



## FORMACIÓN EN NUEVAS TECNOLOGÍAS

---

### Arquitecturas reales

# ICONO TRAINING



# FORMADOR



## Formación en Nuevas Tecnologías



[www.iconotc.com](http://www.iconotc.com)



[linkedin.com/company/icono-training-consulting](https://linkedin.com/company/icono-training-consulting)



[training@iconotc.com](mailto:training@iconotc.com)

¡Síguenos en las Redes Sociales!



Ana Isabel Vegas



Consultora / formadora en Tecnologías  
de la Información.

# Java Avanzado



## DURACIÓN

- 👉 28 horas



## MODALIDAD

- 👉 Remoto



## FECHAS y HORARIO:

- 👉 Días 17, 19, 24, 26 y 31 de Octubre, 2 y 7 de Noviembre. De 15:00 hs a 19:00 hs.

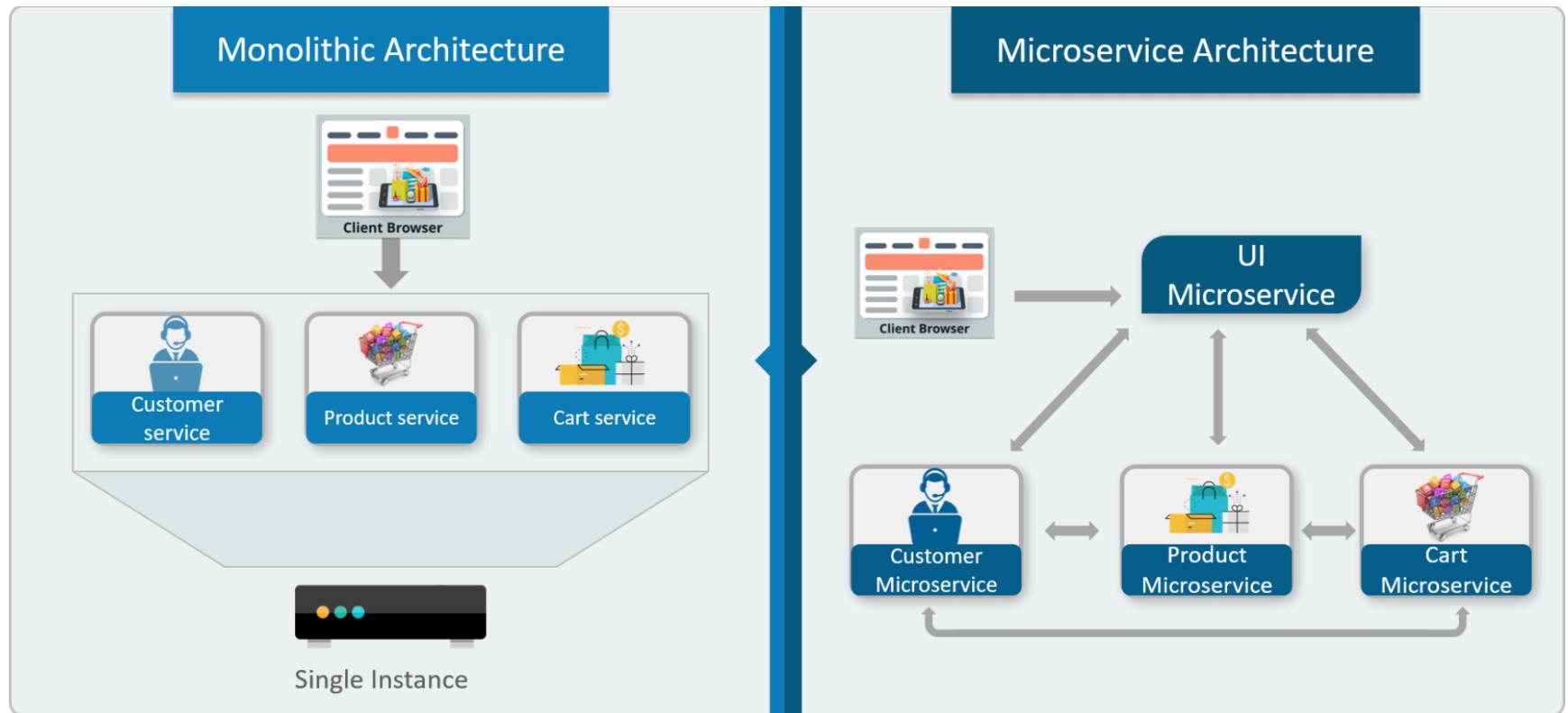


## CONTENIDO:

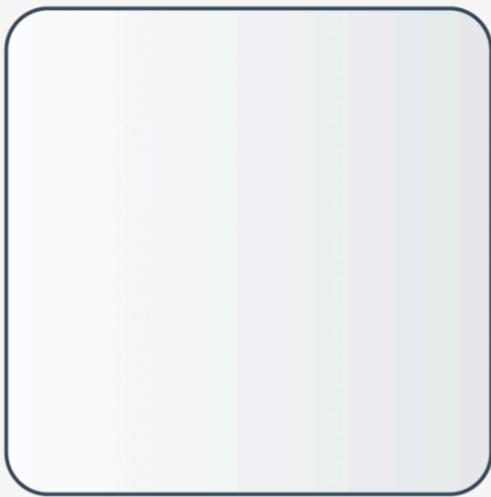
- 👉 Maven
- 👉 Git
- 👉 Arquitecturas reales
- 👉 Junit
- 👉 Patrones de diseño JEE
- 👉 Arquitectura

