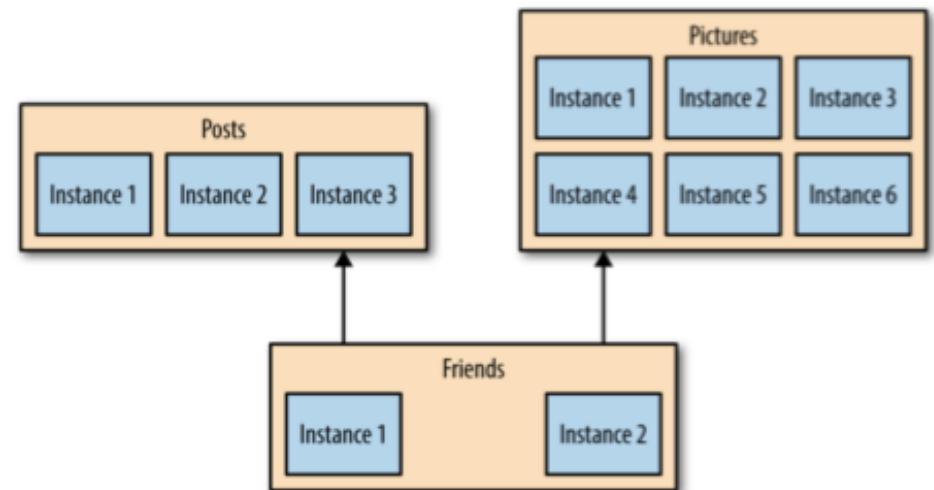
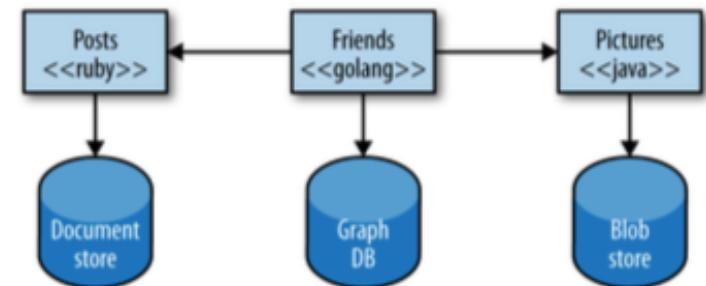


Arquitectura de Servicios: componentes débilmente acopladas

- una componente es una unidad de software que puede ser reemplazada o mejorada (upgrade) en forma independiente
- tradicionalmente las componentes se presentan en forma de módulos o librerías compartidas
- un servicio es una componente que no corre en el mismo proceso y se comunica con el resto mediante un protocolo como http o rpc

Ventajas

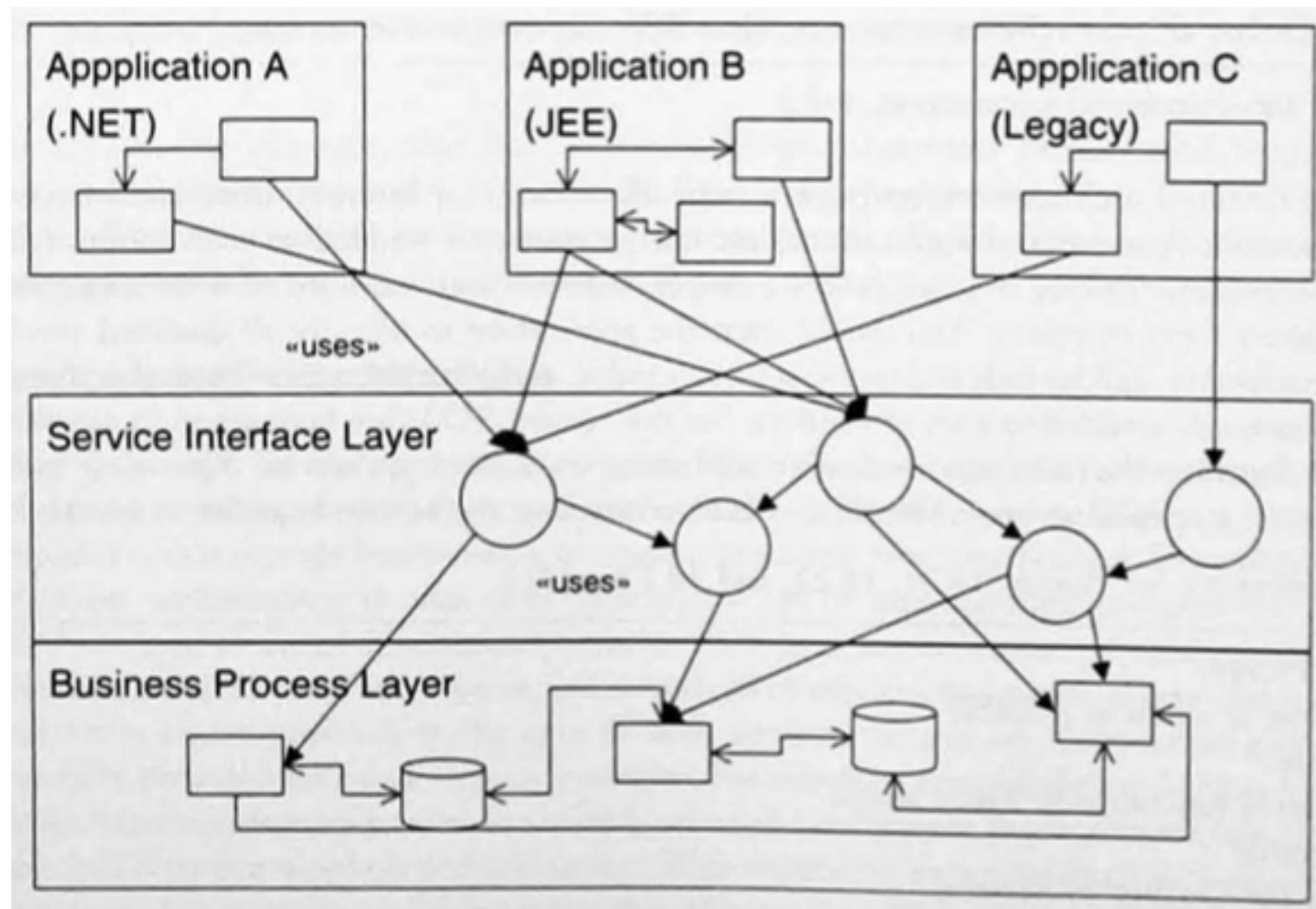
- deployable en forma independiente
 - si se hace un cambio de una componente no es necesario redeployar la aplicación
- interfaz explícita
 - hace más difícil violar acuerdos
- heterogeneidad tecnológica
- resiliencia
- escalamiento flexible



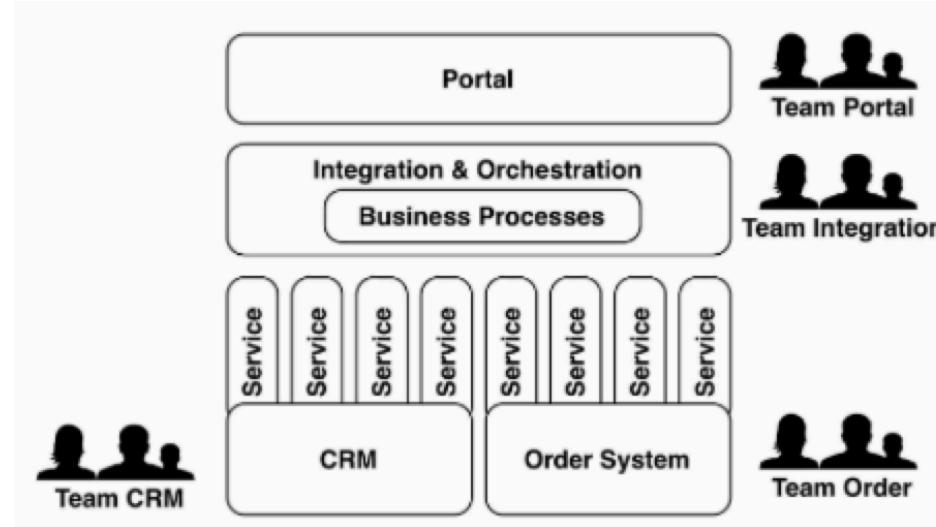
Service Oriented Architecture (SOA)

