encarga de gestionar y domina la conexión y provee el servicio

* **Uso**

HTTP, DNS, FTP, SMTP

* **Inconvenientes**

Cuello de botella

Caída o colapso del servidor

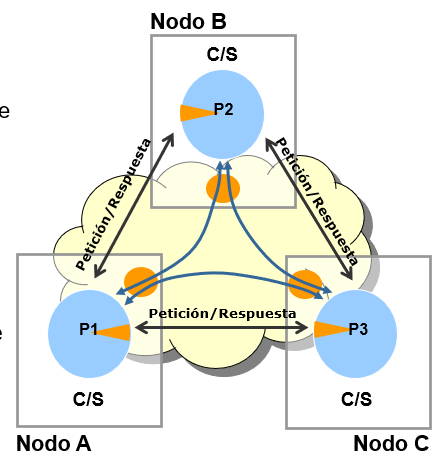
* **Concurrencia**

No es concurrente a no ser que se utilice un cluster, semáforos

O bien colas, almacenan las peticiones aun no procesadas, puede ser de prioridad o no, ()

* Uso de tokens de sesión
* Simplifica la sincronización, fácil de implementar en servicios de red

## peer-to-peer

* **Gestion no centralizada**

Todos los pares (elementos) son iguales actúan como cliente y servidor a la vez

Gestión completamente distribuida => menor control

* **Poco seguro**

No se controla quien da la información

* **Conexión Punto a punto**
* **Escalable**

Se añade un nodo nuevo y ya, Conti más nodos mejor funciona

* **Usos**

mensajería instantánea, compartición de archivos, vídeo conferencia, trabajo colaborativo blockchain

* **Tipos**
  + **Centralizada**

Realizas la petición a un servidor, este realiza la búsqueda en la red PtP y te responde

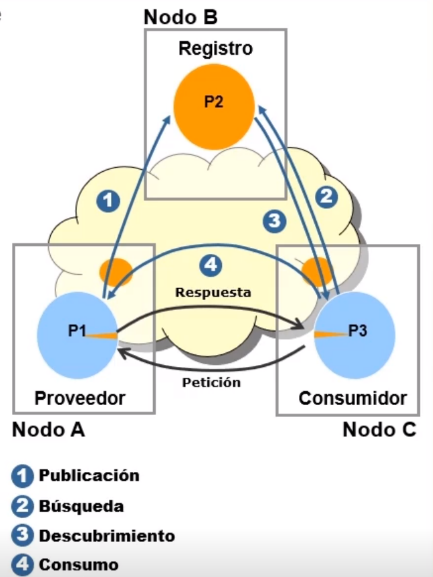
* + **Hibrida**

Realizas la petición a un servidor y este de dice que nodos te pueden responder e inmediatamente te conectas a estos (utorrent)

* + **Descentralizada**

Realizas la petición a la red y te van respondiendo los nodos

## SOA (ARQUITECTURA ORIENTADA A SERVICIOS)

* **Apto para sistemas desacoplados**

Los proveedores pueden tener otra arquitectura diferente de la agencia

* **Principios**
  + **Localización, descubrimiento, publicación**

Saber cómo acceder a las ofertas de los proveedores; buscar las ofertas de los proveedores; y que los proveedores puedan publicar ofertas en la agencia

* + **Interoperabilidad**

Compatible con diferentes tecnologías

* + **Composición**

Combinación de diferentes servicios para dar un servicio de más alto nivel (vuelos, hoteles…)

* + **Desacoplable**

Independencia entre los servicios

* + **Contrato**

La estructura de datos

* + - WSDL (servicio SOAP)
    - RAML (servicio Rest)
    - Api a través de la web
* **Sincrono**

No espera a que una petición tenga respuesta, sigue ejecutando, ideal cuando se tienen muchas peticiones

## MOM (MIDDLEWARE ORIENTADO A MENSAJES)

* Evolución del C/S solucionado el cuello de botella mediante colas ya sea con o sin prioridad
* Permite que la comunicación entre el emisor y el receptor se produzca de una manera completamente desacoplada
* **Asíncrono**

Hasta que no se resuleva una petición no pasa a la siguiente

* **Intermediario**

Es el encargado de almacenar los mensajes del emisor

* + Gestión de prioridades de mensajes
  + Temporizadores para la gestión de mensajes
  + Gestión de formatos de mensajes (mensajes u objetos)
  + Gestión de seguridad
  + Gestión de persistencia de los mensajes
* **No es fácilmente interoperable**

Un único formato de mensajes

* **Centralizado**
* **Tipos:**
  + **Punto a punto (1:1)**

Cada mensaje enviado por el emisor únicamente será procesado por un proceso consumidor, emplea colas **(modelo de cola)**

* + **Publicación/Subscripción (1:M)**

Un mensaje publicado por un emisor será procesado por todos los agentes consumidores que se hayan subscrito a dicho proceso intermediario, emplea topic **(modelo topic)**

## DIFERENCIAS ENTRE SOA Y MOM

* MOM es usado en comunicación asíncrona, SOA síncrona.
* MOM consigue desacoplar el escenario completo gracias al proceso intermediario, en SOA únicamente se desacopla la localización de los servicios.
* MOM puede ser usado para la implementación de la arquitectura SOA.