# Evolución de las arquitecturas

# Arquitectura monolitica vs Arquitectura microservicios

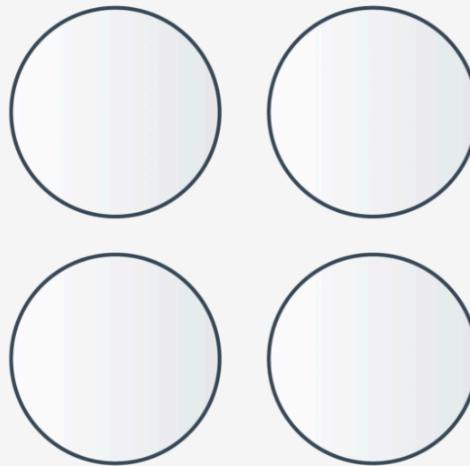


# Vs SOA



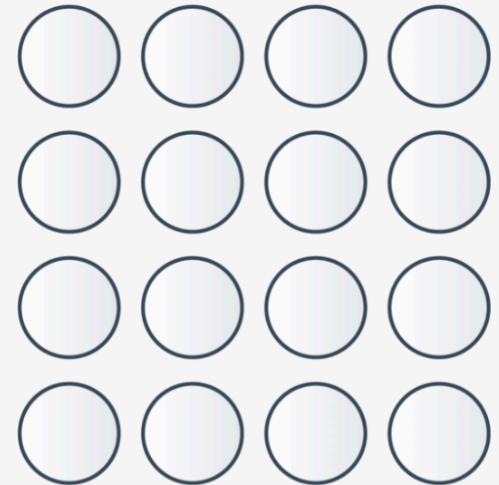
**Monolithic**

**Single Unit**



**SOA**

**Coarse-grained**



**Microservices**

**Fine-grained**

# Tendencia en el desarrollo



La tendencia es que las aplicaciones sean diseñadas con un **enfoque orientado a microservicios**, construyendo múltiples servicios que colaboran entre si, en lugar del **enfoque monolítico**, donde se construye y despliega una única aplicación que contenga todas las funcionalidades.

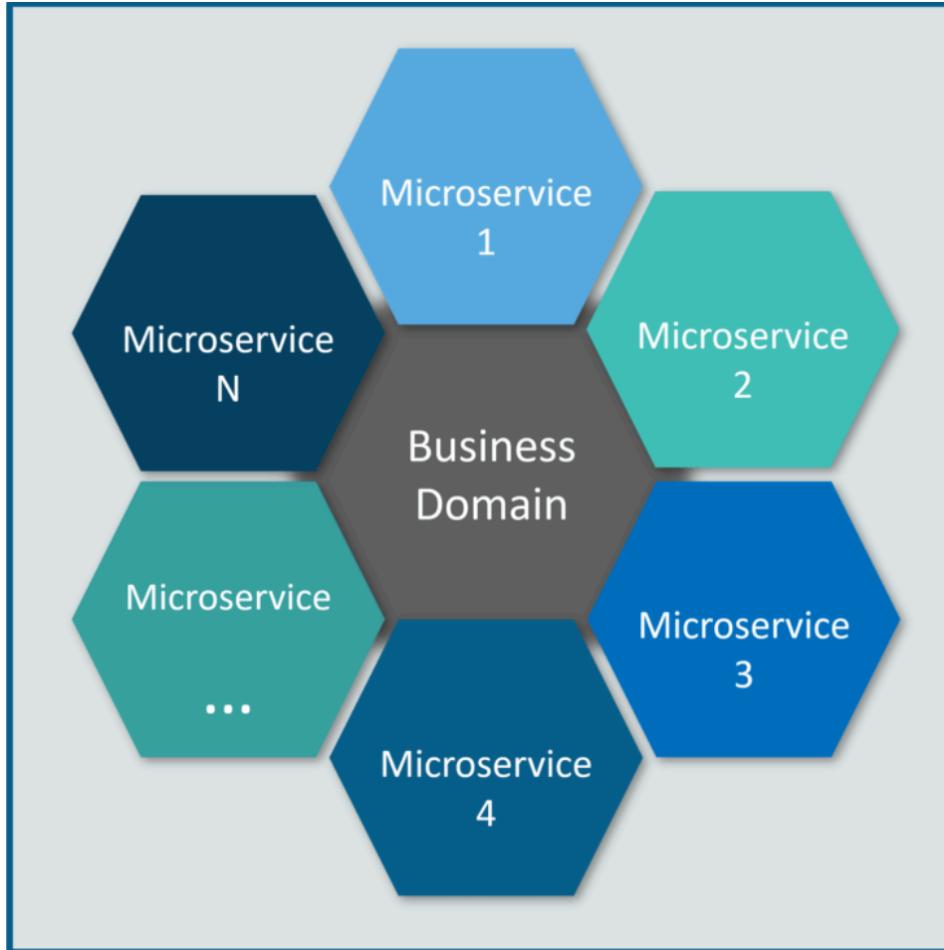
# Microservicios

# Que es un microservicio?



Según [Martin Fowler](#) y [James Lewis](#) explican en su artículo [Microservices](#), los **microservicios** se definen como un estilo arquitectural, es decir, una forma de desarrollar una aplicación, basada en un conjunto de pequeños servicios, cada uno de ellos ejecutándose de forma autónoma y comunicándose entre si mediante mecanismos livianos, generalmente a través de peticiones REST sobre HTTP por medio de sus **APIs**.