- primer intento, mediados de los 2000
- Intimamente ligado a la popularidad de los Web Services y a la formalización de los procesos de negocio
- uso de SOAP
- granularidad gruesa (servicios son proceso de negocio)
- difícil de implementar
- no cumple a cabalidad promesa de desarrollo y deploy independiente

SOA

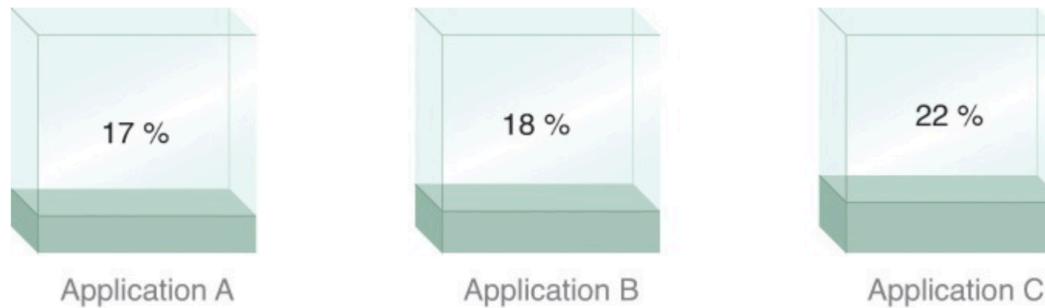


Ejemplo



- CRM - aplicación para manejo de clientes
 - servicios pueden ser creación de clientes nuevos o información de historia de un cliente
- Order System - aplicación que maneja órdenes
- Integration Platform - permite que los servicios interactúen
- Portal - interfaz para los usuarios

Compartiendo Código



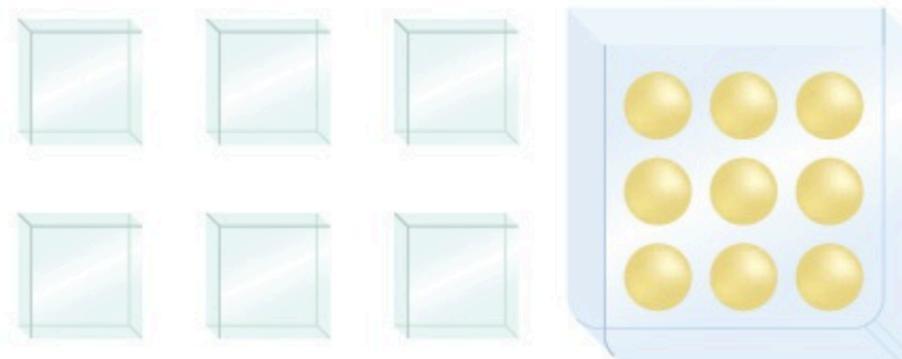
- En arquitectura clásica un proceso de negocios es implementado como una aplicación
- Necesariamente en cada aplicación hay código redundante (20%)
- Con SOA se puede compartir el código que es común



quantity of overall
automation logic = x

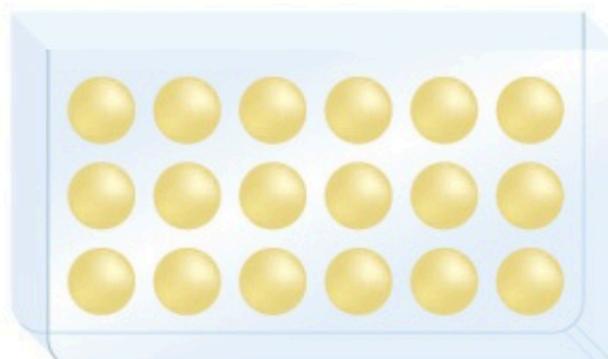


enterprise with an inventory of standalone applications



quantity of overall
automation logic = 85% of x

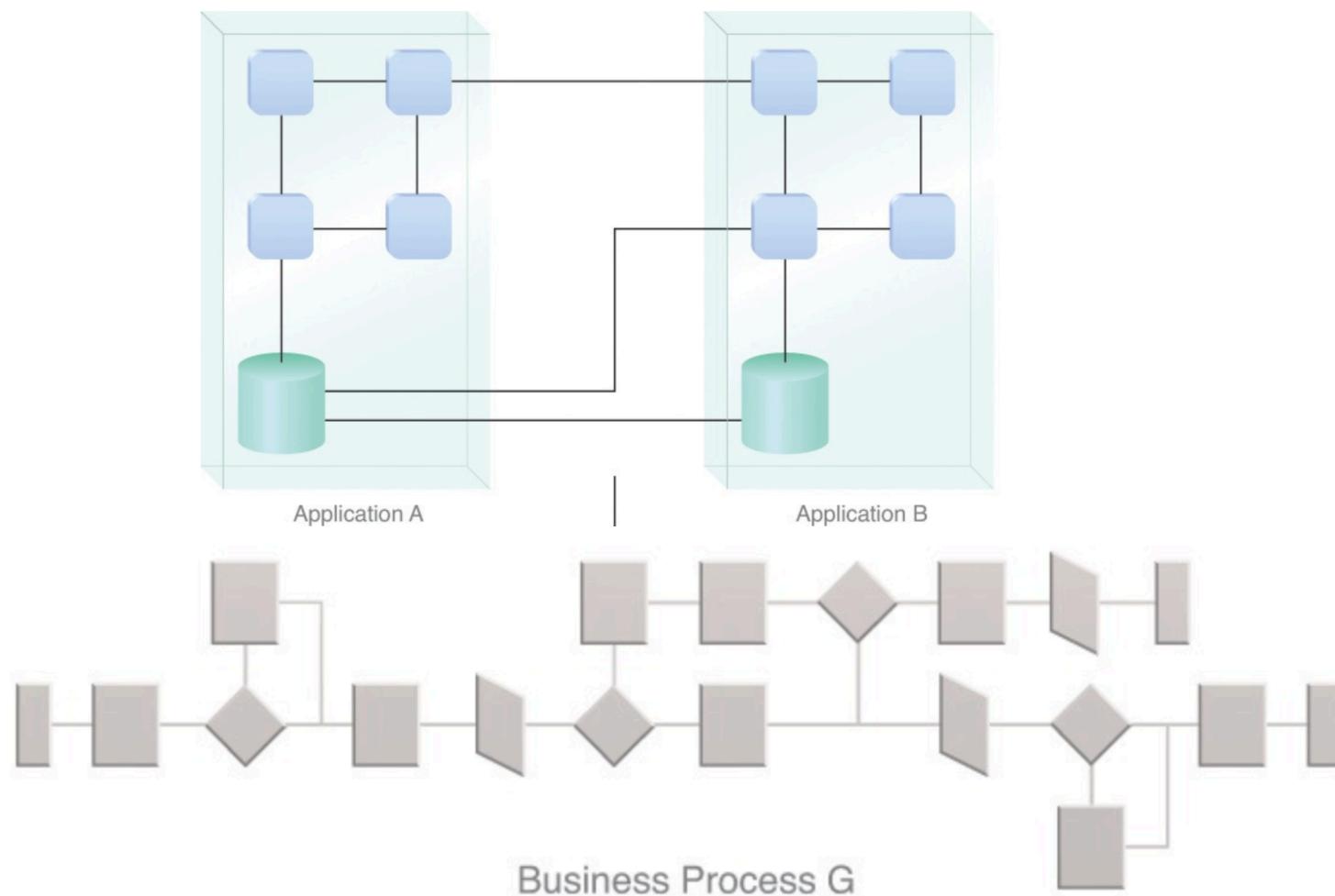
enterprise with a mixed inventory of standalone
applications and services