## DIFERENCIAS Y SIMILITUDES ENTRE MOM Y C/S

* Ambos nos permiten la abstracción del acceso a recursos de red
* MOM posee colas y temporizadores para administrar las peticiones

## Clúster

* Los componentes se encuentran fuertemente acoplados, consiguiendo homogeneidad
* Conjunto de servidores, mejora la conexión C/S al tener muchos servidores así se reduce el cuello de botella

# Arquitectura conceptual

**ACTORES**

* **Clientes**

Los clientes podrán registrarse, realizar ofertas y lo que conlleva consigo ver los productos cancelarlas…, elegir ofertas y realizar pagos

* **Agencia**

Es la encargada de gestionar y guardar las reservas que hagan los clientes en su base de datos, en esta base de datos guardarán los clientes que se vayan registrando, también recibirán ofertes de los proveedores

* **Proveedores**

Los proveedores se encargan de recibir las reservas que realizan los clientes que son gestionadas por la reserva y de enviar a esta las ofertas

* **Bancos**

Los bancos son los encargados de gestionar los pagos

**ARQUITECTURAS EMPLEADAS**

* **Clientes-Agencia**

Para la conexión de los clientes con el sistema de agencia estos se conectarán por media de cliente/servidor ya que clientes sería el Front-End y la agencia sería el back-end, a aparte hay un fuerte acoplamiento debido a que la tecnología es la misma y es la misma empresa, a pate es la arquitectura más utilizada en entornos web.

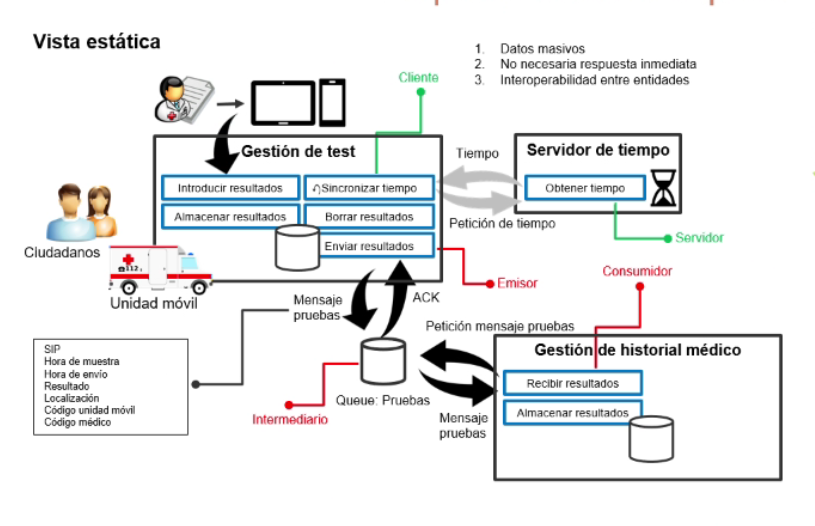
* **Agencia-Proveedores**

Para este caso he usado una arquitectura tipo SOA debido a que cumple todos los requisitos de este, necesitamos interoperabilidad, tenemos un contrato que facilita el desacoplamiento y la independencia con ellos, se va a hacer una composición de varios servicios (packs), necesitamos localizar como acceder a esos proveedores y es síncrono. También con SOA proporcionamos libertad al proveedor que dependiendo de su contrato ya sea REST o SOAP se hará un servicio punto a punto con su contrato.

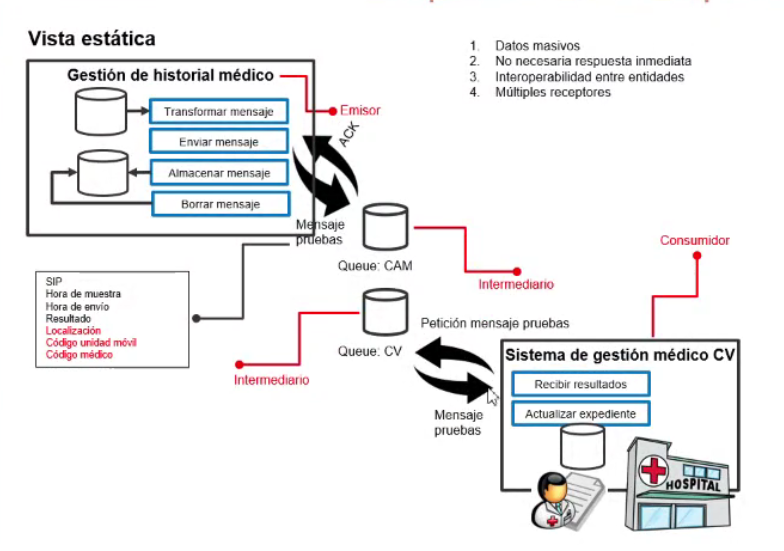
* **Agencia-Bancos**

Para la agencia con los bancos se utilizará cliente/servidor debido a que normalmente los bancos son los que nos indican como conectaros a ellos, y normalmente con un protocolo propiedad del banco que suele estar basado en sockets, por eso he decidido emplear C/S con sockets en el que el banco es el servidor y la agencia el cliente.

## MOM (ejemplo servicio de salud)

Con un servidor de tiempo para que cada centro tenga la misma hora

## MOM (con dos colitas BB)

Es el ejemplo anterior pero una cola para cada comunidad, de esta forma cada comunidad acede a su cola

# Arquitectura técnica

# Palabras refacheras