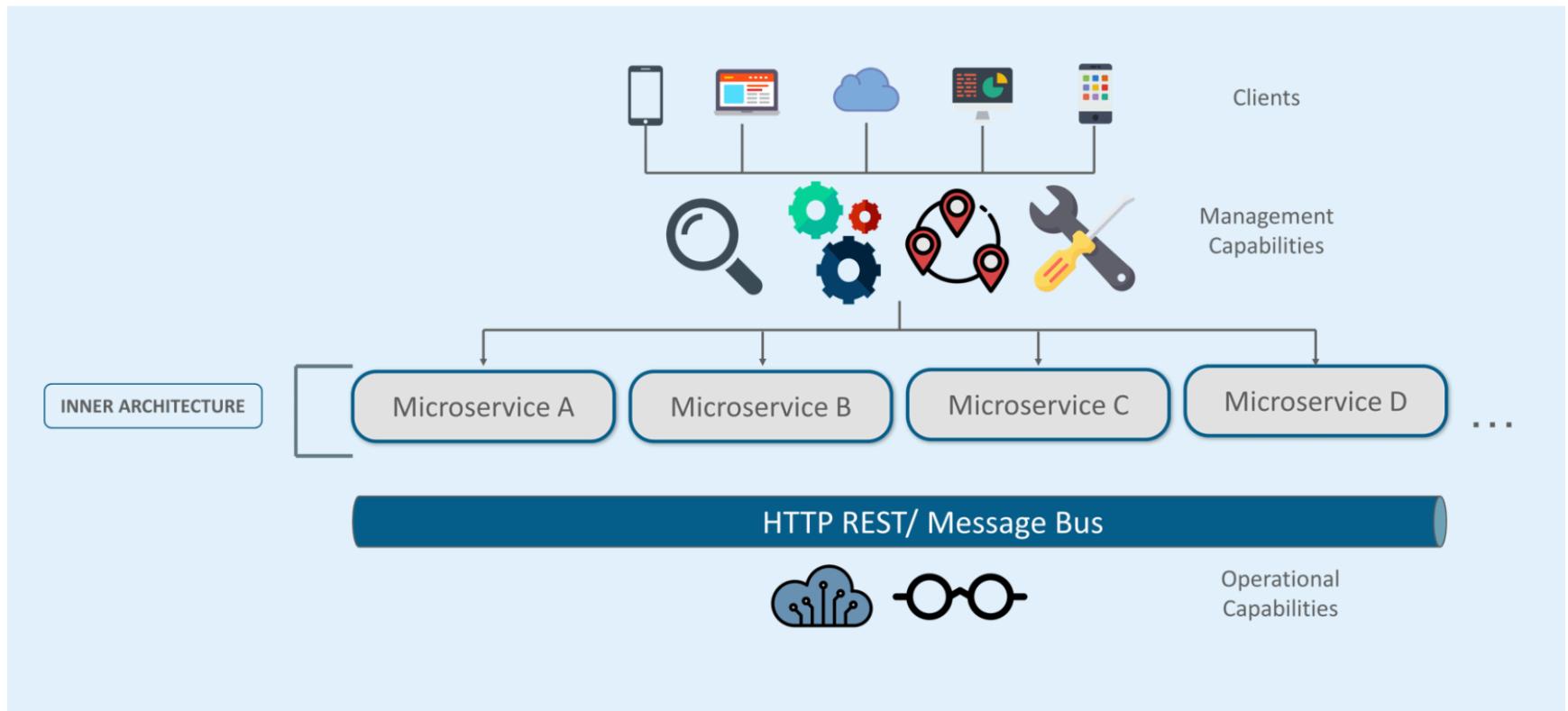
# Que es un microservicio?



# Características de los Microservicios

-  Pueden ser **auto-contenidos**, de tal forma que incluyen todo lo necesario para prestar su servicio
-  **Servicios pequeños**, lo que facilita el mantenimiento. Ej: Personas, Productos, Posición Global, etc
-  **Principio de responsabilidad** única: cada microservicio hará una única cosa, pero la hará bien
-  **Políglotas**: una arquitectura basada en microservicios facilita la integración entre diferentes tecnologías (lenguajes de programación, BBDD...etc)
-  **Despliegues unitarios**: los microservicios pueden ser desplegados por separado, lo que garantiza que cada despliegue de un microservicio no implica un despliegue de toda la plataforma. Tienen la posibilidad de incorporar un **servidor web embebido** como *Tomcat* o *Jetty*
-  **Escalado eficiente**: una arquitectura basada en microservicios permite un escalado elástico horizontal, pudiendo crear tantas instancias de un microservicio como sea necesario.

# Arquitectura microservicios



# Arquitectura microservicios

-  Diferentes clientes de diferentes dispositivos intentan usar diferentes servicios como búsqueda, creación, configuración y otras capacidades de administración
-  Todos los servicios se separan según sus dominios y funcionalidades y se asignan a microservicios individuales.
-  Estos microservicios tienen su propio balanceador de carga y entorno de ejecución para ejecutar sus funcionalidades y al mismo tiempo captura datos en sus propias bases de datos.
-  Todos los microservicios se comunican entre sí a través de un servidor sin estado que es REST o Message Bus.
-  Los microservicios conocen su ruta de comunicación con la ayuda de Service Discovery y realizan capacidades operativas tales como automatización, monitoreo
-  Luego, todas las funcionalidades realizadas por los microservicios se comunican a los clientes a través de la puerta de enlace API.
-  Todos los puntos internos están conectados desde la puerta de enlace API. Por lo tanto, cualquiera que se conecte a la puerta de enlace API se conecta automáticamente al sistema completo