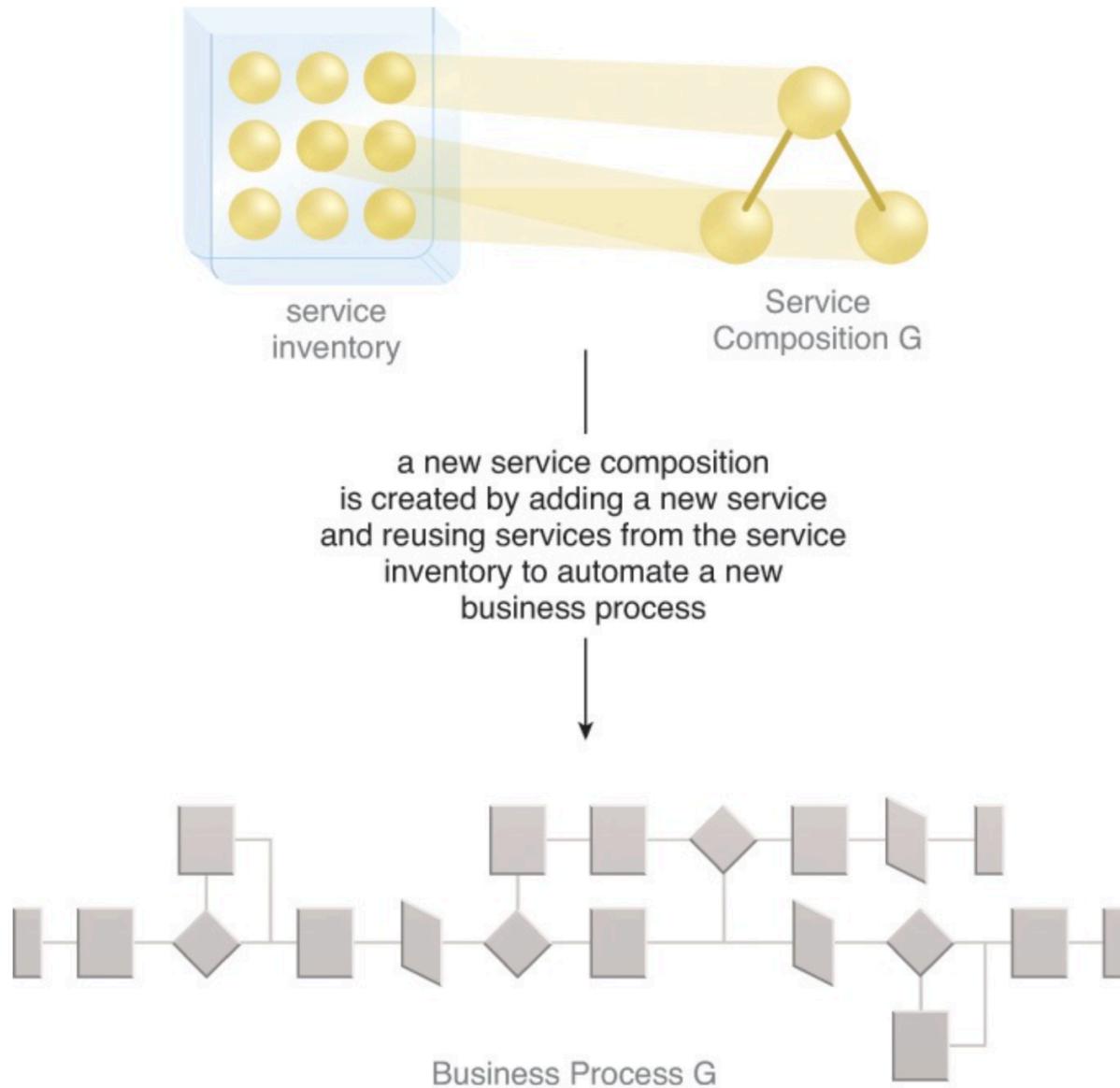
quantity of overall
automation logic = 65% of x

enterprise with an inventory of services

A veces un proceso de negocio requiere integrar aplicaciones

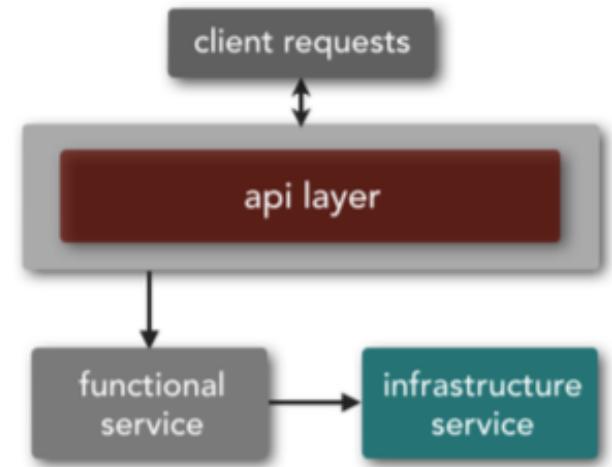


Con Arquitectura de Servicios



La llegada de los microservicios

- 2011 - 2012
- algunos se refieren a "SOA hecho bien"
- primera generación de SOA comenzó a hacerse demasiado compleja
 - SOAP -> middleware
- representa una maduración de las ideas de SOA a la luz de la experiencia práctica
- no requiere capa de middleware
 - cada servicio expone una API (contrato)
- la mayor parte son servicios funcionales con algunos servicios de infraestructura (no se exponen al mundo externo)

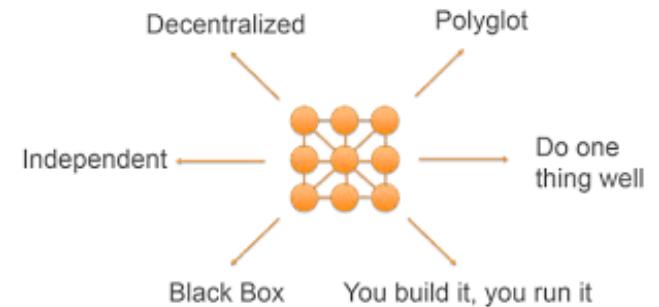


Características de Arquitectura de Microservicios

- Componentes son Servicios
- Altamente cohesivas y desacopladas (smart endpoints, dumb pipes)
- Usan principios y protocolos de la Web (Http, Rest)
- Organización en base a capacidades de negocio y no de especializaciones tecnológicas (UI, DBA, middleware)
- Gobierno descentralizado (un servicio es totalmente independiente)
- Manejo de datos descentralizado

Microservicios vs SOA

- granularidad mucho menor
- No hay necesidad de capa de integración/orquestación
- Cada microservicio es responsable de comunicarse con quien lo requiera (mensaje simples sin inteligencia)
- UI es parte del microservicio
- Alcance no tiene que ser la organización completa



2000's

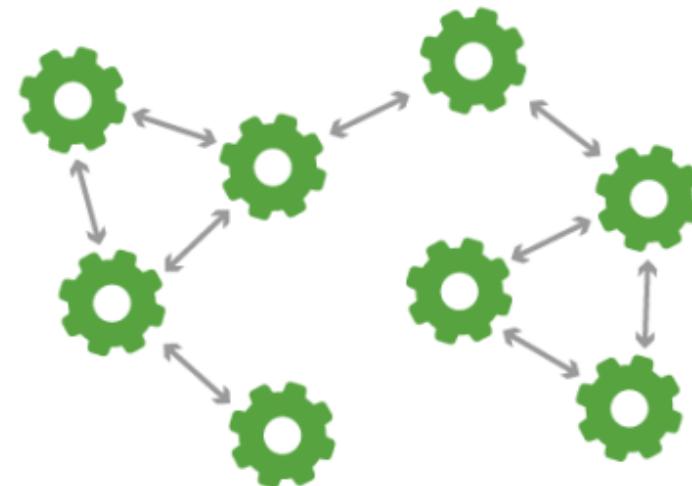
SERVICE ORIENTED ARCHITECTURE



SOA based applications are comprised of more loosely coupled components that use an Enterprise Services Bus messaging protocol to communicate between themselves.