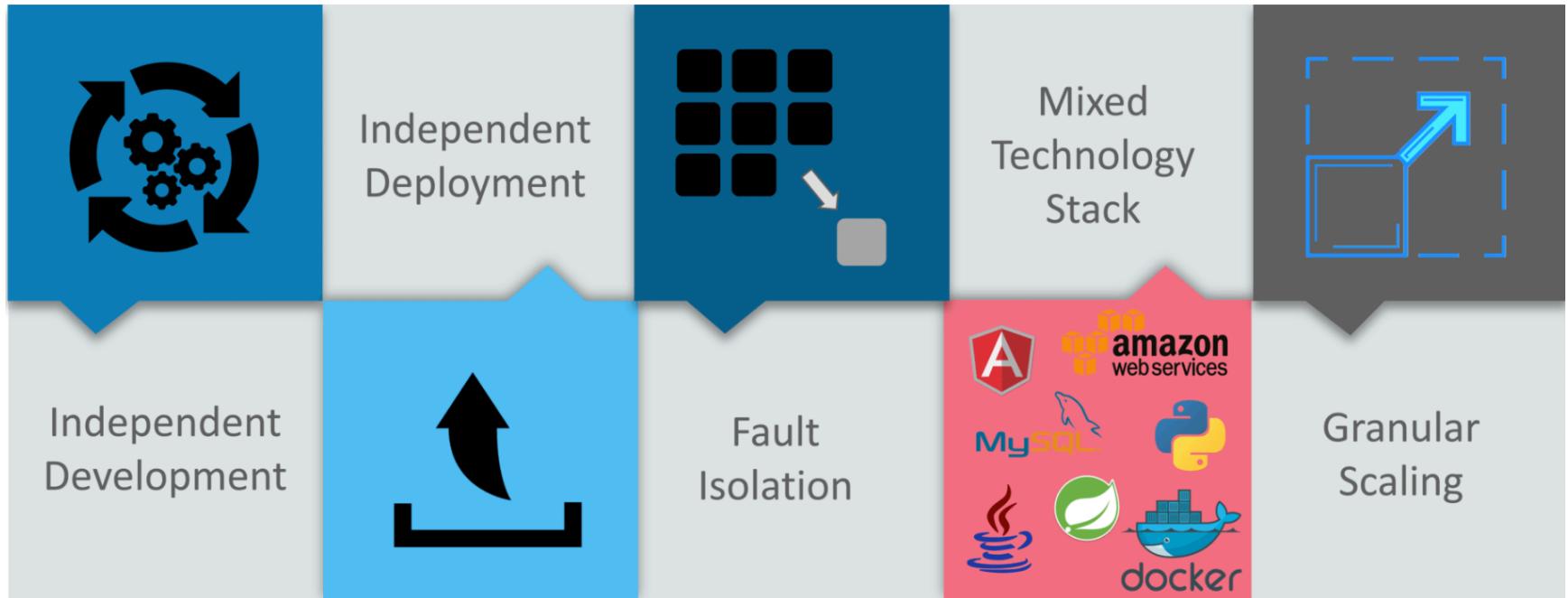
# Características de los microservicios



# Características de los microservicios

-  **Desacoplamiento:** los servicios dentro de un sistema se desacoplan en gran medida. Por lo tanto, la aplicación en su conjunto se puede construir, modificar y escalar fácilmente.
-  **Componentes:** los microservicios se tratan como componentes independientes que se pueden reemplazar y actualizar fácilmente.
-  **Capacidades empresariales:** los microservicios son muy simples y se centran en una sola capacidad
-  **Autonomía:** los desarrolladores y los equipos pueden trabajar de forma independiente, lo que aumenta la velocidad.
-  **Entrega continua:** permite lanzamientos frecuentes de software, a través de la automatización sistemática de la creación, prueba y aprobación del software.
-  **Responsabilidad:** Los microservicios no se centran en aplicaciones como proyectos. En cambio, tratan las aplicaciones como productos de los que son responsables.
-  **Gobernanza descentralizada:** el enfoque está en usar la herramienta adecuada para el trabajo correcto. Eso significa que no hay un patrón estandarizado o ningún patrón tecnológico. Los desarrolladores tienen la libertad de elegir las mejores herramientas útiles para resolver sus problemas
-  **Agilidad -** Los microservicios apoyan el desarrollo ágil. Cualquier nueva característica puede ser desarrollada rápidamente y descartada nuevamente

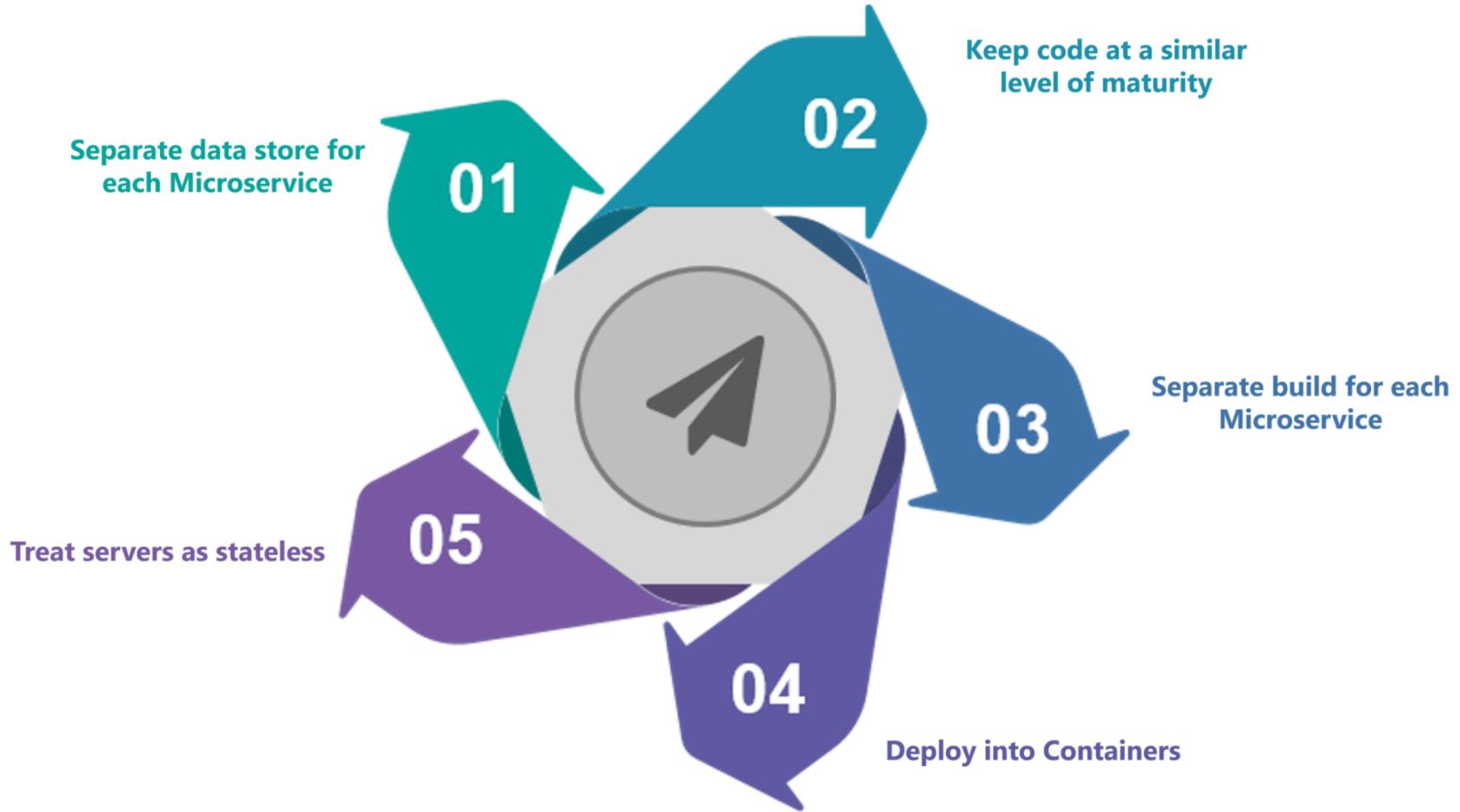
# Ventajas de los microservicios



# Ventajas de los microservicios

-  **Desarrollo independiente:** todos los microservicios se pueden desarrollar fácilmente según su funcionalidad individual
-  **Implementación independiente:** en función de sus servicios, se pueden implementar individualmente en cualquier aplicación
-  **Aislamiento de fallos:** incluso si un servicio de la aplicación no funciona, el sistema continúa funcionando
-  **Pila de tecnología mixta:** se pueden utilizar diferentes lenguajes y tecnologías para crear diferentes servicios de la misma aplicación
-  **Escalado granular:** los componentes individuales pueden escalarse según la necesidad, no es necesario escalar todos los componentes juntos