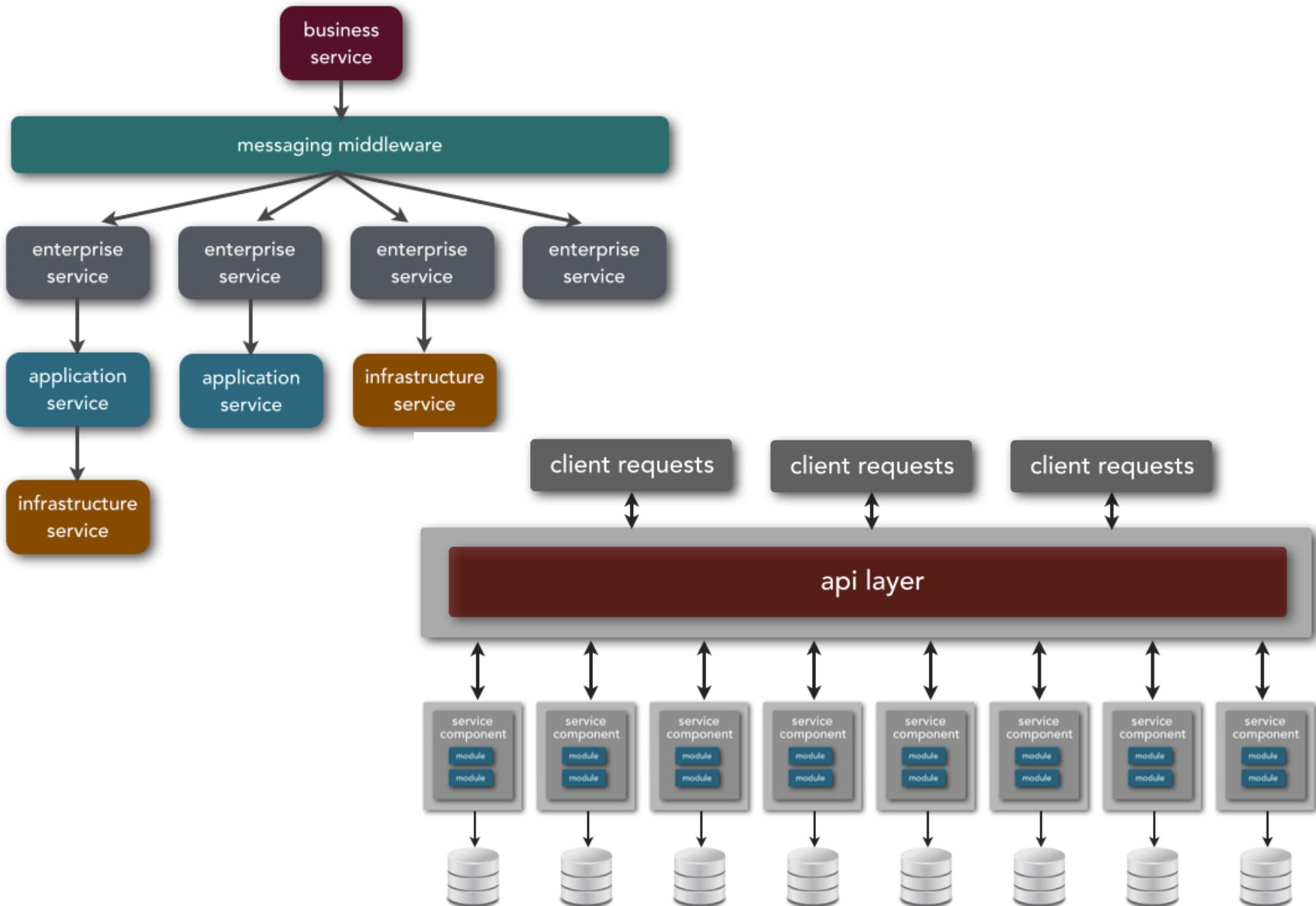
2010's

MICROSERVICES ARCHITECTURE

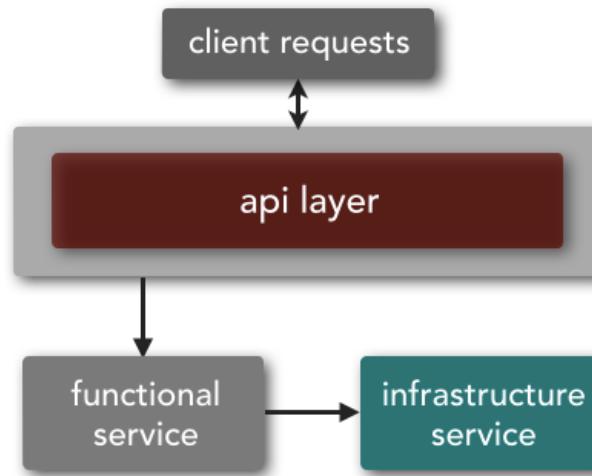


Microservices are a number of independent application services delivering one single functionality in a loosely connected and self-contained fashion, communicating through light-weight messaging protocols such as HTTP, REST or Thrift API.

SOA vs Microservices

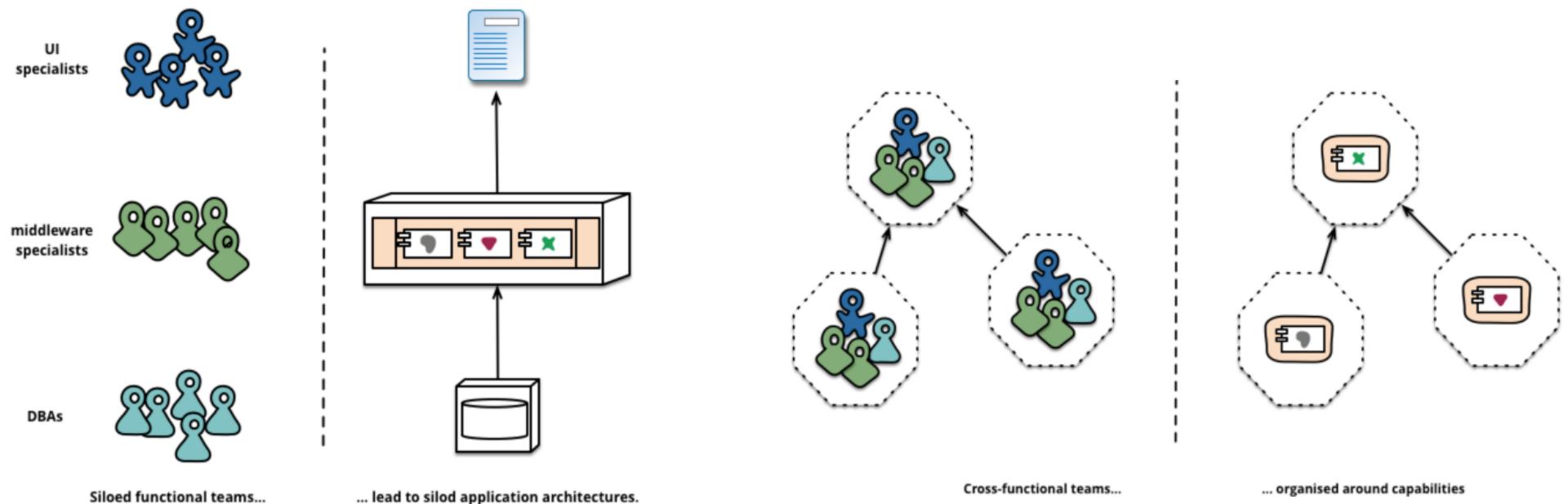


Tipo de Microservicios



- Servicios funcionales - Se exponen hacia afuera
- Servicios de infraestructura - Para uso interno (de los otros servicios)
 - autenticación y autorización
 - monitoreo
 - logging y auditoría