# Buenas prácticas diseño



# Quien los utiliza?

amazon.com®



NETFLIX



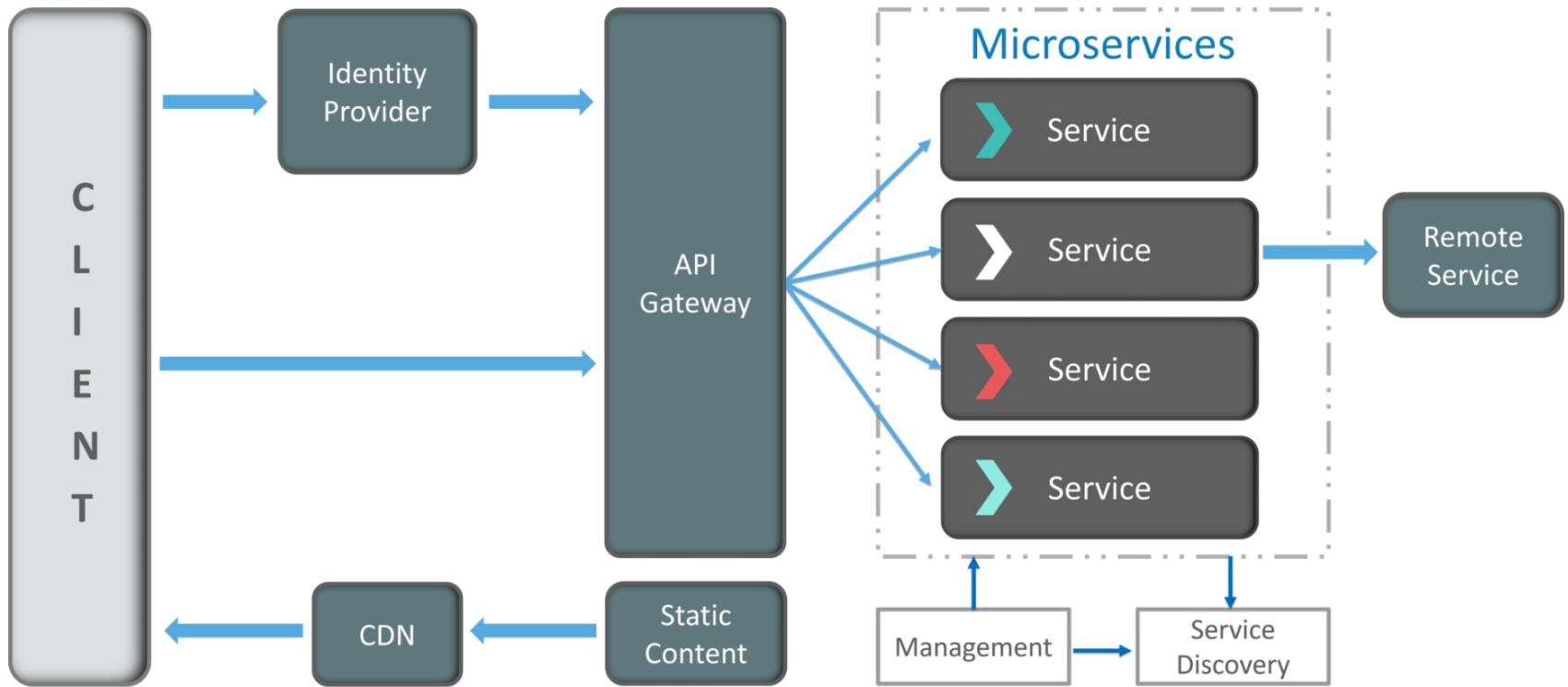
GILT

eBay

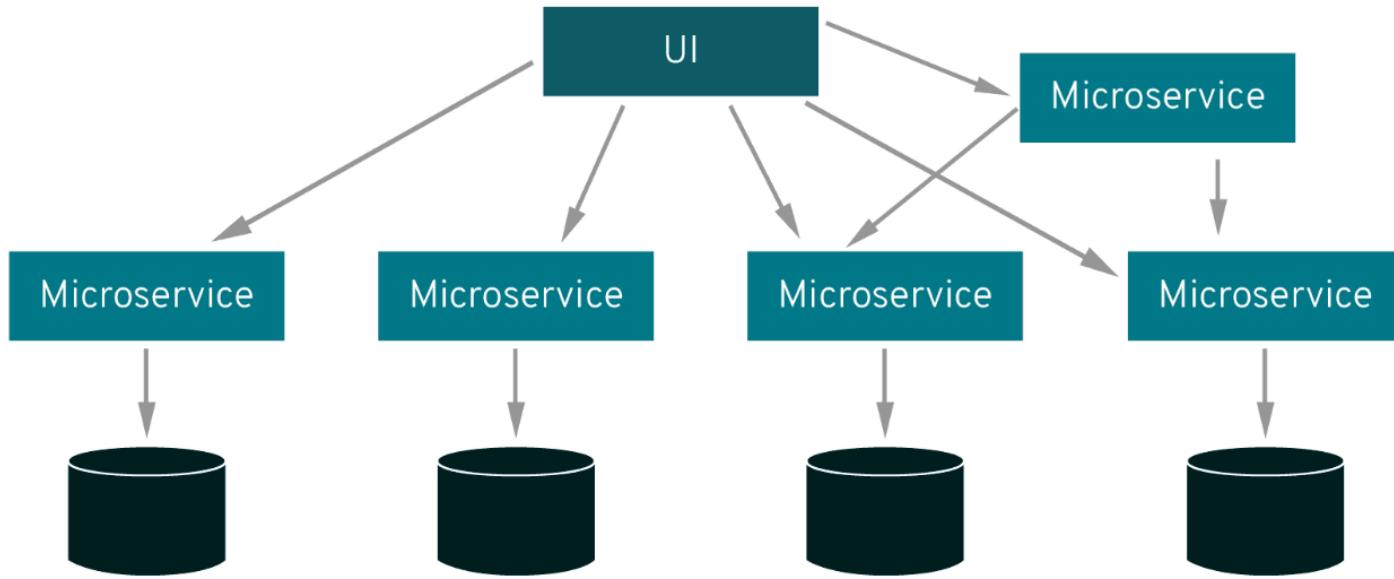


the guardian

# Componentes de arquitectura



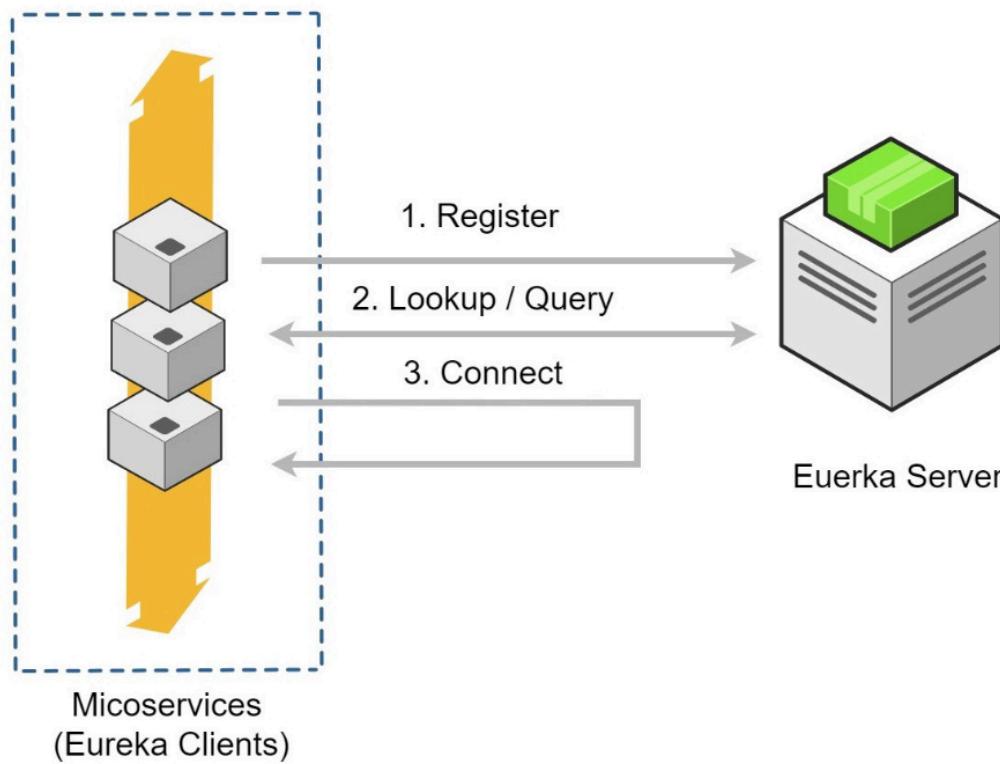
# Consulta entre servicios



# Consulta entre servicios

-  Los microservicios se crean de forma independiente y se comunican entre sí. Además, si se produce un error individual, este no provoca una interrupción de toda la aplicación.
-  La comunicación se lleva a cabo a través de peticiones Rest.
-  Dos formas de implementar el cliente de la petición:
  -  RestTemplate
  -  Feign

# Eureka Netflix



# Eureka Netflix



Eureka es un servicio rest que permite al resto de microservicios registrarse en su directorio.