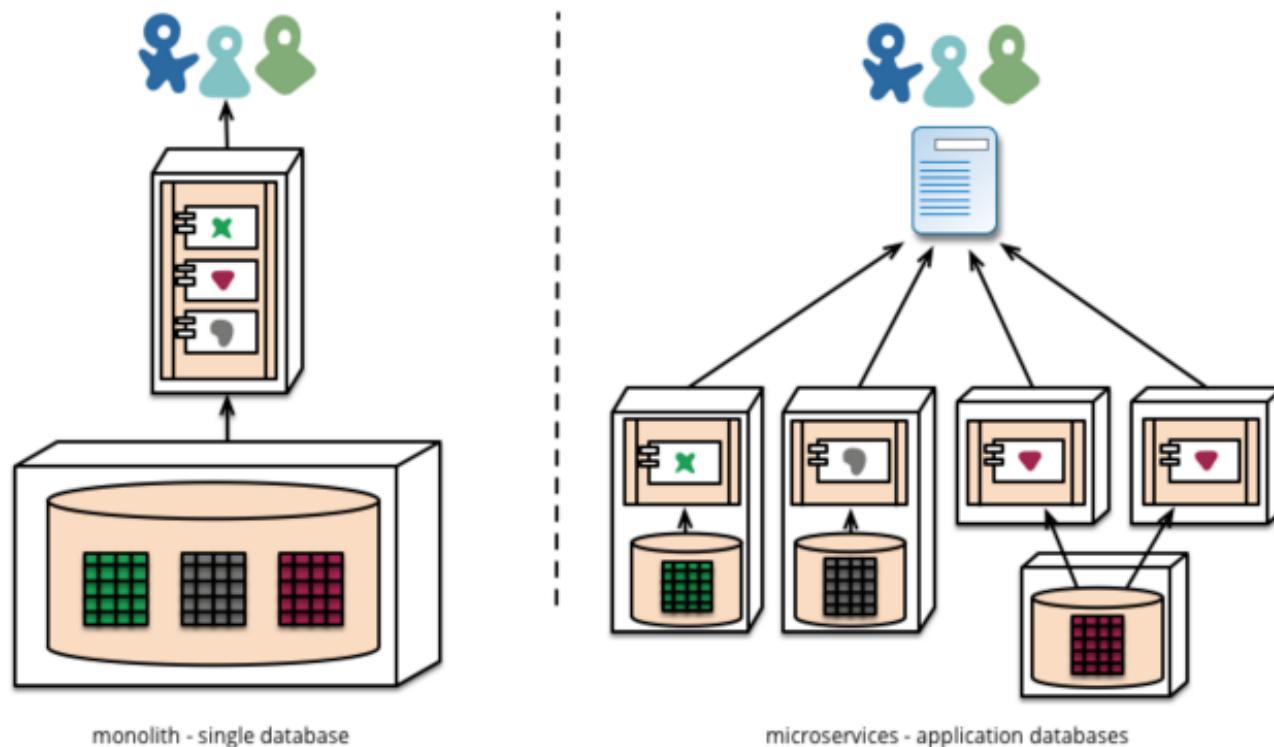
Organización de Equipo de Trabajo



Clásico: front-end, back-end, DB

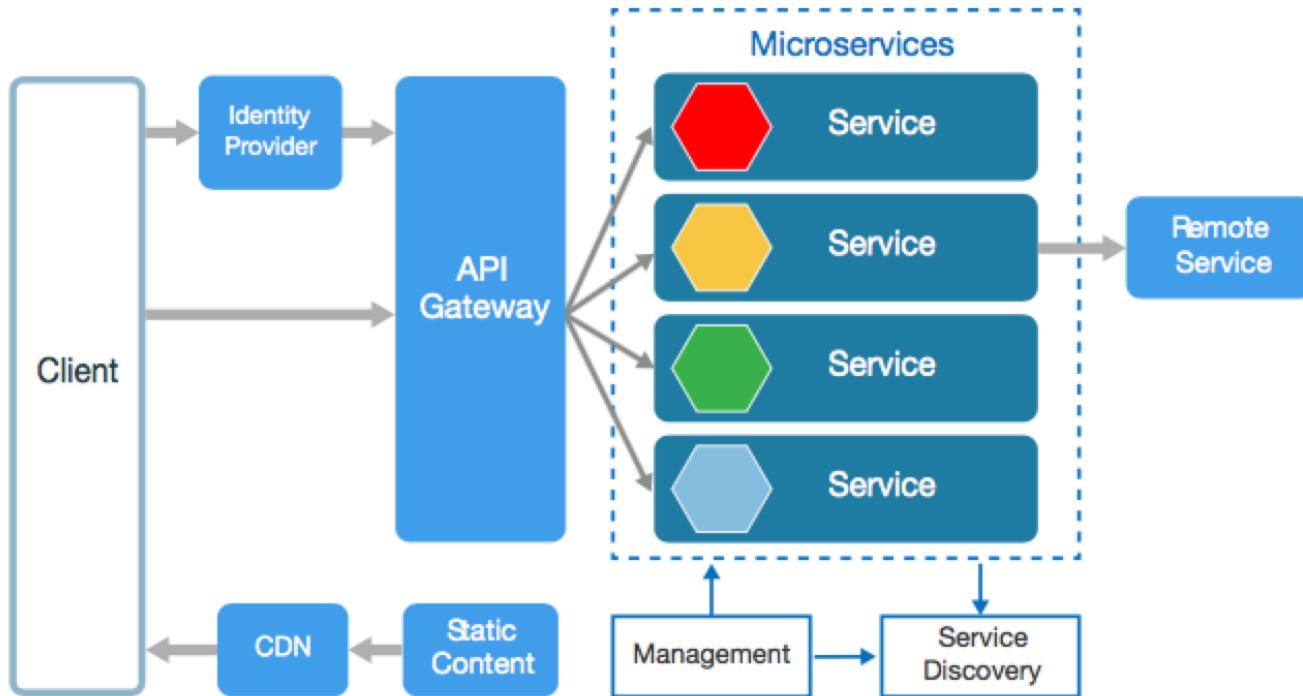
Microservices: equipos multifuncionales

Manejo de Datos Descentralizado



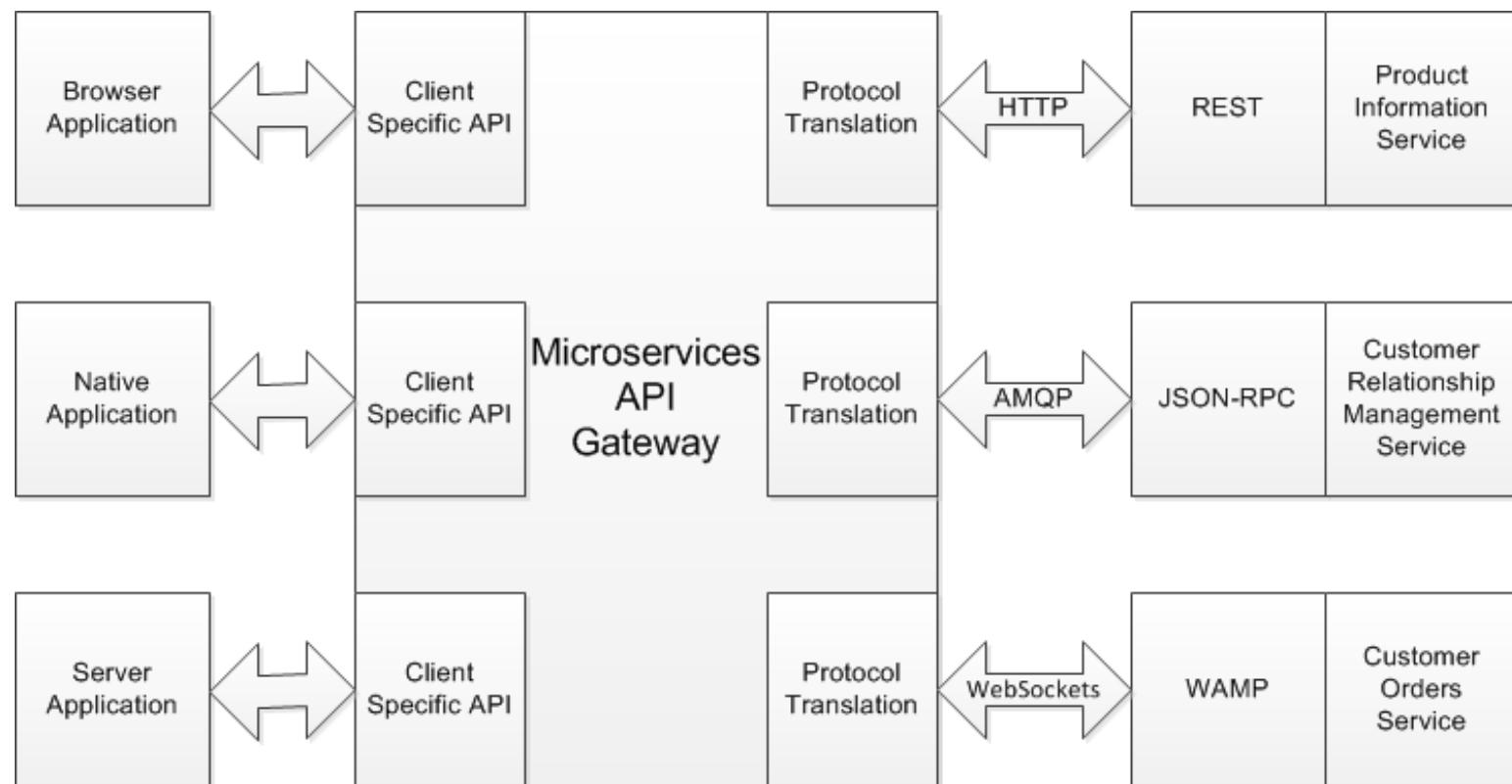
- cada servicio maneja su propia BD
- pueden ser instancias de una BD o tecnologías completamente diferentes
- no se usan transacciones para coordinar, consistencia eventual
- respuesta rápida y escalabilidad se paga con posibles grados de inconsistencia temporal

El API Layer



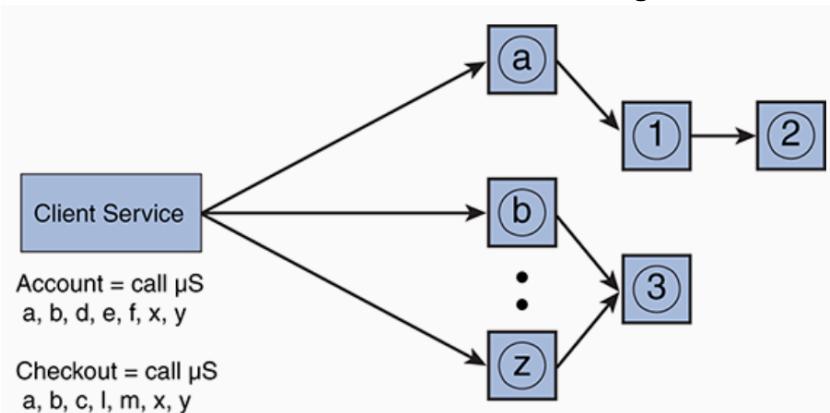
- lo que se expone a clientes
- puede agregar respuestas de varios servicios
- permite cambiar interfaz de servicios
- servicios pueden usar protocolos que no son http
- puede proporcionar servicios generales (logging, load balancing, etc)

API Gateway

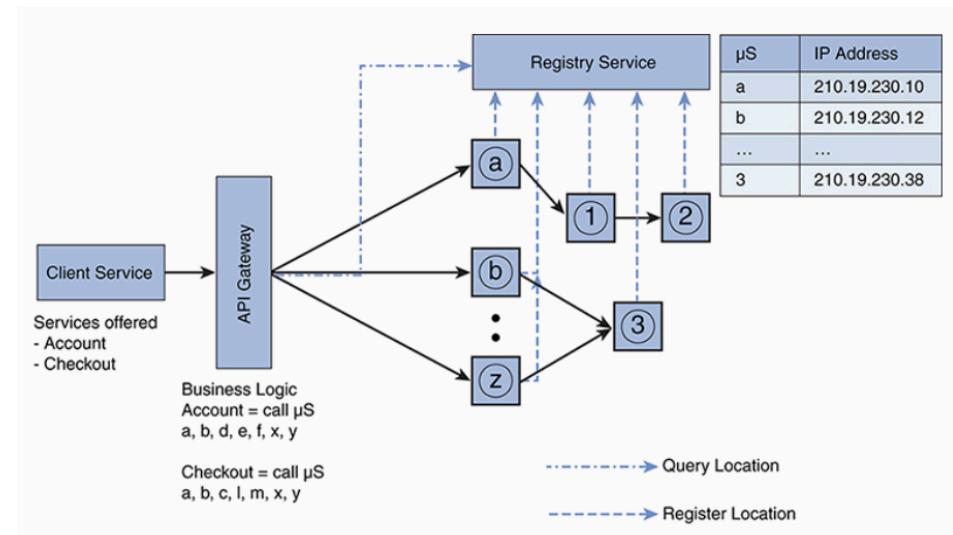


Variaciones

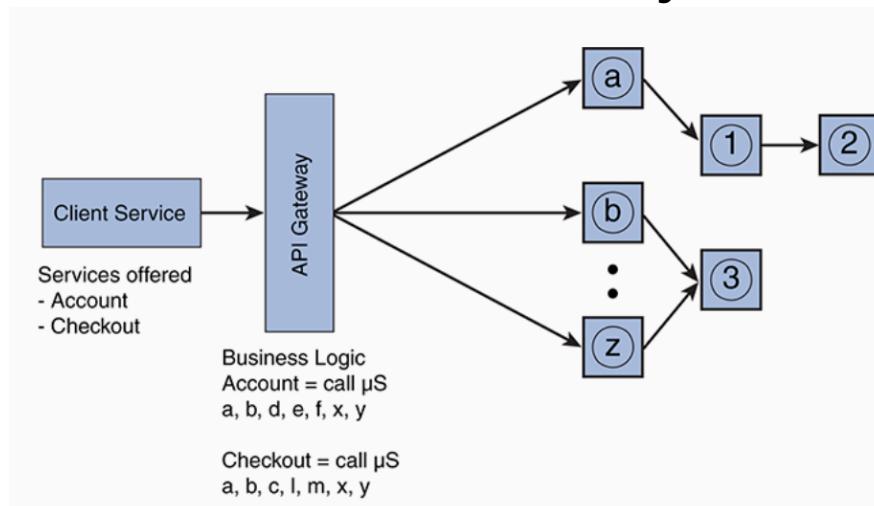
Sin API Gateway



Con API Gateway y Registry



Con API Gateway



No es la panacea

panacea

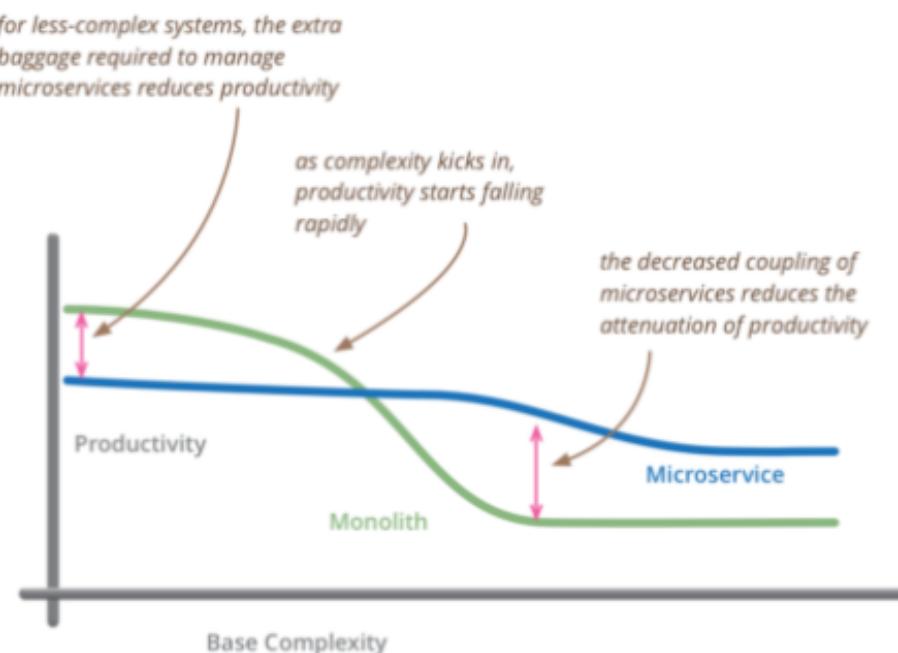
/,panə'si:ə/ (4)

noun

a solution or remedy for all difficulties or diseases.

"the panacea for all corporate ills"

synonyms: universal cure, **cure-all**, cure for all ills, universal remedy, sovereign remedy, **heal-all**, **nostrum**, **elixir**, wonder drug, perfect solution, magic formula, **magic bullet**; More



but remember the skill of the team will outweigh any monolith/microservice choice

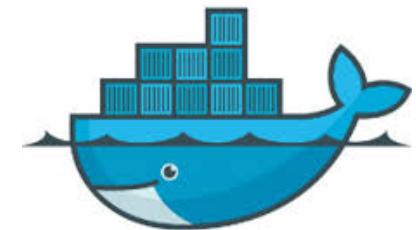
Desafíos

- complejidad puede ser mayor que el de la equivalente appp monolítica
- problemas de gobernanza
- congestión de la red y latencia
- integridad de los datos
- requiere pensar en forma distribuida

Mejores prácticas

- modelar servicios de acuerdo a dominio de negocio
- alta coherencia y bajo acoplamiento
- descentralizar todo (equipos, esquemas, código)
- datos privados para cada servicio