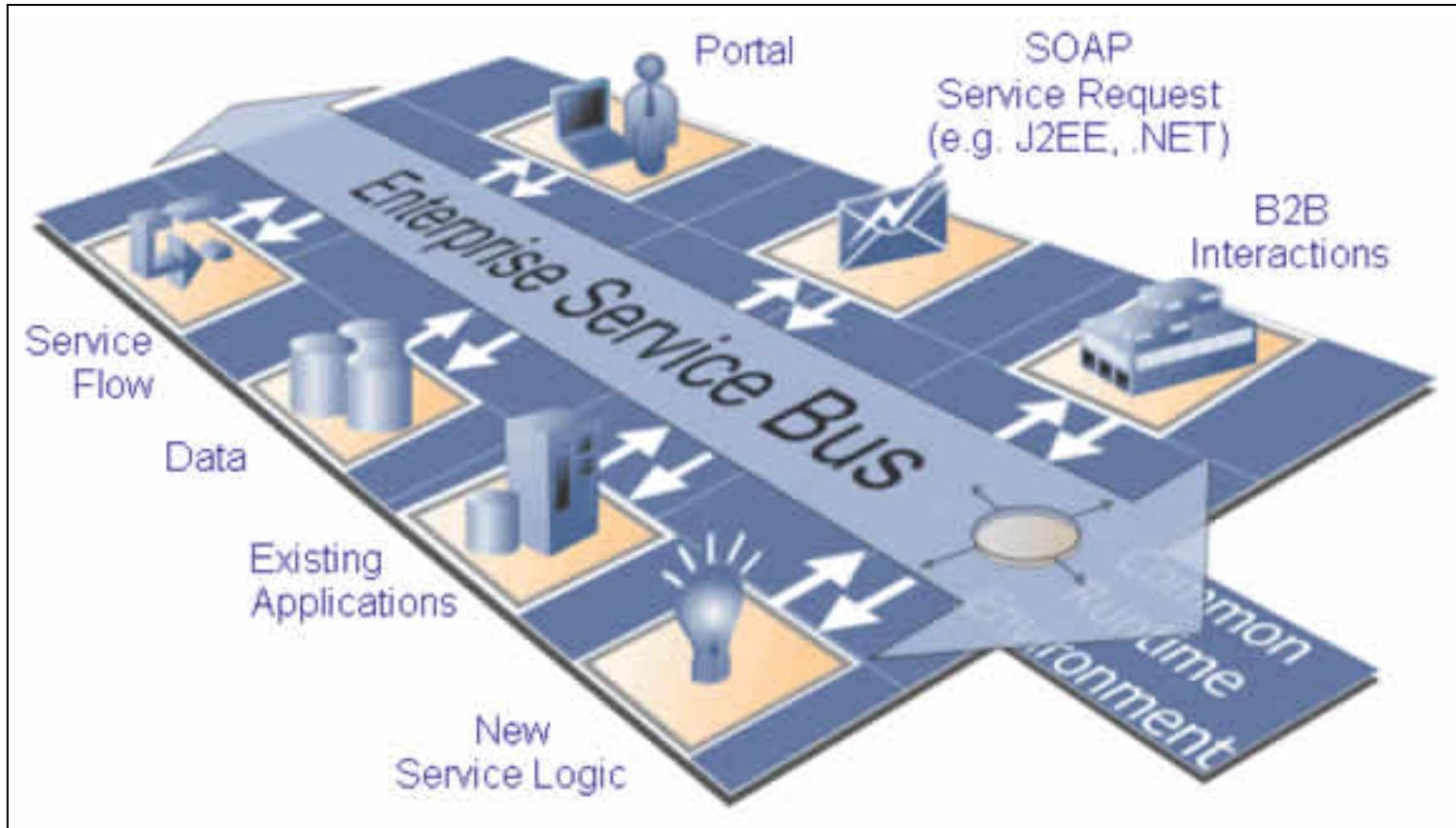


Esto es muy importante, puesto que no es Eureka quien registra los microservicios, sino los microservicios los que solicitan registrarse en el Eureka.