- comunicación a través de apis bien definidas
- api gateway no debe saber nada del dominio

Microservicios y Contenedores



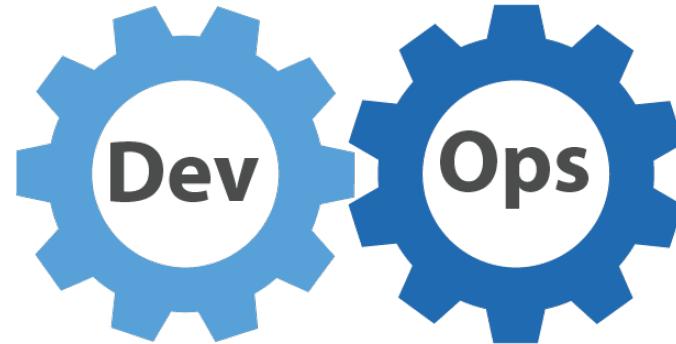
- Un microservicio puede correr en una máquina virtual propia ...
- ... pero hoy en día se ha hecho popular la tecnología de contenedores
- Un contenedor provee todo lo necesario para el servicio siendo mucho mas liviano que una máquina virtual (no incluyen un SO completo)
- La idea es poder mover los contenedores con gran facilidad entre máquinas físicas o en la nube
- Docker es la plataforma más popular para el manejo de contenedores
- Contenedores son una buena manera de desplegar microservicios pero no es la única

Microservicios y DevOps



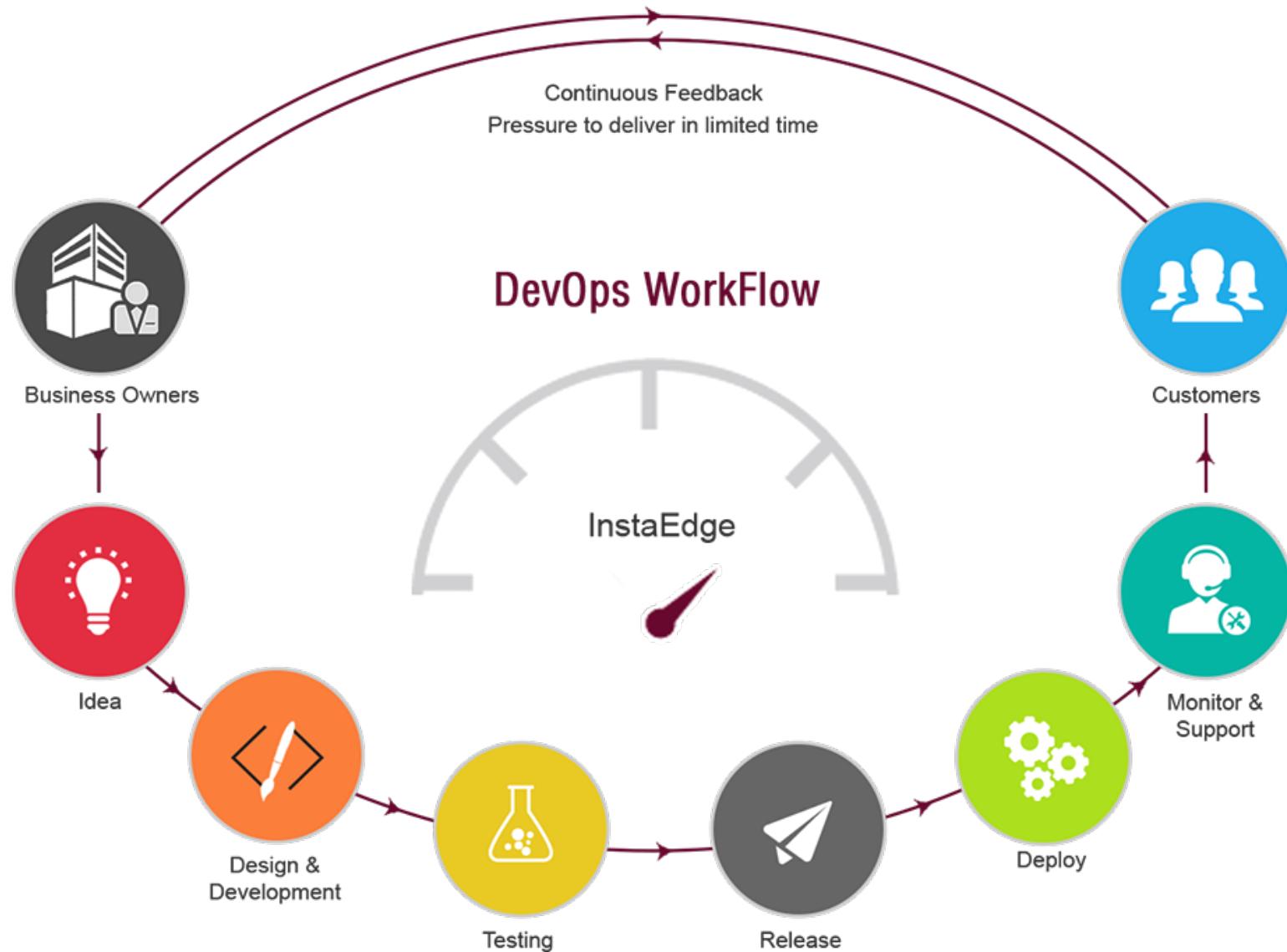
- DevOps - Unir desarrollo de operaciones con el objetivo de acelerar dramáticamente la puesta en producción de nuevas funcionalidades
- Enfoque de microservicios es particularmente apropiado para esto por
 - desarrollo descentralizado
 - independencia del resto de la aplicación
 - puede ser probado en forma separada

La revolución de

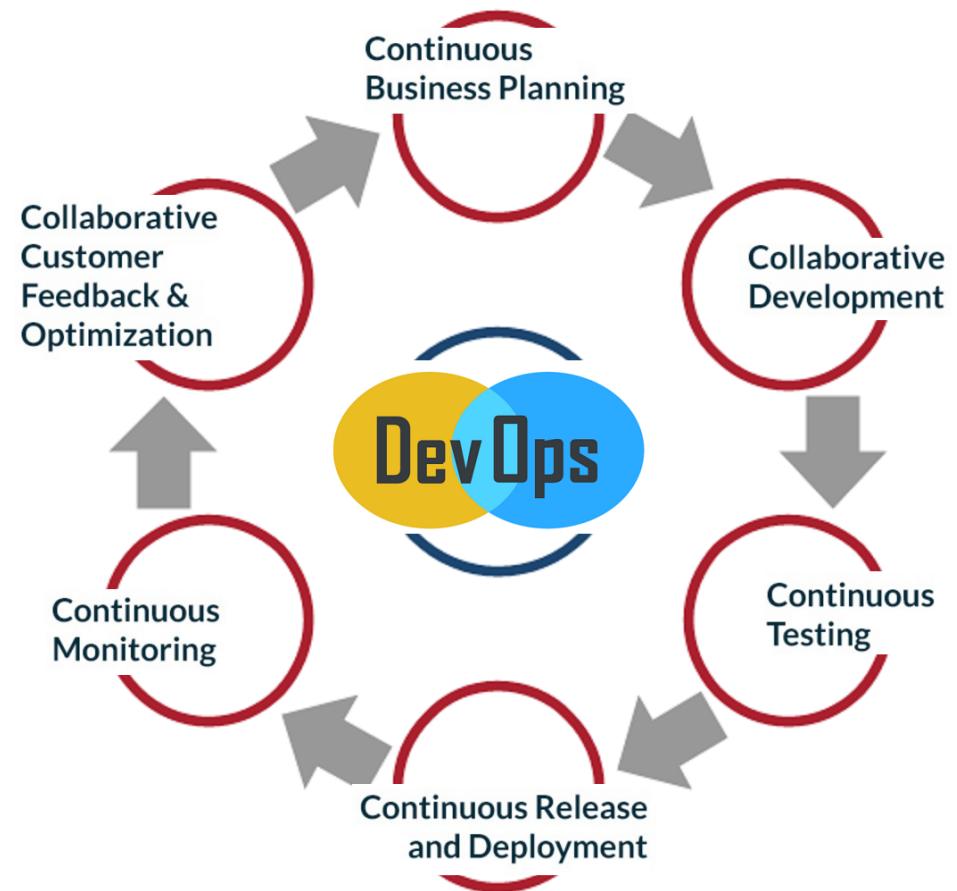


- Una empresa competitiva requiere poner en producción cosas nuevas con rapidez
- Ejemplos:
 - reaccionar a que movistar comienza a ofrecer planes familiares
 - se necesitan nuevas ofertas de roaming mas atractivas
 - un banco sacó el poder bloquear la tarjeta

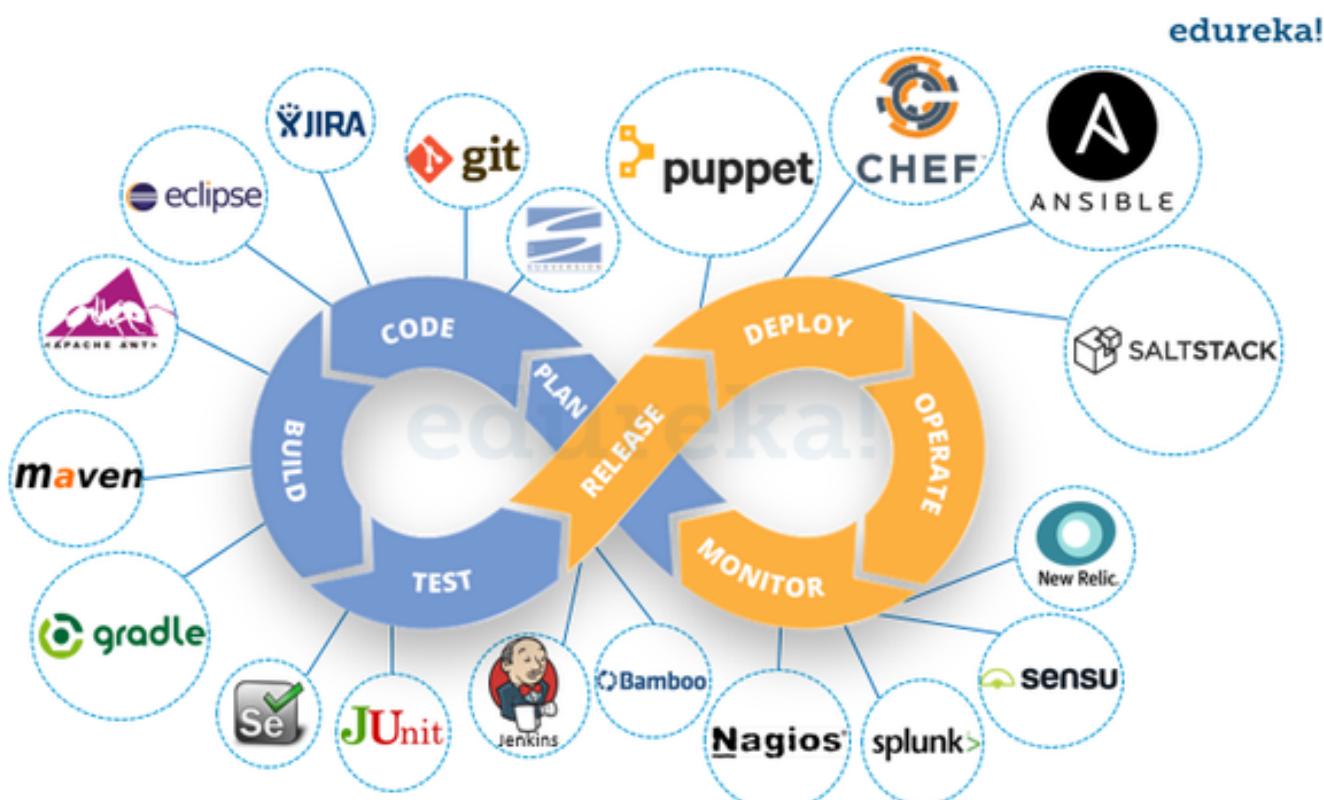
Reaccionando Rápidamente



El ciclo DevOps



DevOps requiere fuerte Automatización



DevOps Tools