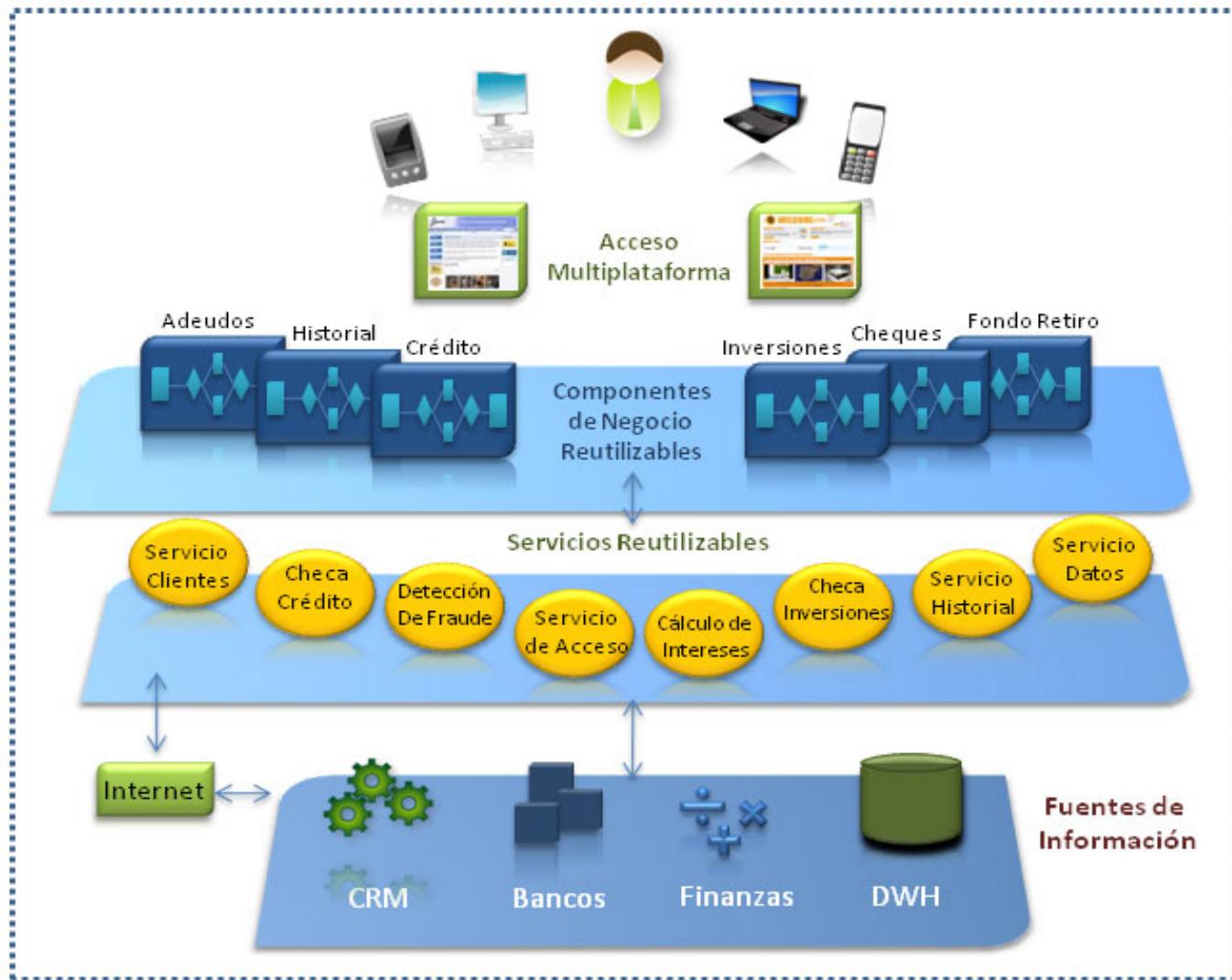
# Eureka Netflix

-  Cuando un microservicio registrado en Eureka arranca, envía un mensaje a Eureka indicándole que está disponible.
-  El servidor Eureka almacenará la información de todos los microservicios registrados así como su estado.
-  La comunicación entre cada microservicio y el servidor Eureka se realiza mediante heartbeats cada X segundos.
-  Si Eureka no recibe un heartbeat de un determinado microservicio, pasados 3 intervalos, el microservicio será eliminado del registro.
-  Además de llevar el registro de los microservicios activos, Eureka también ofrece al resto de microservicios la posibilidad de "descubrir" y acceder al resto de microservicios registrados.
-  Por ello Eureka es considerado un servicio de registro y descubrimiento de microservicios

# ESB

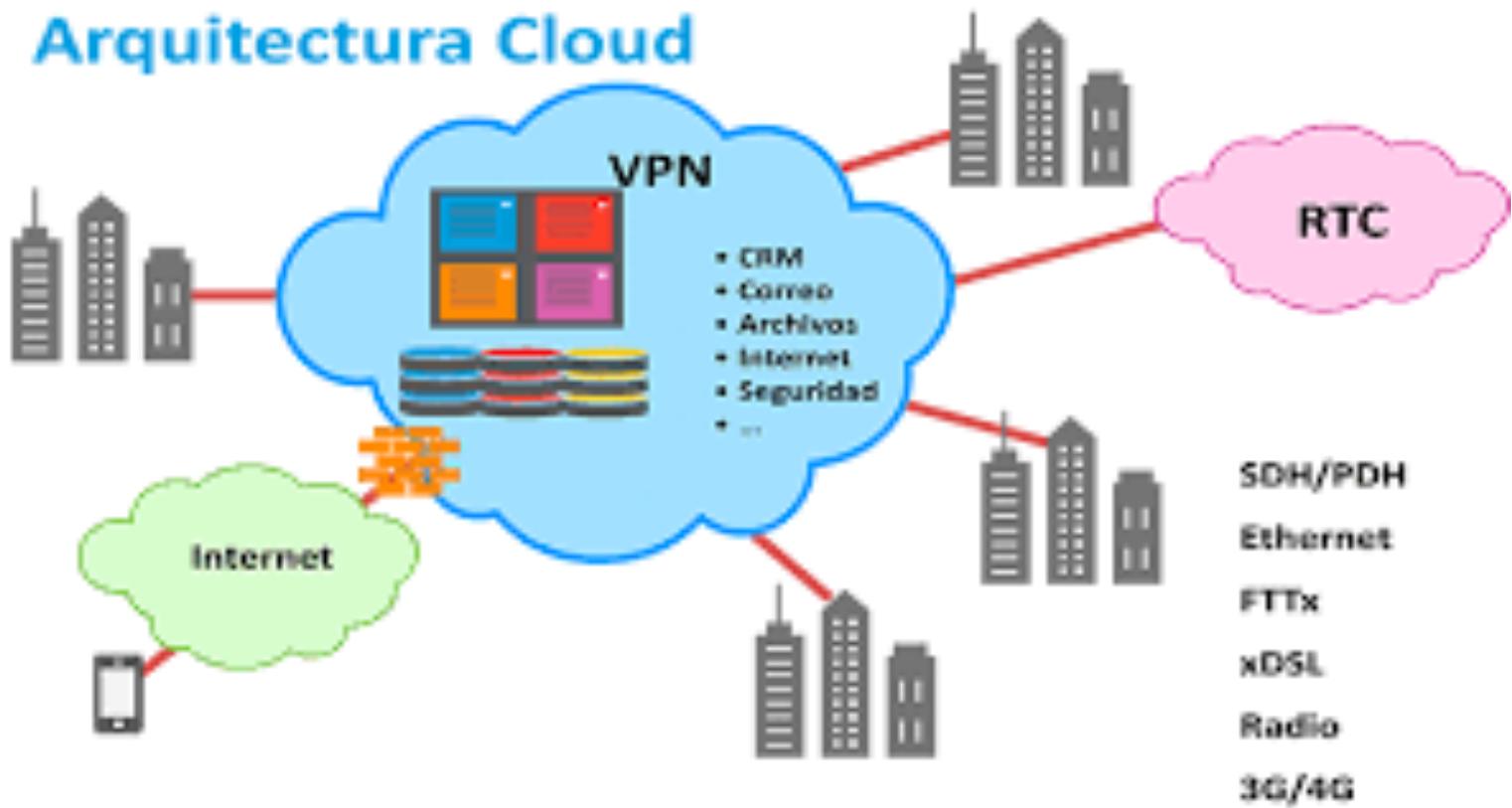


# SOA



# Arquitectura Cloud

## Arquitectura Cloud



Gracias por  
vuestra  
participación



¡Seguimos en contacto!

---

[www.iconotc.com](http://www.iconotc.com)

